Communication Locality in Computation:

Software, Chip Multiprocessors and Brains

Daniel L. Greenfield

University of Cambridge
Computer Laboratory
Trinity Hall

April 2010

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

This dissertation does not exceed the regulation length of 60 000 words, including tables and footnotes.

Note

This dissertation was originally titled '*Rentian Locality in Chip Multiprocessors*' at submission.

Communication Locality in Computation: Software, Chip Multiprocessors and Brains

Daniel L. Greenfield

Summary

This thesis extends techniques from digital circuit interconnect prediction (in particular Rent's rule) to analyse and predict interconnectedness in software, Chip-Multiprocessors (CMP) and Networks-on-Chip (NoC). Since the birth of the microprocessor, transistors have been getting cheaper, faster and more energy efficient, whereas global wires have changed little. At the same time we are moving towards thousands of processing cores on a chip, with software distributed across them. This new era of communication-dominated computing is marked by local computation on a core being cheap, but with global communication between cores and with external memory as expensive. Thus the physical spatial position of software starts to become important. Indeed, it is shown that unless physical locality in communication is exploited, the costs become untenable with technology scaling.

In VLSI (Very Large Scale Integrated) circuits, the fractal connectivity of Rent's rule is a well known predictor of the physical locality of interconnect across many orders of magnitude. It is shown how a generalised Rent's rule can characterise and model both spatial and temporal locality in software, and it is demonstrated that locality effects can be exploited in Network-on-Chip design for fault tolerance. Evidence of Rentian fractal scaling in software is examined across several benchmarks using multiple methods. Given Rentian scaling, many fundamental results are derived for future many-core CMP architectures that relate number of cores, communication, on-chip memory and the Rent's exponent, including some surprising scaling requirements towards fine-grain communication. It is also shown that existing models of an algorithm's asymptotic time and energy cost are inadequate to account for physical communication costs and locality. A new analytical framework that utilises locality and its Rentian characterisation is demonstrated on several example algorithms, and a study is made of the 'embedding problem' for composing embeddings of algorithms together. Finally, in examining the interplay of communication and massively parallel computation at larger scales, we look at the mammalian brain as a proof-of-existence. We show that Rent's rule also appears to apply to neuronal systems, and that this relates to the allometric scaling of communication to computation.

Acknowledgements

Over the course of this PhD, I have been honoured to be in the company of many outstanding people. First and foremost, I would like to thank Simon Moore, my supervisor, not only for his invaluable feedback, but for his enduring support of these blue-sky research ideas. Thanks also go to Robert Mullins for many stimulating conversations, as well as to Alan Mycroft and PhD examiners Steve Furber and David Greaves for their insightful feedback.

For an incredible journey of collaboration, I'd like to thank Danielle Bassett and Ed Bullmore whose sheer enthusiasm have made it all the more enjoyable. For taking on the Herculean task of checking through proofs, I am particularly grateful for Stéphanie Jacquot's help. Thanks also go to Jeong-gun Lee for many fascinating discussions in my first year, and to Nick Barrow-Williams and Chris Fensch for their research support. I also wish to thank colleagues Arnab Banerjee, Rosemary Francis and Andrew West for many pleasurable discussions at the lab. To Alban Rrustemi, I am truly indebted for all his invaluable help, feedback and moral support.

For all her enduring support through the difficult times of producing this, and all the absurd fun we've had in between, I'd like to thank Rei Kanemura.

My deep appreciation also goes to the Gates community and Trinity Hall, for all the treasured friendships and moments over these years.

This work was generously supported by a scholarship from the Gates Cambridge Trust, who I would like to thank not just for their financial support, but also the fervent intellectual community that they have fostered.

Finally, but certainly not least, I thank my parents who were excited to see me take on this challenge, and realise a long-held childhood dream. I thoroughly blame my father's genes for making me all-too fascinated by science and technology and the spirit of tackling the supposedly impossible. I hope that the ideas here will one-day result in even cooler gadgets for him to play with. But in all seriousness, I would like to deeply thank them both for laying down a foundation of a good education. And especially to my mother, I would like to thank her for all her love and support, and for always encouraging me to pursue my dreams, wherever they may take me.

Contents

List of Figures

List of Tables

Glossary

C.elegans	Short for *Caenorhabditis elegans*, or the 'nematode worm'. This is a model organism often used in biology.
CMP	Chip Multiprocessor, consists of multiple processors on a single chip (or stacked die).
Configuration Space	Concept originally from Physics where configurations of objects correspond to a position in a high-dimensional space. For example the configuration phase space of a particle is the 6-dimensional position correponding to its 3-dimensional spatial position and 3-dimensional momentum.
DSI	Diffusion Spectrum Imaging. A neuro-imaging technique that maps myelinated axons by following the diffusion of water molecules as seen by MRI machines.
Grey Matter	Grey matter corresponds mainly to neurons, their synapses and unmyelinated axons. Computation occurs at neurons and synapses, so grey matter is representative of computation.
NoC	Network-on-Chip. These are communication-networks that allow the general transport of data between different parts of a computer chip.
Rent's Rule	A power-law relationship between the I/O terminals at a boundary, and the amount of logic inside the boundary.
VLSI	VLSI – Very Large Scale Integrated circuits – or commonly referred to as 'computer chips', but with 'very large' numbers of devices/transistors on the chip.
White Matter	White matter corresponds mainly to myelinated axons. These axons are projections for long-distance communication between neurons and protected by a white sheath of myelination, so white matter is representative of communication.

Chapter One 1 Introduction

1.1 Overview

We are entering an exciting new time in computing, where communication costs dominate over the raw computational costs. Old assumptions and models of computational complexity that assume cheap communication need to be seriously revisited in an era when moving a word of data from one part of a chip to another can consume a hundred times the energy of 32-bit arithmetic operations. This thesis argues that it is the communication costs of algorithms, rather than their computation costs, that will dominate future computing concerns. That, as we move to thousands of cores on a chip, the physical spatial locality of computation and data becomes critical to performance and cost. However, there is very little in the way of theory, models or even characterisation of such locality for Chip-Multiprocessors (CMP). This thesis adapts and extends the existing theory and models of wire locality in VLSI circuits to the physical and temporal locality of software running on CMPs. It aims to provide a new foundation for characterising, modelling, predicting and exploiting the communication properties of software, which as we show, exhibits Rentian fractal scaling. In doing so, it lays a new communication-centric foundation for CMP software and hardware, and provides fundamental insights into their continued technological scaling.

1.2 Main Contributions

Here, the main contributions of the thesis are listed. Those in bold are considered to be particularly important contributions of the thesis.

Arguments of how the communication costs of algorithms are becoming more important than their computational costs:

- **Demonstrating that non-local communication patterns such as uniformly-random traffic, and transpose traffic lead to unacceptable growth in communication cost with technology scaling.**
- Showing how available external I/O bandwidth scales poorly per-core, implying that data-independent parallelism is not scalable compared to data-*inter*dependent parallelism.

Examining Rent's rule for the Network-on-Chip domain:

- **A bandwidth version of Rent's rule is introduced, with a corresponding hop-length distribution for traffic that characterises the spatial locality of communication. Also, the scaling behaviour of Networks-on-Chip traffic is determined from this model, and the distribution of expected types of router operations is shown.**
- To exploit this locality, a simple fault-tolerant router is introduced by a wrapper that bypasses all router logic, but allows through-traffic to continue in the same direction.
- **New analytical methods employing the hop-length distributions, are demonstrated to evaluate and compare router designs for fault-tolerance behaviour under Rentian, exponential and random-traffic models. The traffic model is shown to make large differences to the fault-tolerance of designs, and can affect the choice of a suitable router.**

Existing software benchmarks for CMP and single-threaded execution are examined to see if they exhibit Rentian scaling:

- **SPLASH-2 and Parsec benchmarks are rigorously analysed on a 32-node CMP simulator, with many shown to demonstrate Rentian scaling, with better fits compared to other candidate distributions. Hop-length distributions, and average bandwidth for these benchmarks are shown to be predicted well, compared to a null hypothesis model.**
- The dynamic-data-dependence graphs (DDDG) extracted from traces of the MiBench single-threaded benchmarks are analysed for fractal dimensionality by box-counting, where many are again shown to have fractal scaling.
- **Applying Rentian analysis to the temporal domain, where the dynamic-data-dependence graphs are analysed for physical Rentian scaling based on their instruction time, and**

shown to have Rentian scaling and clear evidence of this structure. Some evidence is also shown that the inter-access time distribution is related to the Rent's exponent in the predicted manner.

- The dynamic-data-dependence graphs are analysed for topological Rentian scaling and many are again shown to exhibit Rentian scaling.

New theorems are derived for Rent's rule, extending it to more general domains:

- **Proof of asymptotic equivalence between Rent's rule and power-law distance distributions in homogeneous d-dimensional vector space, thus generalising Rent's rule beyond the 2-D Manhattan metric domain of VLSI, to arbitrary finite-dimensional vector spaces.**
- Cost-universality as applied to distance distributions, where for a given head distribution, the optimal power-law tail exponent is shown to be the same under any strictly increasing cost-function.
- **That optimally embedding in any d-dimensional space is equivalent to embedding in a configuration space – where scaling parameters become independent of dimensionality. This allows Rent's rule to be generalised by extending to arbitrary configuration spaces, including the Spatio-Temporal domain of CMP.**

A model for the scaling characteristics of CMP based on the generalised Rent's rule, using Spatio-Temporal interconnects:

- **Derivation of a Spatio-Temporal CMP model of communication on-chip, off-chip and though a 'Rentian' memory that optimally minimises off-chip communication.**
- **Relations derived of how external I/O bandwidth scales with number of cores, on-chip memory, granularity of communication and Rent's exponent, which should assist in the architectural exploration of CMP configurations.**
- **A comparison is made of caches to the perfect 'Rentian memory', which result in bounds on the performance improvement that a scratchpad memory can feasibly attain compared to caches.**
- Scaling laws for optimum tradeoff between number of cores and memory, given I/O limited performance and growth.
- **A comparison of scaling characteristics of data-independent parallel algorithms versus data-*inter*dependent parallel algorithms that obey Rent's rule, showing relatively large reductions in external I/O bandwidth at the thousand-core era by leveraging interdependent parallelism.**

Asymptotic characterisation of communication in software:

- Demonstrating direct extraction of Rent's parameters from sample algorithms: matrix multiplication, sequential Fibonacci, Fast-Fourier Transform.
- **The asymptotic cost of communication is derived for Rentian software, given the Rent's exponent and embedding domain.**
- **It is shown that, in software, optimal embeddings do not themselves compose to form optimal embeddings, thus posing the question of whether Rentian scaling properties can be practically preserved upon composition. This is resolved by proving that hierarchically composed/decomposed embeddings, although not optimal, preserve Rentian scaling at only a (small) constant factor penalty compared to optimal embeddings.**
- A model of spatially distributed data accesses, showing how the power-law distribution of memory-access can restore constant factor or logarithmic average access costs, depending on the exponent.

Examining Rentian scaling in neuronal networks – as a natural parallel computing system at scales beyond existing technology:

- **Demonstrating that the human brain and the nematode worm have neuronal networks with Rentian scaling.**
- Illustrating that small-world networks are fully compatible with fractal Rentian scaling with a worked example.
- **Deriving a relationship between Rentian scaling parameters within a brain, to predict the allometric scaling properties of white-matter versus grey-matter across a range of mammalian brains, and showing that the parameters extracted for Rentian scaling within the human brain correctly predict the allometric scaling properties across mammalian brains.**

1.3 Organisation

The research that comprises this thesis was not arrived at as a simple linear exploration. New analysis lead to new insights, and old ones had to be revisited. Whilst every effort has been made to make chapters have a consistent theme of exploration, this has come at the expense of ordering. In particular, the concepts of spatio-temporal interconnect are introduced in Chapter 5 and used in Chapter 7 for derivation of a Spatio-Temporal Rentian model for CMP. However, this model is tested experimentally on benchmarks in Chapter 5.

Chapter 1 comprises this overview.

Chapter 2 argues that communication costs rather than purely computational ones will start to dominate power, performance and other measures with technological scaling. It shows how communication costs in the Chip Multiprocessor (CMP) domain are fundamentally different to those in the traditional parallel-computing domain, and that old intuition and solutions from the parallel-computing domain lead to poor, unscalable solutions for the CMP domain.

Chapter 3 discusses existing models of computation, of communication, and discusses the role of the fractal scaling known as *Rent's rule* in VLSI.

Chapter 4 argues for a bandwidth version of Rent's rule for communication over a Network-on-Chip, and examines its implications for Network-on-Chip design and fault-tolerance.

Chapter 5 examines experimental evidence using multiple methodologies and shows that many existing software benchmarks obey the fractal scaling of Rent's rule.

Chapter 6 takes the theory of Rent's rule beyond VLSI placement. It consists of theorems showing that Rent's rule is asymptotically equivalent to power-law-tailed distance distributions. It shows that Rent's rule applied to homogeneous spaces preserves an invariant in an equivalent configuration-space. Preserving this invariant allows a consistent generalisation of Rent's rule beyond homogeneous two- or three-dimensional spaces.

Chapter 7 utilises the generalisation from Chapter 6 to derive expressions for the scaling characteristics of CMP, based on the number of cores, the Rent's exponent of software, the amount of on-chip memory, and the granularity of communication. Surprisingly as the number of cores increases, there is pressure for inter-core communication to become more fine-grained.

Chapter 8 demonstrates how one can calculate the asymptotic cost of communication for sample algorithms by direct analysis, supplanting usual measures of computational complexity. It introduces the 'embedding problem', whereby optimal embeddings do not themselves compose optimally, and gives a simple solution that allows algorithms to be composable, and yet also preserve Rentian scaling. It then derives lower-bounds for the spatial-communication component of memory accesses.

Chapter 9 looks at the brain as a proof-of-existence of parallel computing system at scales beyond existing technology, and demonstrates that the human brain also obeys Rent's rule. Furthermore, it shows how Rentian scaling has been preserved across the evolution of mammalian brains, as the allometric scaling of white matter to grey matter across mammals can be predicted by the self-similar Rentian scaling within the human brain. In the computing domain, this implies that Rentian scaling will likely continue to govern communication in computation at far larger scales of integration.

Chapter 10 briefly concludes and discusses potential future work.

Appendix A contains plots of the SPLASH-2 and PARSEC benchmark applications showing best-fits of Rentian and alternative models and a comparison of their predicted versus actual hop-length distributions.

Appendix B comprises plots of box-counting fractal scaling behaviour for the MiBench benchmark applications.

Appendix C comprises plots of physical Rentian scaling and their temporal-distance distributions for the MiBench benchmark applications.

Appendix D comprises plots of topological Rentian scaling and their temporal-distance distributions for the MiBench benchmark applications.

Appendix E describes in more detail the model used to determine best-fit parameters for the Spatio-Temporal Rentian model of a 32-core system, the results of which are in Chapter 5 and Appendix A.

Appendix F lists source code for numerically evaluating the Spatio-Temporal Rentian model for I/O bandwidth versus Rentian memory seen in Chapter 7.

1.4 Publications

Peer reviewed publications

D. Greenfield, A. Banerjee, J-G. Lee, S.W. Moore, *Implications of Rent's Rule for NoC Design and Its Fault-Tolerance*, First International Symposium on Networks-on-Chip, May 2007

(Contributes to Chapters 2 and 4)

D. Greenfield, S.W. Moore, *Fractal communication in software data dependency graphs*. Symposium on Parallel Algorithms and Architectures, June 2008

(Contributes to Chapter 5)

D. Greenfield, S.W. Moore, *Implications of Electronics Technology Trends to Algorithm Design.* 'Visions of Computer Science' BCS International Academic Conference 2008, *and also as:* The Computer Journal, British Computer Society, Volume 52, Number 6

(Contributes to Chapters 2 and 5)

D. Bassett, **D. Greenfield (joint first author)**, A Meyer-Lindenberg, D. Weinberger, S.W. Moore, E Bullmore, *Efficient Physical Embedding of Topologically Complex Information Processing Networks in Brains and Computer Circuits*, PLoS Computational Biology, April 2010

(Contributes to Chapter 9)

Invited paper

S.W. Moore, **D. Greenfield**, *The Next Resource War: Computation vs. Communication*. System Level Interconnect Prediction, June 2008

Chapter Two

Motivations: communication vs. computation

2

For most of the history of computer science, computation has been 'expensive' and communication practically 'free'. This fundamental, and yet implicit assumption, underlies our analysis of algorithms, and guides our intuition when constructing solutions to problems in Computer Science. Unfortunately, this assumption has been eroded with technological scaling as we can no longer 'hide' the underlying laws of physics. Indeed, the tables have turned – as we shall show, it is now communication that has become expensive whilst computation is practically 'free'. We shall show that looking forward, it is the communication of algorithms rather than their computation that will become the dominant factor in energy consumption, cost and performance metrics in the many-core era.

2.1 Broken assumptions: no escape from Physics

The exponential growth of transistor density, as predicted by Moore's law, has allowed transistors to scale over the last 35 years from dimensions of 10,000nm (Intel 4004) to 45nm (current Intel and AMD processors). This dramatic growth in transistor density has allowed processors to scale in complexity from the 2300 transistors of the Intel 4004, to the 731 million transistors of the Intel Core i7 (including cache). During this period, however, chip sizes have largely been unchanged at about 10-20mm on a side, depending on cost. While transistors have been getting smaller and faster, the wires that connect one region of the chip to another, have not scaled. Relative to transistors, they have gotten exponentially worse in speed, power, and manufacturing cost.

In table 2.1 we see how technology scaling affects the relative cost of communication versus computation. This table is adapted from a presentation given by Dally [34]. We can see that communication is no longer inexpensive – thanks to decades of technological scaling, even at the older 130nm scale, merely transferring 32-bits of data from one

technology node	130nm CMOS	45nm CMOS
transfer 32b across-chip	20 ALU ops	57 ALU ops
transfer 32b off-chip	260 ALU ops	1300 ALU ops

Table 2.1: Comparing trends in energy consumption, adapted from Dally [34]

part of the chip to another consumes the equivalent of 20 ALU operations. Moreover, at current scales of 45nm, it is even worse, and this disparity will continue to grow exponentially with technological scaling. Furthermore, we see that in order to transfer data off-chip, the costs become even steeper, with large transistors required to drive the pins. To some extent, new technologies can help mitigate these, but as we shall see, they do not halt the larger trend. What this means is that communication, both on-chip and particularly off-chip, can be many orders of magnitude more expensive than computation.

Computer Architects have done a remarkable job insulating software engineers from the changes due to technological scaling. They have kept the illusion of ever faster serial-processors alive until recently, when diminishing returns to instruction-level parallelism, excessive power-consumption costs and slow on-chip interconnect, have resulted in abandoning this illusion, and forced software engineers to move to multi-core scaling. Whilst computer scientists rush to cushion this blow, by trying to make parallel programming easier, they need to be aware that more challenges from technological scaling are waiting in stall, that they cannot be insulated against.

With continued technological scaling, we are transitioning into an era with thousands or even millions of cores. With so many cores, one might even say that ‘processors are the new transistors’. Software and instructions can no longer be thought of as operating in some sort of Platonic realm. Software, and data, instead have a physical spatial position. Instructions are actually executed at some physical position, and data needs to be stored and retrieved from a physical position and transported to another physical position. This has profound implications for the future of software design.

Suppose two algorithm designers Alice and Bob construct their own data-structure to store and retrieve information. As there is too much data to fit on one core, it necessarily spills over to the memory of surrounding cores (or even a neighbourhood of memories, off-chip). This can be much like the binary tree traversal in figure 2.1. Here colours correspond to the level of the tree. Although binary tree traversal is thought of as $O(\log N)$ this is only the case if communication costs are ignored. A physical implementation necessarily has the data reside somewhere, and this limits the spatial locality. If the nodes are placed randomly but in the 2-D neighbourhood of the root node, then on average each traversal of the tree takes time $O\left(\sqrt{N}\right)$, resulting in a total time of $O\left(\sqrt{N}\log N\right)$. If nodes are carefully placed in two dimensions (such as at the bottom of figure 2.1), then the average distances between levels of the binary tree hierarchy grows exponentially, but traversal takes time $O\left(\sqrt{N}\right)$. Thus even if Bob carefully produced an algorithm that has

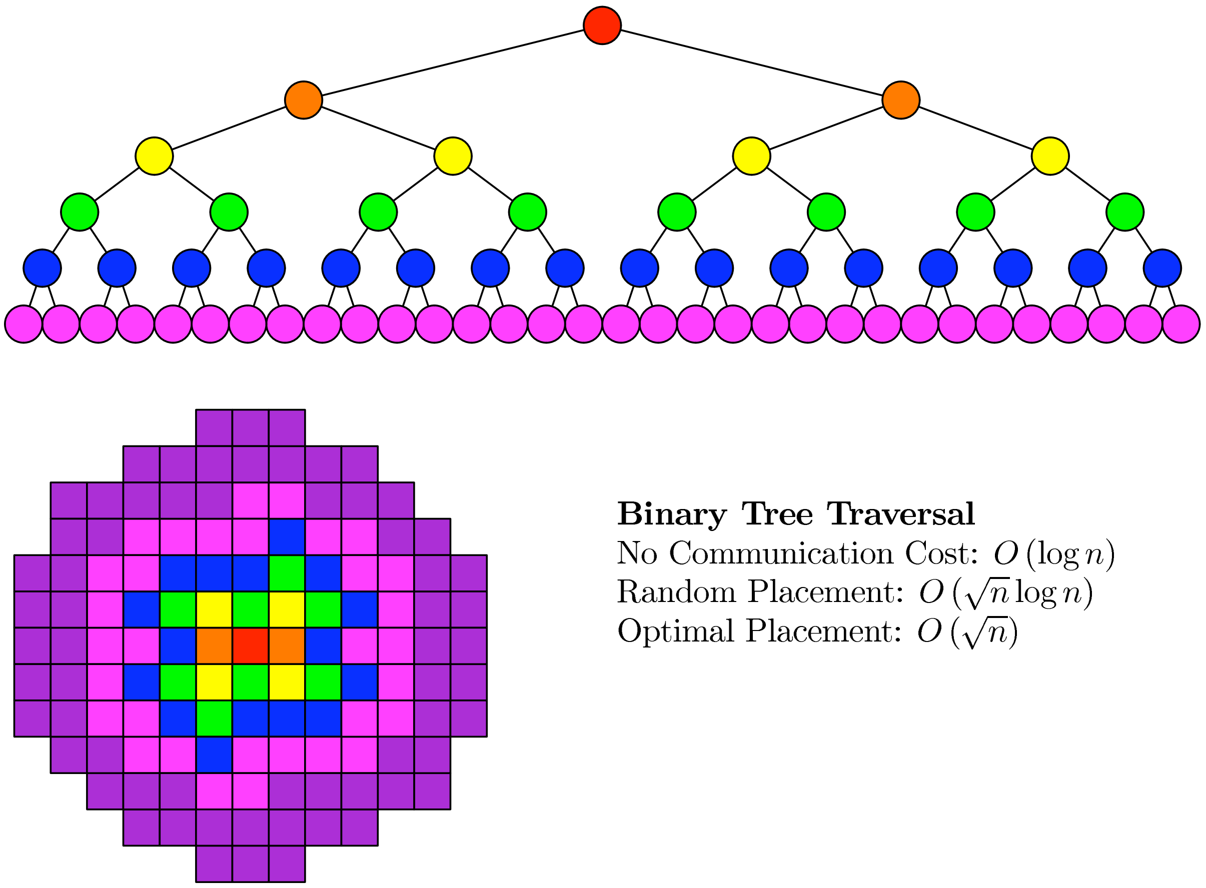

Binary Tree Traversal
No Communication Cost: $O(\log n)$
Random Placement: $O(\sqrt{n}\log n)$
Optimal Placement: $O(\sqrt{n})$

Figure 2.1: Algorithms and data-structures must be physically embedded in space (here the 2-D space on-chip). Asymptotic tree traversal costs are shown for binary tree placements, assuming that traversal costs are proportional to distance.

computational complexity $O(\log N)$, Alice could produce a poorer computational complexity algorithm of $O\left(\sqrt{N}\right)$ but focused on communication costs, and asymptotically outperform Bob's algorithm. Indeed, in general, Alice will do no worse than Bob by focusing on communication complexity, but can also do better than Bob as here. We will show in Chapter 8, that the asymptotic scaling of communication costs is at best equal to the asymptotic scaling of computational costs, and can typically be worse. This may mean that communication complexity may one day replace computational complexity as the measure of asymptotic scaling in algorithms.

2.2 The many-core memory bandwidth wall

We noted (see Table 2.1) that communication power off-chip is not scaling with transistor performance. Despite this, rather than using less I/O bandwidth, demand for I/O bandwidth is growing, and technology is not keeping up. Figure (2.2) shows the estimated growth for total chip I/O bandwidth based on ITRS predicted figures [59] for pin-count in their 'cost-performance balanced' model and the predicted bandwidth growth of high-

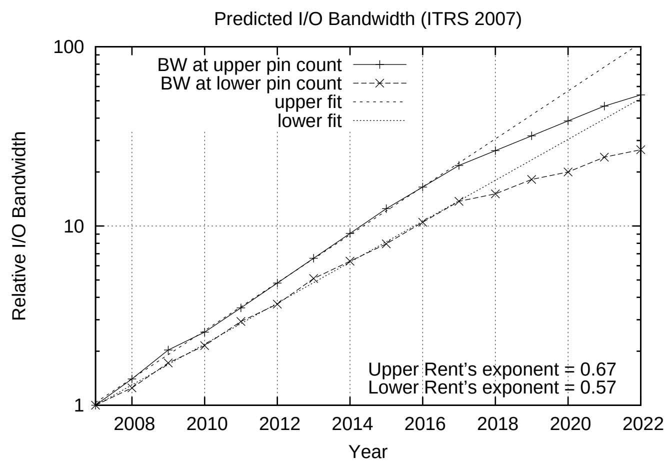
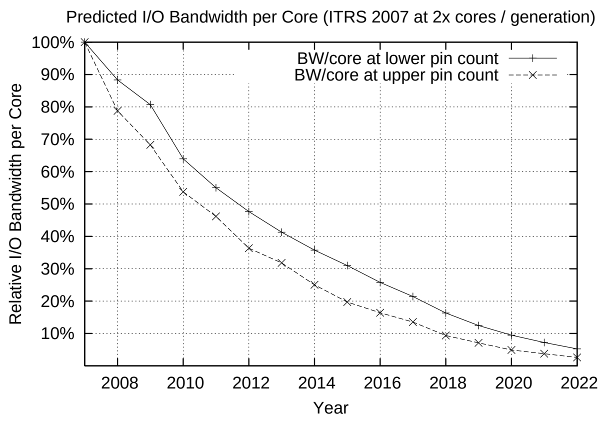


Figure 2.2: ITRS I/O Bandwidth Growth



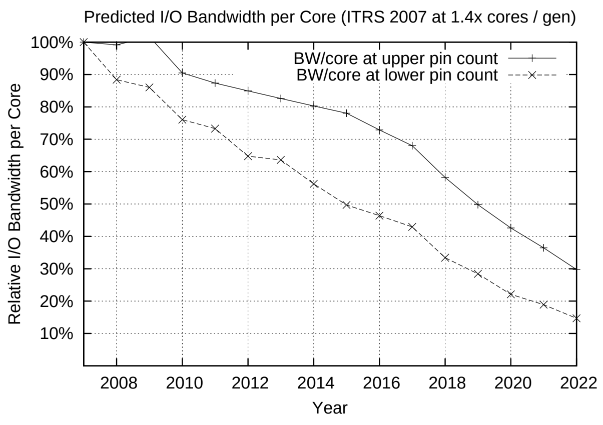


Figure 2.3: Predicted scaling of I/O Bandwidth per Core using ITRS numbers, based on 2x cores per generation or 1.4x cores per generation

speed pins. We note that after 2017 the growth changes to a more conservative trend as manufacturable solutions for these are not currently known. The rate of growth can be shown by fitting a range of Rent's bandwidth-exponents (Rent's rule is discussed in Chapter 3, and the bandwidth version explained in Chapter 4). These are estimated for the years up to 2017, yielding exponents between 0.57 to 0.67, depending on pin-count growth rates.

We should note that these numbers may not account for the power and thermal constraints that are also present on the system that may prevent pin counts and pin bandwidth from growing as fast as this. Also, the size of I/O pads and drivers do not tend to change much with each process generation, so the exponential growth in pin count may have other costly repercussions for die size. Optical off-chip communication may help improve available bandwidth, which is discussed further below, however it is unlikely to be a panacea.

From the predicted I/O bandwidth (figure 2.2) we can derive the external memory bandwidth per core (figure 2.3), available on average to each core. The transistor density doubles every technology generation and this has led some people to believe that the number of cores could also double each process generation. However the ITRS deems the number of cores doubling per *two* process generations to be more representative of current power-limited design trends, and this is what is currently seen in the marketplace. Both trends are shown, and we observe the exponential drop in available bandwidth that can impose a significant bottleneck in performance. This shows that cores cannot feed their bandwidth from external I/O and instead communication from other cores on-chip is required to make up the shortfall. We also note that this external bandwidth has considerably larger communication latencies and consumes more power than on-chip communication. Indeed, a reason why cores cannot scale linearly with transistor growth is because larger on-chip memories are needed to reduce latency and external I/O bandwidth demands. In Chapter 7, we shall examine this in more detail.

Benia et al. [2] characterised the diverse PARSEC benchmark suite, which includes emerging applications in Recognition, Mining and Synthesis (RMS) that Intel have declared as important future workloads [53]. They simulated on 1 to 16-core CMP models and noted that each core has high bandwidth demands, with algorithms operating on large working-sets that far exceed on-chip memory. They showed that the bandwidth demands grow approximately linearly with the number of cores (or worse), so that an exponential growth in cores means an exponentially growing demand in I/O bandwidth. As such they also predict that the external I/O bandwidth of CMPs will be '*their most severe limitation of performance*'.

There are interesting implications for systems and software programming. The notion that if we had hundreds of cores that we would merely run hundreds of separate applications is not sensible, even if users wanted to do so, because their external communication needs would be unlikely to scale. Algorithms which assume simple independent data-parallel operations distributed across cores also effectively operate like entirely separate cores that need their own allocation of I/O bandwidth. While a lot of focus in parallelism so far has been on finding or specifying *independence* between data, especially in loops, and exploiting this as data-level parallelism, we see that this strategy may not be scalable to large numbers of cores due to these external I/O constraints. Instead we need to exploit *interdependence*. In traditional multicore architectures, external memory is effectively employed as a large communication crossbar. Indeed, by using such an approach, many algorithms can be factored into simple data-parallel operations with external memory taking care of communication complexities / interdependencies. In other words, the strategy many are adopting is to push out interdependencies to memory. For example, each computation stage of the Fast Fourier Transform is independent and data-parallel, however the communication between stages is complex and (external) memory is often used to act as the communication medium. Similar approaches are used for matrix operations and many other algorithms. This factoring of algorithms into batch jobs of independent computation which are glued together by (external) interdependent memory operations is not scalable. Instead, to achieve scalability and locality, we believe that *interdependence* needs to be exploited on-chip with more local computation and communication between cores required. In short, algorithms must exploit communication locality between cores and threads so as to avoid communicating off-chip, and to minimise communication costs on-chip.

Stream processing goes part-way to addressing this by factoring stream-processing stages into cores with explicit communication between them. That is, their interdependence is explicitly mapped into a linear (one dimensional) or near-linear inter-core communication graph. However, with a growing number of cores, this block level partitioning only goes so far. Additional techniques are needed for further utilisation of cores with efficient inter-core communication.

For loops, software pipelining [82, 69] is an approach that essentially takes the dependency graph of operations in a loop and distributes them across cores to form a pipeline across multiple tiles. It is an effective technique for fine-grain parallelism.

Affine Partitioning [70, 71] is another approach that partitions algorithms with affine dependencies, typically within nested loops. It does so by converting the affine dependency structure into a set of linear equations and finding provably minimal communication solutions. The solution results in an instruction sequence in space (across a mesh of tiles) and in time (by cycle count) with a set of communication patterns. Software pipelining can then further extend the parallelisation.

Not all parallelisation approaches, however, result in partitionings with better internal versus external communication utilisation. Thread Level Speculation (TLS) [96] techniques on loops rely on very high *independence* between blocks of loop iterations for speedup, and thus appear very close to data-parallel in nature. They then allow for some interdependencies by aborting and restarting threads upon encountering dependency conflicts, with obvious penalties. Thus when operating on large datasets, such TLS techniques may also be significantly limited by external I/O bandwidth.

As external bandwidth, power and latency constraints motivate us to further exploit internal bandwidth, we envision that many more techniques will be developed for *interdependent* communication and parallelism.

2.3 Networks-on-Chip and traffic patterns

On-chip communication has evolved along with technological scaling, with its changing nature of computation vs. communication costs. This is illustrated in figure 2.4. Whereas, it used to be simple to merely link source and destination with a single wire (top of figure), as clock frequencies increased, and the demand for wire bandwidth grew, the quadratic *RC* delay of wires due to their resistance and capacitance per unit length, became problematic. Repeaters were inserted to break up long wires, constraining wire delay and power-consumption to grow linearly with distance. As wire bandwidth demands grew even further, interconnect became pipelined, thus allowing multiple symbols

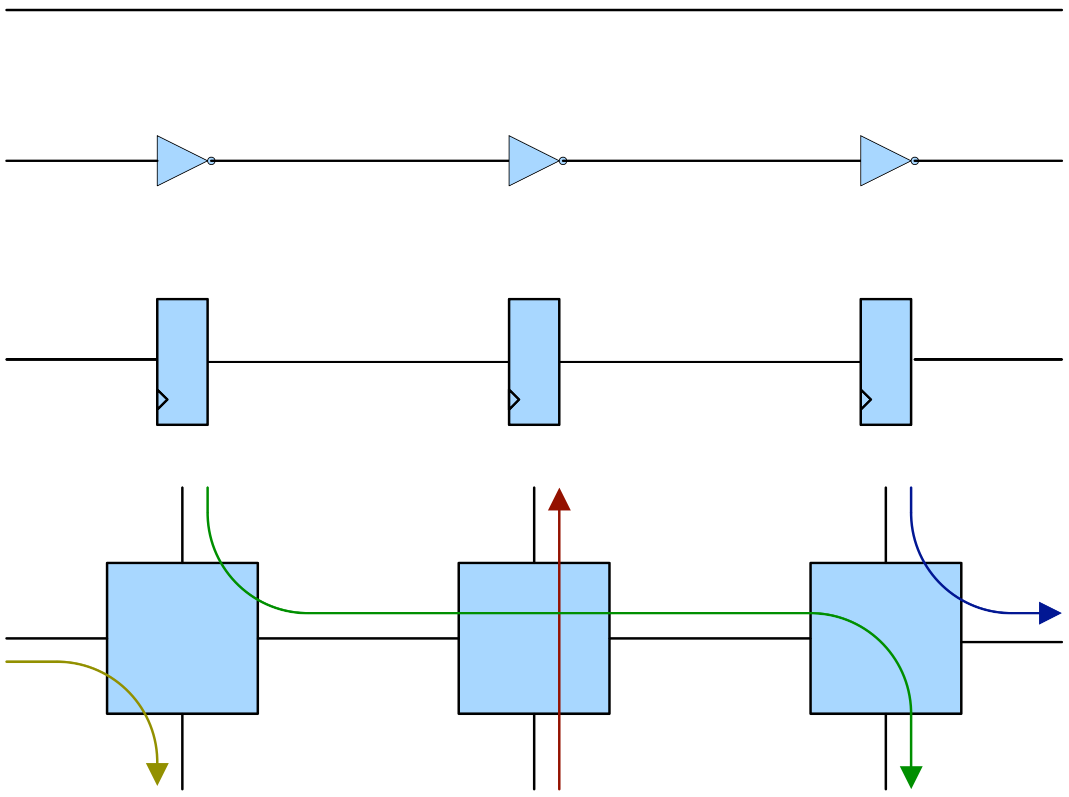


Figure 2.4: Virtualisation of interconnect. From wires, to buffered wires, to registered wires, to Networks-on-Chip.

to be in-flight at the same time without risk of inter-symbol interference. Now, the point-to-point interconnect is giving way to a more general communication fabric – Networks-on-Chip [35] with point-to-point circuits or packets. This is seen at the bottom of figure 2.4, where symbols can be routed by the network from multiple sources and destinations. Wires can then be re-used by multiple pairs of source and host instead of just a dedicated link. As one can see, the general trend has been to employ more computational resources to better utilise the relatively exponentially growing cost of on-chip communication.

Some may argue that the current move towards parallel computing connected by a network can merely utilise the High-Performance Computing (HPC) research conducted in the 90s and adapt it to the on-chip domain. Because of the very different nature of on-chip communication costs, this is absolutely not the case, and Chapter 3 addresses this in more detail. Nonetheless, many Networks-on-Chip designs are evaluated and analysed using methods taken from the HPC domain that are inappropriate for the on-chip domain. For example, let us look at the very commonly used uniformly-random traffic and transpose traffic patterns that are employed to test designs.

Let us consider an array of tiles connected by a mesh network, where each tile generates on average one unit of bandwidth. We want to consider the effects of technology scaling on the NoC. As we move to newer process technologies, the clock frequencies of the Processing Element (PE) and NoC may also increase, but not necessarily by the same

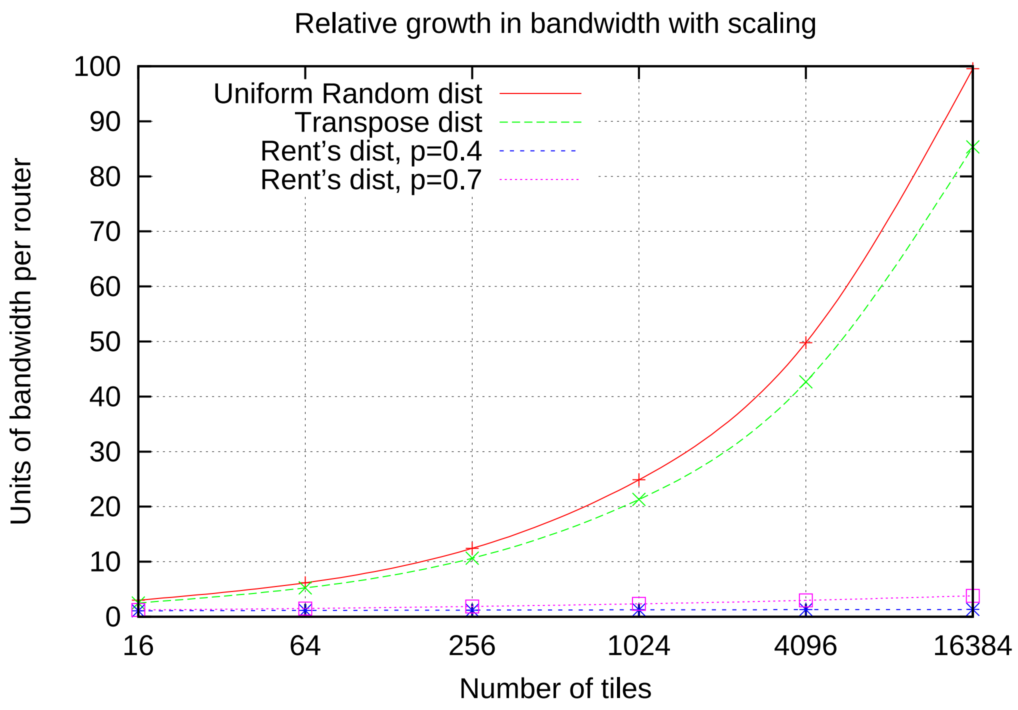


Figure 2.5: Scaling behaviour of *bandwidth per router* for different traffic distributions. The scale is such that, on average, one unit of bandwidth is generated per tile.

amount. For our analysis, we will optimistically assume that the NoC frequency scales by the same amount as the PE frequency. However, one must also take into account the scaling behaviour of the traffic over the NoC. Uniform random and transpose traffic are commonly used to assess NoC designs, so it is instructive to look at the implications of such traffic behaviour. Uniform random traffic occurs when each tile has an equal probability of communicating with any other tile. Whereas transpose traffic has the tile at position (x, y) communicate with that at (y, x).

Figure 2.5 shows the average bandwidth requirements *per router* with scaling. This should not be confused with the bi-section bandwidth, which grows even faster. Here, the amount of traffic generated by each tile is a single unit at all scales, and thus does not include the further effects of clock frequency scaling. We can clearly see that for these traffic patterns each router has exponentially growing needs, doubling per quadrupling in tiles, so even if the router link bandwidth did scale with the PE frequency, the number of links required between routers would need to grow exponentially to keep up. Each new link corresponds to more physical wires per router, which translates to an exponentially growing number of metal layers. Now, one might argue that different NoC topologies, such as a hierarchical mesh, hypercube, or other complex topology, might be able to handle this better. Indeed, in the HPC domain, at sizes insulated from the physics of technological scaling, this strategy works very well. Unfortunately in the on-chip domain, regardless of the topology, different NoC topologies do not change the *physical* distance data must travel, nor the quantity of extra physical wiring still needed to transport the extra flow of data. Thus they cannot circumvent this exponentially growing need for metal layers and bandwidth. Indeed, another way of interpreting figure 2.5 is that the metal layers for NoC routing also need to double per quadrupling in tiles, with corresponding power and performance implications. We have already discussed the growing relative cost of on-chip communication – so we can see that a rapid exponential growth in traffic *per core*, on top of an exponential growth in the number of cores, makes such non-local communication patterns extraordinarily expensive.

Clearly, such scaling behaviour is unacceptable, but it is a simple consequence of assuming uniform random or transpose traffic patterns. Indeed, if logic cells in a VLSI design were randomly placed, the amount of wiring and metal layers required would scale in a similar fashion. The reason this behaviour does not occur in the VLSI domain is because *locality* is exploited by placing connected cells close to each other, thus minimising wiring and improving timing and performance. Indeed, also seen in figure 2.5 are plotted the expected growth in bandwidth *per router* if Rentian locality is exploited instead, which is in line with the regular growth in VLSI interconnect demands. Rentian locality will be discussed more in Chapters 3 and 4. For small arrays, non-local traffic such as the random-uniform model, might still be tolerable. However, as we move into hundreds or thousands of tiles on a chip, the exponentially increasing cost of such non-local traffic patterns makes it impractical, and thus exploiting locality in communication becomes *essential*.

2.4 New interconnect technologies

A number of new interconnect technologies are on the horizon to potentially help tackle these communication challenges [59]. These technologies are discussed is more detail below. Unfortunately, the benefits of these technologies are not scalable, and so represent only one-off improvements of up to a single order of magnitude, and with significant costs in terms of manufacturing and integration. It is possible that other forms of interconnect may arise in future, however, they are likely to be limited resources in much the same way as existing interconnect is. Longer interconnect necessarily has a larger cost in terms of propagation delay, interconnect ‘wiring’ cost, switching resources and their associated energy consumption, than a shorter interconnect does. Thus, we believe that there will continue to be an incentive to reduce communication costs by exploiting locality, regardless of any future interconnect technology breakthroughs.

2.4.1 Optical

For off-chip communication, optical interconnect is a promising candidate to existing solutions, although with some challenges still ahead [59]. It is likely to reduce power consumption, in a one-off fashion, by up to an order of magnitude. We shall focus, however, on on-chip optical interconnect.

We should also point out that optical interconnect necessarily runs as a circuit-switched medium. Although the energy consumed for transmission is largely independent of distance, longer circuits take up more switching resources than shorter circuits, preventing the rest of the system from utilising these switching resources. Thus there is a resource cost (and associated power cost) for longer interconnects compared to short ones. Moreover, optical ring resonators are relatively bulky units that can only steer either individual wavelengths or bundles of wavelengths together as a set. Unlike an electronic switch which can steer traffic from multiple sources in an arbitrary fashion to multiple destinations, the optical version needs time to warm or cool so as to match the target set of wavelengths, and then can only steer that set of wavelengths whilst leaving the other sets of wavelengths untouched. This greatly restricts the flexibility of optical interconnect, compared to electronic ones. The waveguides that serve as the equivalent of ‘wires’ for an on-chip interconnect, are necessarily wide ($1 - 4\mu m$) as they must support the relatively large wavelengths of photons compared to electrons. By using high refractive index waveguides it is possible to use smaller width waveguides (e.g. $1\mu m$) but necessarily at the expense of a slower speed of light (e.g. ~1/3) in that medium [50]. These width constraints mean that optical interconnect does not actually scale along with new technology nodes – their size remains constrained to top-level global interconnect.

One of the primary motivations for utilising on-chip interconnect is to try to reduce interconnect power-consumption, however recent work by Dokania and Apsel has challenged the view that on-chip optical interconnect could achieve this goal [38], even with off-chip light sources that are merely modulated on-chip. After taking into account the energy needed for the thermal regulation of these on-chip modulators, they found that any energy savings over traditional wire-based interconnect are effectively nullified.

2.4.2 LVDS transmission lines

Low-voltage differential signalling is already used for off-chip communication, however they are not, as yet, utilised on-chip. Recent work by Ito and Mineyama et al. [58, 75] constructed current-mode on-chip LVDS transmission lines with carefully designed passive equalisation. They demonstrated an order of magnitude reduction in energy consumption, and improvement in latency. Their 90nm CMOS implementation of length 5mm, was capable of just over 10 Gbps. We should note, however, that the interconnect, transmitters and receivers take up a large amount of area – and that this area does not shrink with technology scaling. The LVDS wires themselves take up a very large width of $20\mu m$ including spacing, making them suitable only for top-level global interconnect. This compares poorly to the current $0.14 - 2.0\mu m$ global interconnect pitch width at the 45nm technology node [59]. Compared to optical, Mineyama shows that [75] it consumes even less power at these distances. This may make them a considerably more attractive option than on-chip optical communication. Nonetheless, much like optical, the benefits are likely to be one-off – they do not improve much with technology scaling.

2.4.3 Carbon Nanotubes

Carbon Nanotubes are also being seriously considered for integration [59], although they currently pose a number of manufacturing challenges. They have superior conductivity for longer length interconnects compared to current copper interconnect, as well as excellent thermal conductivity and do not suffer from electromigration concerns. Their likely power benefits, however, are small and one-off. They are merely a replacement material for copper interconnect, as they are still electrical wires with resistance and capacitance, even if not made of metal.

2.4.4 Surface plasmon resonance

Surface plasmon resonance avoids the large wavelength width requirements of photonic interconnect, potentially allowing a much higher density of interconnect and even replacing shorter, intermediate levels of on-chip interconnect. In this technology, electromagnetic waves propagate along the boundary of metal and a dielectric medium. By tuning the frequency of light to match the resonant frequency of plasmons, the light energy can effectively be converted into a resonant plasmon that propagates along the surface, whereupon encountering an irregularity it can be scattered back into light. A lot of work has gone into manufacturing the small flawless interfaces needed for this form of communication. Unfortunately recent theoretical work by Marklund et al. [74] has shown that even with a flawless interconnect, this form of propagation may have even worse scaling characteristics at nanometre scales than ordinary wires, as the damping length varies according to the fourth power of wavelength.

2.5 Summary

In short, thanks to technological scaling, communication is becoming exponentially more expensive than computation. Moreover, non-local communication patterns will become prohibitively expensive as we move to many cores on a chip, since data and executing instructions have physical positions on the chip. Instead, we desire to minimise the cost of communication by exploiting as much communication locality as possible. The problem is, there is not much understanding of communication locality, particularly physical spatial locality, in software. We qualitatively know that temporal locality is important for cache behaviour, but what are at its foundations, and what can we expect in software? How can we quantitatively model locality or predict it? This thesis aims to address these fundamental questions for the new era of communication-centric computation.

Chapter Three 3 Background

There has been abundant prior work in trying to model or characterise communication, primarily in the context of how communication affects other measures such as time complexity (performance) in computation, or area in digital circuitry. Here we discuss these and their limitations, before delving into the more practical VLSI wiring characterisation known as Rent's rule.

3.1 High-performance computing

Parallel processing has been around for a long time, and much research has already been done in trying to extract ever more performance out of them. Generations of high performance parallel computers have been built using a collection of individual single-chip processors coupled to a standard network interface or customised router, and linked together by a communication topology such as a hypercube. These routers have traditionally been slow and bulky compared to the delay of links between them. Although the absolute cost of communication can be high, the physical scales at which communication takes place is large (dedicated router chips orders of magnitude bigger than individual area-constrained on-chip routers), where the cost and delay is dominated by the routers themselves and their communication with the processor, rather than the actual transmission of data over the links, that are typically transmission lines / cables. This means that the topological distance, i.e. the number of router hops between two nodes, dominates cost over actual physical distance. The high performance parallel processing domain therefore admits very high dimensional and complex topologies that can greatly reduce its communication costs and improve performance. Indeed, high-dimensional hyper-cubes are popular due to their ability to easily embed other problems [48].

At the scale of individual NoC routers on a chip, however, there is considerably greater cost in connecting them into high dimensional topologies. Thus the intuition gained from the parallel computing community of building complex routers or non-local topologies needs to be treated carefully and skeptically. Primary concerns about load-balancing of computation and traffic do not necessarily apply in the CMP domain, where power constraints are more dominant.

3.2 Existing models of parallelism and communication

There have been a number of models of computation with which parallel algorithms have been studied. For the most part, however, these have been mainly applicable for the High Performance Computing domain. Apart from Billardi's work on D-BSP, these models typically do not account for the energy and delay costs of moving data that *increase* with its *physical distance*, although Valiant makes his Multi-BSP model highly parameterisable so that these attributes *can*, in theory, be incorporated by choosing suitable parameters. Moreover, these models have nothing to say about the expected properties of communication locality, whether in physical space, or in time.

3.2.1 PRAM model

The PRAM model [42] is a parallel version of the RAM computation model. It is widely adopted by algorithm designers to analyse the performance of algorithms, however it is also a very simple model that assumes an infinite number of processors, and that memory is uniformly accessible in constant time, regardless of how it is shared, and has no resource contention. As such it assumes ideal communication, and does not account for the non-uniform communication costs of real systems, or their memory access costs. It works best in the domain where communication and synchronisation costs are negligible compared to computational costs.

3.2.2 BSP model

The *Bulk Synchronous Parallel* model (BSP) model of Valiant [106] was introduced in 1990 and is also widely used for algorithm analysis. This model is considerably more realistic. It consists of a set of p processor-memory pairs, a communication network with a throughput-related parameter g (time interval between consecutive messages), and a synchronisation mechanism between them with latency L. Valiant envisioned parallel algorithm execution to proceed in *supersteps*. Each superstep consists of three successive stages:

1. Parallel computation – where each processor performs computations only on its own *local* data
2. Communication – where processors exchange data with other processors

3. Barrier synchronisation – where each processor waits until all the other processors have finished

The parameter g is set so that for h messages of size one, it takes time hg on the processor to deliver it. Note that it does not capture any locality in the communication, where one set of processors may take a longer or shorter time to communicate than another. The BSP model allows analysis of the time-complexity of algorithms, but not their physical communication costs. The analysis consists of determining the cost of each superstep, as the sum of the largest computation time of the processors, the longest communication time of them, plus the synchronisation latency. The total time cost of the parallel algorithm is then given by the sum of the superstep costs needed to perform it.

Valiant also introduces the notion of *parallel slackness*, whereby algorithms designed for the PRAM model can be run at only a constant factor penalty compared to the PRAM model, but on the BSP model. This works provided there are sufficiently large numbers of parallel PRAM processors to be simulated for each BSP processor. Essentially, it consists of multithreading to hide the latencies of communication and barrier synchronisation, in order to achieve near-ideal throughput. However, as was discussed in Chapter 2, modern CMPs are severely I/O bandwidth constrained rather than simply I/O latency constrained, thus such a technique would not be a panacea for modern CMP.

3.2.3 LogP model

Similar to the BSP model, the LogP model [32], introduced in 1993, also tries to capture some elements of the communication overhead. The LogP model consists of four parameters: L – maximum latency of communication, o – processor overhead of communication during which it cannot do other computation, g – minimum interval between consecutive messages, and P – the number of processors. Unlike the BSP model it does not constrain computation, communication and synchronisation into separate phases. Furthermore, any synchronisation required is achieved through communication. However, as in the BSP model, communication costs are uniform between the processors, and so do not capture any locality.

3.2.4 D-BSP model

The Decomposable-BSP model of Torre et al. [105], introduced in 1996, extends the BSP to have decomposable clusters – whereby supersteps only communicate and synchronise within their own cluster. They allow for the synchronisation and communication operations to be faster with smaller clusters, thus capturing some locality information. Bilardi et al. [13] expanded on this, and defined a more restricted *recursive* version of D-BSP so that it forms a binary decomposition tree of possible clusters. At each level of cluster size, a separate parameter for the time-interval between consecutive messages g_h and barrier synchronisation latency L_h is defined.

This model, finally, encapsulates some information about communication locality. Although it is not necessarily tied to the physical constraints of, say, the two-dimensional CMP surface – parameters can be chosen so that it reflects these constraints, by having the latency L increase exponentially with higher levels of the hierarchy. Like previous models, this one is primarily concerned with asymptotic time, rather than more recent concerns of cost such as energy consumption. Also, as this model was constructed for the HPC domain where there is abundant attached local memory (per processor node), it does not model the very large time and energy costs of external I/O communication to reach external memory versus local memory or on-chip memory. Nor does it model the demands of multiple cores on a constrained external I/O bandwidth.

3.2.5 Multi-BSP model

More recently, in 2008, Valiant updated his BSP bridging model specifically for the CMP domain [107]. Valiant shows that with different parameters, his Multi-BSP model encompasses the PRAM, D-BSP, and other models. Like the D-BSP model, this model assumes a number of levels of hierarchy for a CMP, with corresponding clustering of cores, memory, bandwidth cost and latency penalty. At each level of hierarchy, the model assumes there is a cost g_h for bandwidth and a cost L_h for latency, similar to the D-BSP model. He then introduces a memory term for each level of the hierarchy, that restricts the size of the problem that can be tackled at each level, as communication is assumed to occur through slots in the shared memory. By allowing for a full hierarchy, this Multi-BSP model contains multiple hierarchical levels of communication delay from neighbouring nodes, to distant nodes on chip, and even to off-chip external memory. Although it manages to capture a good deal of locality by its hierarchy, it does not capture the direct physical nature of locality. For example, there are parallel algorithms that rely on a 2-D communication with its neighbours, such as Conway's Game of Life and systolic matrix multiplication. In the case of Valiant's model, there are necessarily physically neighbouring nodes on a 2-D CMP that can only communicate by traversing up and down through the entire hierarchy, rather than simply directly as physical neighbours. Nonetheless, this is a significant improvement in capturing information about locality, compared to previous approaches. Unfortunately, this model is also primarily concerned with analysing and optimising asymptotic time complexity, rather than, for example, the energy consumption of communication induced by the algorithm.

3.2.6 A note on Amdahl's law

Amdahl's law is often brought up when discussing limits to parallelism [56]. It should be made clear that the tradeoffs posed between faster 'serial' processing and wider parallel processing are actually based on *coarse-grain parallelism* such as tasks and threads, and only as the problem is constructed. Advocates of the need for faster 'serial' processors may be forgetting that largely, these processors are faster because they utilise Instruction

Level Parallelism (ILP), deeper pipelining, memory parallelism and some speculative parallelism – which are simply different fine-grained forms of parallelism. They may not be inherently faster at executing serial code, so much as they are better at extracting such fine-grained parallelism on the fly. In a sea of processors, the internal computation involved in extracting this parallelism can be offloaded to other simpler cores, and this fine-grained instruction level parallelism distributed amongst a neighbourhood of cores. Indeed Taylor shows how a Scalar Operand Network [103] can distribute operands, for ILP, in an array of processors such as MIT's RAW project.

The correct application of Amdahl's law would be to characterise the *strictly serial* component of an algorithm than cannot be sped up by pipelining, or with fine-grained parallelism, or speculation, and thus would have little advantage to be executed on a specialised, fast 'serial processor'. In short, arguments misusing Amdahl's law, demonstrate not so much the limits of parallelism itself, but of the coarse-grained task or thread-level parallelism that most software engineers currently refer to as 'parallelism'.

3.3 Communication complexity approaches

Aside from parallel computing and algorithms work done in the HPC domain, some theory has also been developed for analysing communication costs.

3.3.1 Yao's two-party and multi-party models

Yao [113] studied the minimum amount of communication needed between two parties, say Alice and Bob, in order to compute a function together, where both parties have mutually exclusive parts of the function input data. In this model, the parties are assumed to have infinite computing resources at their disposal, and that the cost of communication between Alice and Bob is fixed. The communication complexity of a protocol is given by the maximum number of bits transferred between Alice and Bob to solve the function under arbitrary inputs. The communication complexity of the problem is then the lowest computational complexity over all possible protocols. Although some application was made to evaluating boolean functions, and extended to partitions of boolean circuits (where communication is across a divide between parts of a single boolean circuit), it doesn't account for the cost of communication increasing by distance. Nor does it consider the physical embedding of communication problems. Furthermore, even for boolean circuit evaluation, it is in general an NP-complete problem to determine the communication complexity for arbitrary functions (as it contains boolean satisfiability as a sub-problem). Thus, characterising realistically sized circuits for communication complexity is largely impractical.

3.3.2 Thompson's area-time complexity

Thompson [104] developed VLSI models where wires take up area and take time according to their length. He examined the tradeoff between VLSI area and computation time and showed that there were lower bounds to Area-Time-squared complexity (AT^2). Unfortunately, the model is rather dated. It assumes that area taken up by wiring directly comes at the expense of logic area, because wiring and logic occupy the same plane. However VLSI no longer resembles this – modern VLSI have multiple metal layers allowing wires to reside on separate planes and not directly interfere with logic area. That is, there are separate communication and computation resources in VLSI - as there are on CMP. Also, Thomspon assumed that wire delays vary according to the logarithm of length, which was an appropriate model at the technology scales of the time. Unfortunately, with technology scaling we have moved closer to the underlying physics, with delay linearly proportional to distance. This is something that Chazelle and Monier addressed [24] by enforcing linear communication time, however their model still assumes that wiring is in the same plane as logic. Thus, although there is a large body of literature with Area-Time optimal algorithms proven, these do not actually help in characterising the *communication* costs or performance for modern VLSI, or in CMP.

3.3.3 Leiserson's layout analysis

Similarly to Thompson, Leiserson examined the area versus interconnect tradeoff, but utilised the theory of graph separators to analyse connectivity [67]. Graph separator theory bears some resemblance to topological Rentian bi-partitioning, described later, as it relates the minimum number of edges that need to be removed to partition a graph into two equally-sized regions, to the total number of nodes in that graph. For some simple types of graphs such as planar graphs, k-dimensional meshes, cube-connected-cycles and other restricted graphs, separator theorems are employed to asymptotically determine their area of layout. We should point out again, however, that these results assume a single resource of both logic and wiring, whereas modern VLSI have separate interconnect resources consisting of multiple planes of metal wiring. Thus they do not help in characterising the actual communication costs for modern VLSI, or in CMP. Moreover, they only apply to graphs that have known separator results.

3.3.4 Bilardi's analysis

Bilardi and Preparata considered the theoretical impact of fundamental physical limitations to parallel computation – in particular the speed of light and the finite size of devices [14, 15, 87]. Indeed, these constraints of bounded communication speed and of physical separation of computation and storage due to the finite size of devices, also lies at the heart of this thesis. They also note that under these constraints, highly complex topologies such as binary trees or cube-connected cycles, cannot do any better in asymptotic latency compared to a simple mesh structure. Their analysis mainly consists of the upper and lower bounds in simulating the PRAM model and higher-node versions of its own model. They are primarily concerned with data-locality, as the farther away memory is located, the longer it takes to access. They demonstrate super-linear improvements in performance with the number of nodes, due to increased memory access locality as well as parallelism. However, their memory access patterns may not capture the true nature of software locality. Its relevance is discussed more in Chapter 8.

3.3.5 Fox's analysis

Some work by Fox [43] tried to analytically characterise the overhead of communication compared to computation for high performance computing machines with a hypercube and mesh topology. He primarily examined stencil operations regarding the system of partial differential equations that govern neighbour forces in a 2-D crystal lattice, but also looked at matrix multiplication and the Fourier transform. In particular, for matrix multiply he determined by a decomposition analysis that it has an 'information dimension' of two. Its relevance is discussed more in Chapter 8.

3.4 Rentian scaling

In the 1950s whilst working at IBM, Rent discovered empirically that the number of terminals for packages followed a power-law relationship to the number of gates within the package, that was not explicitly designed for. This emergent scaling property was first described by Landman and Russo in 1971 [64], and was denoted as Rent's Rule. It takes the general form:

$$T = kG^p$$

where T is the number of terminals, G is the number of gates, p is the power-law exponent between 0 and 1, and k is a constant of proportionality that has later been interpreted as the average number of terminals per gate [29].

Although it started as a simple scaling relationship for the number of pins on a device package, Landman and Russo [64] remarkably showed that this behaviour was also present in the internal scaling of logic within integrated circuits, at a continuum of scales. That is, Rent's rule is a *self-similar* property of circuits. Moreover they found that the Rent's exponent is characteristic of the type of circuitry present – with an exponent of 0.5 for regular structures like static-RAM, to higher values of 0.75 for random logic.

Donath in 1979 showed that the self-similarity of Rent's rule within circuits, has direct implications for the average wirelength of the circuits [39], namely that for $p > \frac{1}{2}$, the average wirelength would grow as $\Theta\left(G^{p-\frac{1}{2}}\right)$ and the total length of wire needed for the entire circuit would grow as $\Theta\left(G^{p+\frac{1}{2}}\right)$. Later, Donath showed that Rent's rule actually predicts a power-law wire-length distribution for $p > \frac{1}{2}$ [40] of the form $\Pr(L) \propto L^{2p-3}$, and validated it on actual designs.

Since then, Rentian scaling properties, and their associated wire-length distributions have been experimentally validated on many real designs and with multiple partitioning techniques [98].

3.4.1 Regions I, II and III

There is a considerably lower cross-sectional density of connections between VLSI chips than there is within a VLSI chip. This is because pins are necessarily orders of magnitude larger than local on-chip interconnect. What this means is that near the chip-scale, the Rentian relationship between terminals and gates diverges considerably from the relationship at lower scales, as the region is no longer self-similar. For example, at the chip level, partitioning the design into two results in a common high-density count of terminals at the partition, but this is entirely dissimilar to the low-density count of pins around the rest of their boundaries. This expected drop-off in power-law scaling at the highest scales is called *Region II*, with the main power-law scaling behaviour of Rent's rule denoted as *Region I*. An example for a VLSI benchmark circuit can be seen in figure 3.1(modified after Stroobandt's paper [97]), where we can clearly see a drop-off in the otherwise power-law behaviour, at the top levels of the chip hierarchy.

Stroobandt also showed that in certain cases, a Region III phenomena can appear [97] that diverges from Region I at low scales. This is due to the properties of the building blocks used to create the circuit – as these gates have a small, fixed number of terminals, they effectively constrain the allowable interconnection complexity at these low scales. Such behaviour is dependent on the property of the library of gates allowed in a design.

3.4.2 Application

Rent's rule has been found to be abundantly useful in the analysis of large circuits in VLSI [98]. As well as its original use for predicting pin counts, it is used to predict wire length distributions [40], and even for estimating critical paths in complex circuits even for future technology nodes [20]. Rent's rule has been found to apply across vastly different scales, from only tens of gates to multi-million gate designs and beyond.

More recently, Rent's rule has been derived from first principles using basic assumptions [29]. This means that it is not just an empirical observation, but can arise from something more fundamental and can even be more generally applicable. Christie and Stroobandt [29] suggest that Rent's rule arises naturally as a result of the locality of connections – that is blocks are placed/mapped with consideration as to how they need to connect to each other so as to not create undue wiring.

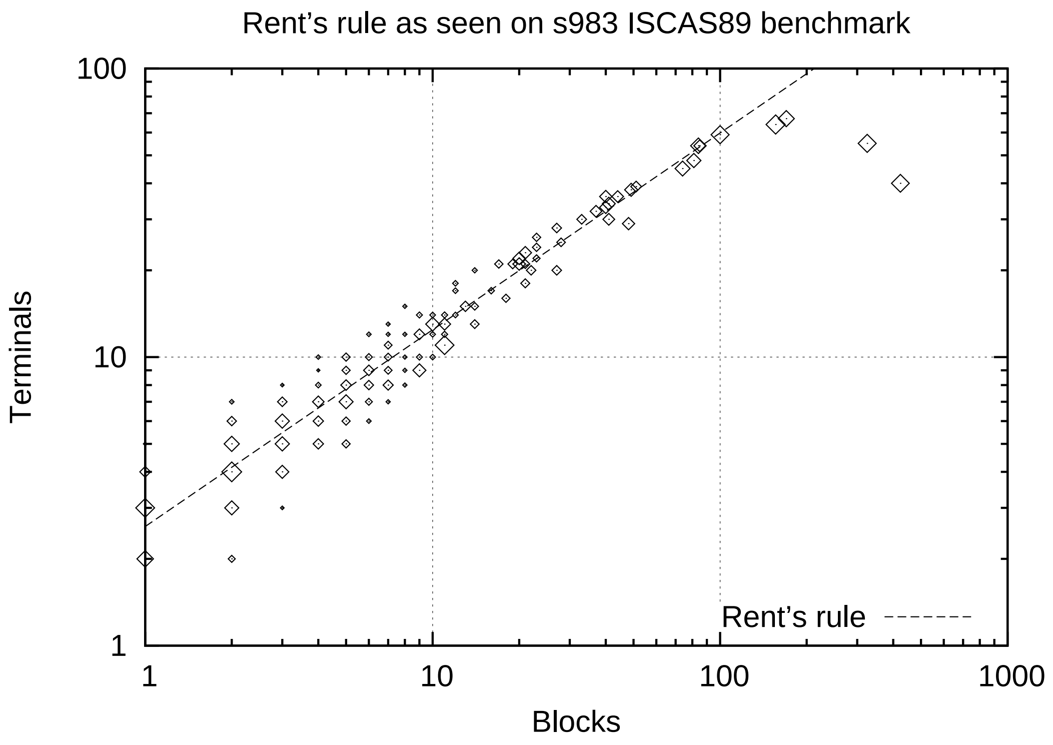


Figure 3.1: Example of Rentian scaling on a benchmark VLSI circuit. This figure is modified after Stroobandt's original paper [97]. Note the presence of a Region II near the chip-level scale. The average behaviour of Blocks (i.e. gates) to Terminals is a power-law in the main Region I.

3.4.3 Topological vs. Physical Rentian scaling

Logic circuits can be represented as graphs, with nodes representing gates, and edges representing wires. More generally, wires may actually connect to multiple destinations, and thus are represented as hyper-edges. Rent's rule can be determined and applied in two different ways – topologically or physically. In a physical analysis, a contiguous region of logic is taken and the number of gates/nodes counted within it. The number of nets that connect from within the region to outside the region are then counted as terminals. By taking multiple regions of varying sizes, the physical Rent's parameters can be estimated. In the VLSI community, because logic is mapped to physical positions by a placement algorithm, this is typically referred to as the Rent's parameters *by placement*. For our purposes, we choose to instead refer to this as the *physical* Rent's exponent, as the position of nodes is not necessarily due to such a placement algorithm, as we shall see later.

As the circuit can be represented by the graph/hyper-graph, we can also topologically determine a Rent's exponent by partitioning the graph into topological regions/clusters. Although a number of approaches are available [99], a simple one to describe is min-cut bi-partitioning. In figure 3.2 we see a simple graph on the left. This is partitioned into two roughly-equal sized sets of nodes such that there is a minimum number of edges from one partition to the other. In figure 3.2 this top-level partitioning is shown in green.

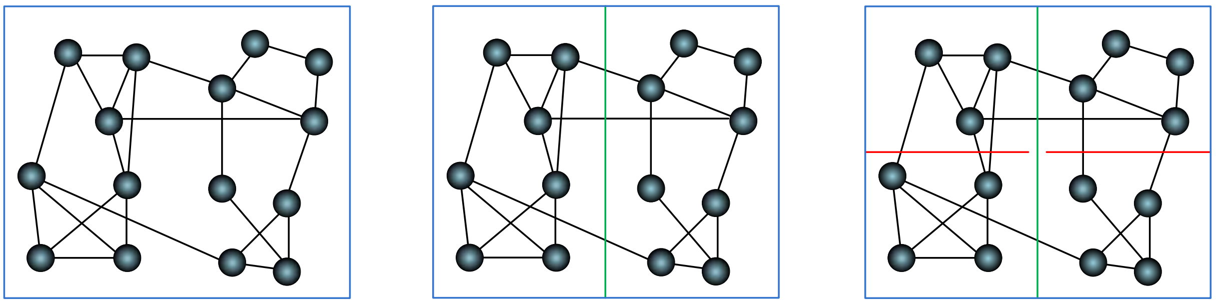


Figure 3.2: Two levels of min-cut bi-partitioning (green, then red) applied to a sample graph.

This can be applied recursively, and we see in figure 3.2, another level of partitioning is shown in red. Min-cut is known to be an NP-complete problem, so practically this is performed by heuristic approaches. To determine the Rentian scaling, at each level of the partitioning hierarchy we can count the number of nodes, and the number of edges that crosses a higher-level partition. For example, in figure 3.2, at the second level of partitioning, there are (counting clockwise) 4, 4, 4 and 3 gates, with 4, 3, 4 and 5 terminals respectively.

The topological and physical Rent's exponents can actually be different [109]. In VLSI, the physical Rent's exponent is determined not only by the connectivity properties of the circuit, but also by their physical placement onto the chip. A poor placement leads to a higher physical Rent's exponent than a good placement, and a random placement leads to a limiting-case physical Rent's exponent of one.

3.4.4 A-priori wire estimation

Following on from Donath's original derivation of wire-length distributions, a number of improved results have since been derived. Due to assumptions in Donath's average wire-length derivation regarding hierarchical partitioning, it turned out his predictions were off by about a factor of two [39, 40]. Donath's wire distribution derivation also assumed an infinitely large circuit, so that the power-law distribution continued to unbounded lengths. Real chips, of course, have a maximum wire-length constrained by the die size. Later derivations were able to take these factors into account and better predict the wire-length behaviour at scales near the chip's side-length [98, 29]. Whereas previous approaches utilised a hierarchical partitioning analysis, a stochastic approach introduced by Davis demonstrated better predictions by treating connectivity as a stochastic function of occupation probabilities [36]. Further extensions were later made into the optical domain [84], and into 3-D stacked die [60, 116] amongst other areas.

3.4.5 Fractal dimensionality

A number of people have noted that Rent's rule can be considered a measure of the fractal dimensionality of the communication graph [41, 83, 43, 84, 99]. Although their reasoning is arrived at largely independent, their arguments are essentially identical – the edges crossing the boundary of a region can be thought of as measuring the surface area of an object, whilst the number of nodes inside the region can be thought of as counting the volume of that object. The Rent's exponent can then be thought of as the scaling ratio between the surface-area to the volume. For simple d-dimensional meshes, one can easily see that this indeed produces a matching Rent's exponent of $p = \frac{d-1}{d}$. More simply this is just: $p = 1 - \frac{1}{d}$. Stroobandt gives further arguments as to what properties a measure of fractal dimensionality should be expected to possess, and how these properties eliminate other potential candidates, but that are all satisfied by this measure alone [99].

3.5 Summary

Although a number of approaches exist for understanding the communication properties of high-performance parallel computers and digital circuits, none of these models is appropriate for describing the expected locality and costs of communication for the CMP domain, and indeed many of them make assumptions that are invalid for current VLSI technologies. Perhaps the most interesting model of locality is Rent's rule, which rather than merely being an analytical model of asymptotic complexity, also produces quantitative predictions of the expected locality of wiring that have been validated on many real designs. However, the prior work in Rent's rule does not consider how it might be applicable to chip-multiprocessors in either the communication characteristics of software running on it, or for the networks-on-chip that connects the cores together.

CHAPTER FOUR

4

RENTIAN LOCALITY FOR NETWORKS-ON-CHIP

Various tiled architectures exist ranging from Chip Multi-Processors (CMPs) to arrays of programmable logic or even to heterogeneous arrays that include custom-IP blocks. If one were to replace the wiring between these blocks with packets over a NoC, then it is important to determine what the properties of this traffic might be. Indeed, this knowledge should aid us in the analysis and development of improved NoC routers.

In this chapter, the importance of traffic distribution models is shown in the analysis of NoCs. A case for a Rent's-based model is made and its implications explored. The reasons why Rentian-behaviour might be expected are discussed, and a bandwidth version of Rent's rule is derived. From this, a hop-length distribution, the NoC equivalent of a wire-length distribution, is constructed and explored for scaling and traffic implications. To better explore the differences caused by using alternative locality models, fault tolerance design is used as a case-study. Firstly, an example semi-random task graph is mapped using simulated annealing for optimisation. The result is used to generate comparable best-fit Rent's and exponential hop-length distributions. The expected distribution of traffic types under Rent's rule is then analysed, and the observations are used to design a new fault-tolerant router. The three hop-length distributions: Rent's, exponential and uniform random, are applied in the analysis of this fault-tolerant router. It is shown how the choice of traffic model makes a significant difference to the impact of faults on routability and in the quantity of congestion around faults. Rent's distributions are shown to result in considerably better fault-tolerance characteristics than uniform random traffic in these analyses.

4.1 Extending Rent's Rule for NoC

4.1.1 Argument

Future massively parallel CMP will execute applications with enormous amounts of parallelism. CMPs are very different from multi-chip multiprocessors due to the much higher-bandwidth and lower-latency communication channels available between cores on chip, enabling fine-grain parallelism [81].

As was discussed in Chapter 2, the currently used models of uniform random, transpose and other non-local traffic patterns have unacceptable scaling behaviour for CMP and are not sustainable. For scaling to continue, it is essential that locality be exploited, the question then is characterising how much is actually available, and in what form we might expect to find it.

Software in CMPs is mapped to physical locations, with data flowing along paths from one core to another. This combination of fine-grain parallelism and data-flows is analogous to circuits in VLSI. Indeed, the MIT RAW project uses the term *software circuits* to describe such mappings [101], explicitly noting the similarity to VLSI-like place-and-route. The difference with VLSI is that software circuits use processors instead of dedicated logic blocks, and use a NoC instead of inter-block wires. Indeed, one could transform data-flows in one domain to the other, interchanging dedicated logic blocks with processors, and wires with a NoC.

If software circuits are anything like hardware circuits, then it is interesting to explore whether they follow similar laws, such as Rent's rule. To re-iterate, for VLSI designs, Rent's rule relates the number of terminals in a boundary, to the number of blocks within that boundary by a power-law relation:

$$T = kG^p$$

Where T is the number of terminals, G is the number of blocks (gates), k is the average number of terminals for each block, and p is the Rent's exponent.

We shall refer to this as the classical-version of Rent's rule, as it applies to its traditional domain of logic design.

In the following sections, we wish to show that a similar relationship can be applicable to NoC bandwidth, and examine what that would mean for scaling.

4.1.2 Comparing VLSI with NoC

As shall be shown in section 4.1.3, for Rent's rule to emerge, a certain type of bandwidth locality is required to be present at multiple scales. The reasons why locality of tasks is desirable for NoC are directly analogous to those for classical logic placement as can be seen in Table 4.1. Just as it is typically undesirable to place two blocks on opposite ends of the chip and wire them up, it is also undesirable to map two communicating tasks to tiles at opposite ends. Indeed, for critical paths, it is desirable to place tasks as closely together as possible.

The emergence of Rent's rule, however, is dependent on the properties of the underlying graph being mapped, the mapping algorithm's objective, and the underlying physical topology. The primary requirement for Rent's rule is a certain type of locality at multiple scales. This requires that the graph being mapped contains sufficient locality. For example, complete graphs have no locality to exploit, and uniformly random graphs have very little locality to exploit. The mapping algorithm must also favour this type of locality, even if it is not an explicit objective, but rather an implicit one such as minimising critical paths and congestion. Finally, the underlying physical topology needs to be structured so that some form of multi-scale locality can exist. For example, a single central star network does not lend itself to such locality, whereas a multi-scale star network does.

4.1.3 Derivation

Christie-Stroobandt based

To arrive at a Rent's rule for bandwidth, we can follow a directly analogous derivation to that of Christie and Stroobandt's [29], but in place of Gates and Terminals we can use Blocks and Bandwidth. For completeness, this is described below:

Let us consider a boundary with N blocks and external bandwidth B. Suppose a small perturbation is made to the boundary such that ΔN additional blocks are included. Without additional information we can only estimate that the additional blocks are likely to require the same amount of communication per block as the original N blocks. Thus we would expect the bandwidth to increase by:

$$\Delta B = \left(\frac{B}{N}\right) \Delta N.$$

Now let us suppose that some of this added bandwidth is actually internal, between the original G blocks and the additional ΔN blocks. This internal bandwidth reduces the external bandwidth seen on the new boundary, which we characterise by a parameter p, where $0 \leq p \leq 1$:

Domain to minimise	Wires	NoC
Delay	Wire delay	NoC latency (& congestion)
Congestion	Wire-density	Cross-sectional bandwidth
Power	Wire buffering & length	Hop-length & router utilisation

Table 4.1: Factors in communication locality

$$\Delta B = p\left(\frac{B}{N}\right)\Delta N.$$

If ΔB and ΔN are small in comparison to B and N, we can approximate this to a differential equation:

$$\frac{dB}{B} = p\frac{dN}{N},$$

which yields the bandwidth version of Rent's rule:

$$B = bN^p,$$

where b is a constant of integration corresponding to the average bandwidth per tile.

We should note that the exponent here is different in value and meaning to the classical Rent's exponent, thus it is important to make a clear distinction between them. Let us refer to the classical Rent's exponent relating terminals to gates as the Rent's terminal-exponent, and for this new derivation as Rent's bandwidth-exponent.

Now, consider a 2-D topology with four regions of equal block count. If these are combined together, then let us define α to be the proportion of external bandwidth to the four combined individual bandwidths:

$$\begin{aligned}\alpha &= \frac{b\,(4N)^p}{4bN^p} \\ &= 4^{p-1}.\end{aligned}$$

So α is independent of N, and thus independent of scale. Thus another way of interpreting Rent's rule is that at each scale, we would expect the amount of bandwidth locality (characterised by α) to be the same.

Indeed, we can provide an alternate derivation of Rent's rule with only the assumption of multi-scale locality.

Multi-scale locality based

Let $B\,(N)$ be a function describing the average external bandwidth for boundaries containing N blocks. Suppose that for all N up to the size of the design, the following condition holds:

$$\frac{B\,(4N)}{B\,(N)} = 4\alpha.$$

We shall call this the *multi-scale locality constraint.*

Applying repeatedly with n levels of hierarchy we have:

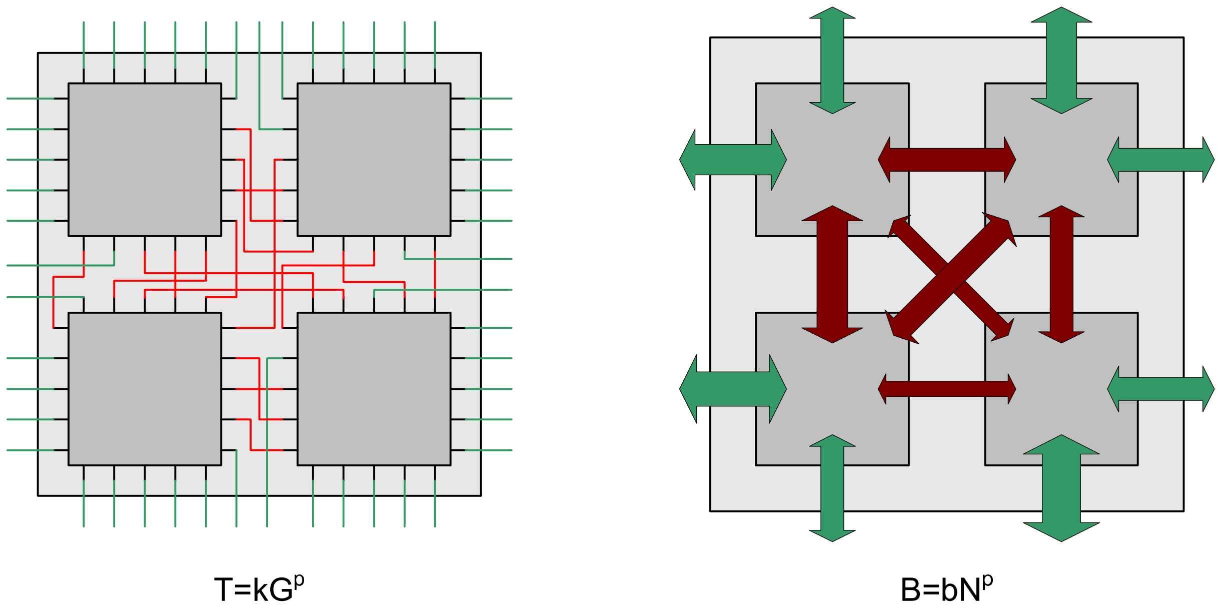


Figure 4.1: Rentian scaling in different domains. In VLSI the Rent's exponent characterises the locality of external wires (green) to total wires (red and green) at each level of the hierarchy. In NoC, the wires are replaced with packets, and the Rent's exponent characterises the locality of packets at each level of the hierarchy.

$$\frac{B\left(4^{n}N\right)}{B\left(N\right)} = \left(4\alpha\right)^{n},$$

In particular let us set N to one, and define $x = 4^n$. Since N and the particular boundaries are arbitrary, let us assume this relation holds continuously, extending n to $\mathbb{R}^{+}$:

$$\frac{B\left(x\right)}{B\left(1\right)} = \left(4^{p}\right)^{\log_4 x},$$

then,

$$B\left(x\right) = B\left(1\right)x^{p},$$

which we recognise as:

$$B = bN^{p}.$$

4.1.4 Comparison to classical Rent's Rule

There is an important distinction to be made between the classical and bandwidth versions of Rent's rule. In the classical version, the terminals and blocks in a design are fixed, regardless of how they are used, and so the Rent's terminal-exponent is also fixed. In the bandwidth version, especially over a CMP, the bandwidths are not fixed and the usage of the NoC can vary dramatically from one application to the next. Thus the Rent's bandwidth-exponent is actually dependent on the software/application currently running on top of it and may change accordingly. In both the classical and bandwidth versions, the Rent's exponent effectively characterises each level of the hierarchy. In figure 4.1 we see one level of the hierarchy with four sub-blocks. For VLSI there are terminals that connect within this level of the hierarchy (red), and those that connect outside (green). Analogously there are packets that flow inside (red arrows) or flow outside (green arrows). If this split between internal communication and external communication is self-similar, i.e. if we find approximately the same split across a spectrum of the hierarchy, then it follows the corresponding version of Rent's rule.

The bandwidth-version of Rent's Rule does not just apply to CMPs. For a VLSI design consisting of many blocks wired together, if one replaced these wires with a NoC, then the Rent's terminal-exponent of the former may match the bandwidth-exponent of the latter under certain conditions. This happens if the wire/terminal switching activity for the original version is highly uniform, and independent of scale, then the number of wires is directly proportional to the expected bandwidth, and the Rent's rule for wires automatically translates into the bandwidth version with identical exponent. This also requires that long wires should have the same expected switching activity as short wires, otherwise the rule may not follow, or a different exponent may result. Additionally, this assumes a seamless transition from wires to NoC and neglects the effects that NoC congestion, saturation and latency may have on bandwidth.

It should be noted that the classical version of Rent's rule can be used to estimate bandwidths in VLSI by assuming a particular wire activity. However, this is quite distinct from a bandwidth version of Rent's rule itself. The emergence of both versions of Rent's rule arises from the type of placement optimisation used. In the classical domain, logic is placed so as to reduce wiring, and the number of wires between blocks *may* correspond reasonably well with the inter-block bandwidth requirements. In the bandwidth domain, tasks are mapped so as to reduce communication. So they are optimising different attributes, resulting in different Rent's rule domains. For example, when using a NoC, the amount of communication between two blocks does not correspond to the amount of wiring between those two blocks. Indeed, there might not be any direct wiring between communicating blocks, and even when there is, the use of wires is shared and wire activity is application dependent. Thus one cannot merely use the classical Rent's rule to estimate such communication.

4.2 General Implications

4.2.1 Hop-length distributions

Just as the distribution of wire-lengths is of importance in the classical VLSI domain, so is the distribution of hop-lengths in NoC. The effect that hop-length distribution has on various analyses shall be demonstrated in later sections.

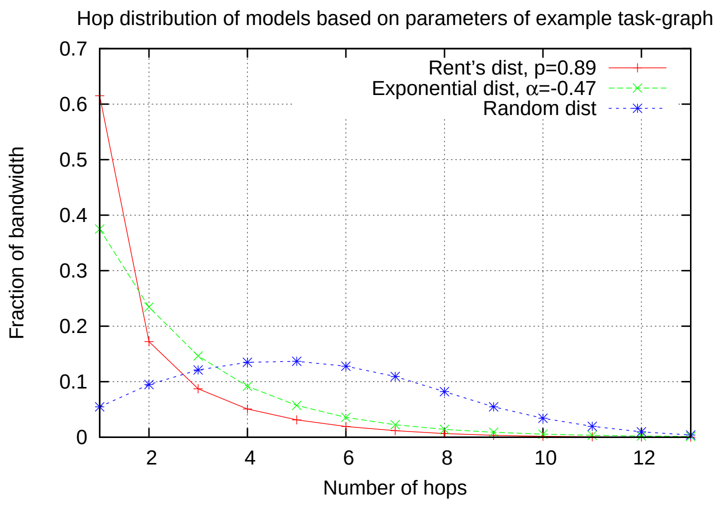


Figure 4.2: Three distributions fitted to a semi-random task-graph.

The site function, giving the degeneracy of a physical length s, is:

$$D(s,\,L) = \begin{cases} s\left(s\left(2L - s\right) + 1\right) & 1 \le s \le \frac{L}{2} \\ \frac{4}{3}L\left(1 - L^2\right) - \frac{5}{3}s\left(1 - s^2\right) + 6Ls\left(L - s\right) & \frac{L}{2} \le s \le L \\ \frac{1}{3}\left(2L - s + 1\right)\left(2L - s\right) * \left(2L - s - 1\right) & L \le s \le 2L \\ 0 & s > 2L \end{cases}$$

And the occupation probability is:

$$\begin{aligned} Q\left(s\right) &= \frac{1}{2s}[(1 + 2s(s-1))^p + (2s(s-1) + 4s)^p \\ &\quad -(2s(s-1))^p - (1 + 2s(s-1) + 4s)^p] \end{aligned}$$

The proportion of total internal bandwidth for hop-length s is then:

$$F(s,\,H) = \sum_{h=1}^{H} \frac{4^{H-h}\left(4^{p(h-1)+1} - 4^{ph}\right) Q\left(s\right) D\left(s,\,2^h\right)}{\left(4^H - 4^{pH}\right)\sum_{m=1}^{2h+1} Q\left(m\right) D\left(m,\,2^h\right)}$$

Where there are 4^H tiles arranged in a mesh with side-length $L = 2^H$, and p is the Rent's exponent.

Figure 4.3: Equations describing Christie and Stroobandt's equally optimised partitioning and placement model [29]

Three hop-length distribution models are examined here. The first model is based on Rent's rule using Christie and Stroobandt's equally optimised partitioning and placement model (described in figure 4.3). The second assumes a uniform random model. The third examines a semi-random generated task graph that was mapped by simulated annealing, found to be close to exponential in distribution.

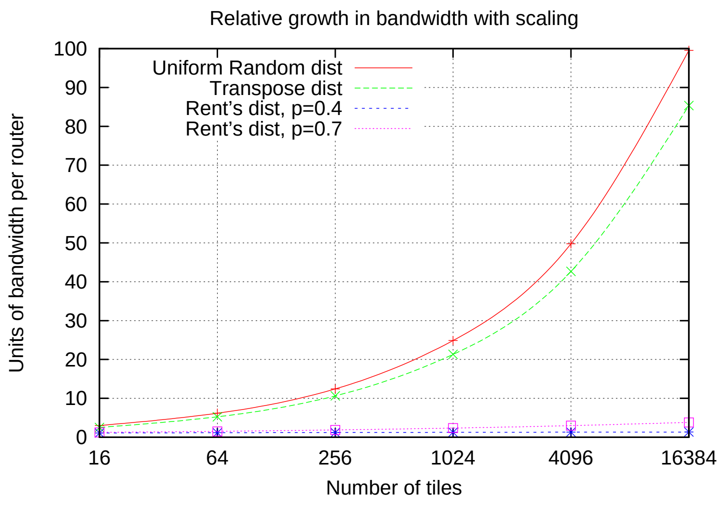


Figure 4.4: Scaling behaviour of bandwidth per router for different traffic distributions. The scale is such that, on average, one unit of bandwidth is generated per tile.

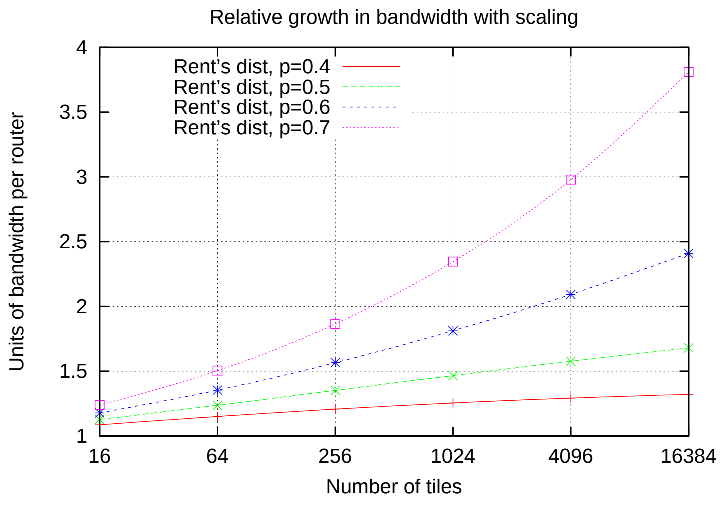


Figure 4.5: Relative bandwidth per router for different Rent's exponents as the number of tiles grows. The scale is such that, on average, one unit of bandwidth is generated per tile.

4.2.2 Implications for scaling

We can utilise the model to estimate the scaling behaviour of traffic under Rent's rule. Using Christie and Stroobandt's equally optimised partitioning and placement model, we can calculate the average internal bandwidth per router. This is merely the weighted sum of the hop-length distribution:

$$B_{av} = \frac{\sum_{s=1}^{2L} F(s,\, H)\, s}{\sum_{s=1}^{2L} F(s,\, H)}$$

Figure 4.5 shows how this scales with the number of tiles. There are a number of assumptions going on here. Firstly, it is assumed that the amount of traffic *generated* by each tile stays fixed with scale, and there is no scaling to account for increased clock frequencies. We note that a small change in Rent's bandwidth-exponent can increase the average traffic per router at large scales. For an exponent of 0.4, there is little change with scaling, however an exponent of 0.7 leads to a many-fold increase. Higher exponents lead to even greater increases with scaling, however, as seen in figure 4.4, this behaviour is insignificant compared to the scaling of random uniform and transpose traffic patterns.

4.2.3 Prior work

The equivalent of VLSI placement in the CMP-domain is task-mapping. Interestingly, some work has already been done by Hu et al. [57] regarding the impact of different task-mappings on NoC energy consumption. Their aim was to optimise energy consumption, and thus optimising locality wasn't explicitly their goal. However, because their NoC energy consumption model depended on the distance packets had to travel, optimising locality implicitly became a goal. They took a Multimedia System (MMS) consisting of an *.h263* video encoder/decoder as well as an *mp3* encoder/decoder, and then partitioned the benchmark into 40 tasks that were then mapped to a 5x5 array of tiles. Compared to random mappings (or rather the median result of 3000 random mappings), they observed a 60.4% reduction in energy consumption, on average, for sample video clips. As can be seen in figure 4.4, the Rentian model can directly be used to predict the expected reduction of aggregate bandwidth for optimal versus random mappings on a 5x5 tiled system. Hu's energy model has power consumption increasing linearly with packet distance, so the aggregate bandwidth becomes a reasonable proxy to compare total dynamic energy consumption. Using the Rentian model, for a very wide range of exponents of 0.4-0.9, we would expect reduction in aggregate bandwidth of between 60-71%, and with the more likely higher Rent's exponents of 0.7-0.9 we would expect 60-66% in savings. Thus this purely analytical derivation agrees well with Hu's experimental results. As the size of the system increases, there is a rapid growth in the predicted savings towards 100%. We should note that if this MMS design were implemented in VLSI, the inefficient wiring of random placement would not be tolerated for a commercial product. When replacing wires with packets, we would argue that this inefficiency also should not be tolerated.

4.2.4 Hot-spots

So called 'hot-spots' occur where a computation or communication resource has significantly higher utilisation than average. This can be an argument for redistributing the load of traffic and computation across the cores. In an era of increasing thermal constraints, 'hot-spots' in the resource-sense can literally turn into a 'hot-spot' problem in the thermal sense. However, one must be very careful to distinguish between the *variation* in power consumption from the *total* power consumption. By distributing tasks randomly, we have already shown that the total power consumption rises dramatically with scale. Thus applying this 'fix' may indeed cause power consumption to be more uniform but would also cause the entire system to be significantly hotter everywhere, defeating the entire purpose of seeking uniformity and load-balancing in the first place. Certainly, network congestion can affect performance, and therefore energy-performance, but this means that the physical mapping of tasks needs to be handled more carefully, rather than just randomly.

In discussing the concept of something being *distributed* vs. *local*, one must also be careful to distinguish between *physical locality* and *topological locality*. In the topological domain, hubs are high degree nodes, i.e. they are connected to many other nodes which are topologically one hop away from it. Thus hubs have high topological locality, but are problematic when it comes to actually physically placing them. For example, a hub may have dozens of topological neighbours, but might only accommodate four physical neighbours, so it must necessarily have longer average links to communicate with its topological neighbours. In general, high degree nodes are more ‘centralised’ but can lead to poor physical locality, whereas topologically ‘distributed’ structures allow better physical locality. The argument for distributing resources should then focus on the topology of the task-graph being mapped, rather than the mapping itself. Such a topology should be restructured to be more distributed and balanced in the topological sense.

Concerning Rent’s rule, we should note that analogous concerns happen in VLSI, with higher fan-outs and fan-ins for certain logic blocks or even gates. Whilst the occasional presence of special hubs may lead to local perturbations in the locality from Rentian statistics, they do not necessarily affect the average behaviour – indeed their rate of occurrence are even predicted by VLSI Rentian models [114].

4.3 Applied to Fault Tolerance in NoC

In this section we first take a look at several distribution models describing locality, and then apply them as a case-study to the design of a fault-tolerant router. Since the probability of encountering a fault grows with the distance a packet must traverse, it is shown how locality impacts internal router utilisation which can be exploited in router design. It is then shown how the model of locality can have a significant impact on the evaluation and comparison of competing router architectures.

4.3.1 Distributions Models

The Rent’s rule-based distribution

For the purposes of our NoC analysis, we shall restrict ourselves to a packet-based non-hierarchical mesh of homogeneous tiles. We use Christie and Stroobandt’s equally optimised partitioning and placement model [29] which is described in figure 4.3. This assumes a more optimal placement at the lowest scales than what may commonly be found for gate-level place-and-route of large designs. In such designs, the impact of moving a gate several gate-pitches away is small enough to be almost negligible, thus potentially resulting in less sensitive placement at the lowest scale. For the NoC, the impact of moving a task several tiles away can be quite significant in latency.

For this model of hop-distribution, we find that the first hop accounts for the bulk of NoC traffic (75-90% for Rent's exponents between 0.4 and 0.7). This means that communication between neighbouring tiles is the primary contributor to NoC congestion and power consumption when governed by Rent's rule. If true, given the large cross-sectional lengths and wiring density available for communication between neighbouring tiles, this suggests that a separate low-latency communication path should perhaps be created for this, without the latency and power overheads of packetisation and NoC routing. We should note that when there is less optimal placement at the very lowest scale, the first two hops account for the bulk of traffic instead.

Indeed, more recent work by Barrow-Williams et al. [9] has vindicated this as a strategy. After our initial publication [45] suggesting this approach, they arrived at similar conclusions regarding cache-coherence and employed a separate neighbour-communication path. Utilising this neighbour-communication path resulted in a significant reduction of network traffic.

While this analysis assumes homogeneous tiles, we can also estimate the behaviour of heterogeneous tiles. We note that a heterogeneous tile array may not be heterogeneous at all scales. For example, we can consider a 2x2 array of heterogeneous tiles that is then homogeneously tiled to a 16x16 area. In such cases, the placement of tasks is only restricted at the lowest levels, likely resulting in a significant deviation from Rent's rule at these low-scales. In these cases, Christie and Stroobandt's non-equally optimised partitioning and placement model can be used to assign a separate Rent's exponent for that particular scale of heterogeneity.

The uniform random distribution

When each source is equally likely to send traffic to each destination, we have a uniform random traffic distribution. This is a simple traffic model commonly used to evaluate NoC. The resultant hop-distribution corresponds to the site function in figure 4.3. Random mappings of task graphs also obey this hop-distribution. We see in figure 4.2 that low hop-count interconnects are poorly weighted, whereas medium-sized interconnects, of around half the side length, dominate.

The exponential distribution

The hop-distribution is a function of the mapping algorithm used and what it is trying to optimise. If the mapping algorithm's objective is insensitive to the bandwidth or latency of communication, then it may very well produce a 'random' mapping from a communication perspective. For real applications, however, whether in the VLSI, SoC or CMP domains, communication is already a pressing system-level issue, and with further VLSI scaling will only become more so. Whether a mapping algorithm is trying to optimise for performance, or for power, locality in communication will become an implicit or even explicit objective. However, depending on the application and mapping objective, a Rent's distribution does not necessarily result.

A semi-random task graph was generated by Task Graphs For Free [37]. The graph is not completely random in that it is split into multiple stages, with random links from each stage to the next or later stages. As these links between stages are random, there is only limited opportunity to extract some locality. It should be noted that although this tool is commonly used, it is questionable whether this approach generates realistic task-graphs.

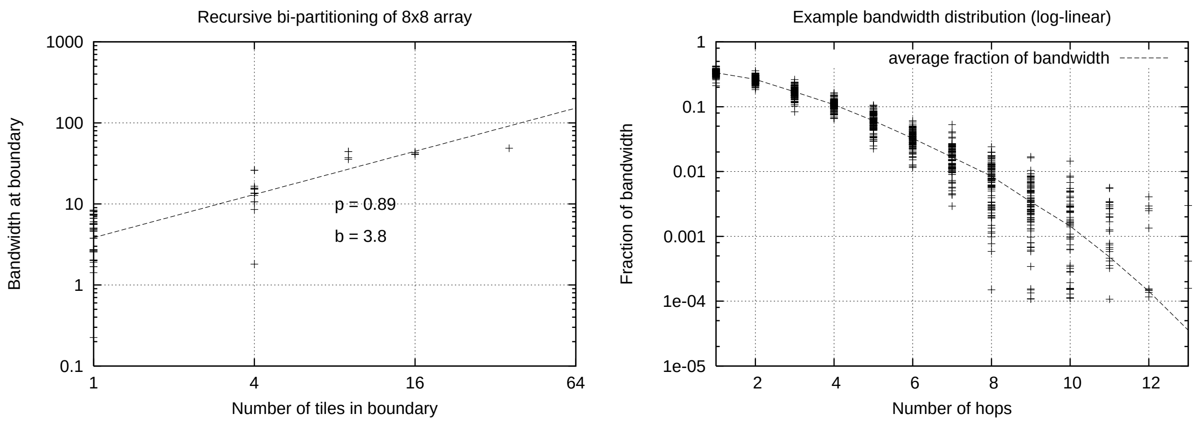
A graph with 250 tasks was generated and mapped to an 8x8 array of tiles. Each task was randomly allocated a processor load, and each task edge randomly allocated a bandwidth load. Simulated annealing was chosen for task graph mapping. The analysis by Orsila et al. [51] on simulated annealing parameter-selection was utilised. The objective function chosen for the task combines throughput and aggregate bandwidth, with the primary objective being to maximise throughput by minimising worst case load.

$$ObjFn = N_{Tasks} \times N_{Tiles} \times \max_{i \in tiles} \{load_i\} + AggBW$$

A minimum for the total $ObjFn$ is sought by the simulated annealing. Here N_{Tiles} is the number of tiles, and N_{Tasks} is the number of nodes in the task-graph. For each tile, the load ($load_i$) is calculated to be the worst case of either the router load in each routing direction, or the processor load. This is deemed to be more representative of throughput measurement since a tile's throughput can be processor-limited or router-limited. The worst-case load out of all tiles is then deemed to be representative of the inverse throughput of the entire mapped task-graph.

The aggregate bandwidth ($AggBW$) measures the sum bandwidth of all tiles, and minimising this is a secondary objective. A decrease in worst-case-load can easily come at the expense of an increase in aggregate bandwidth. However, it is still in the interests of throughput maximisation for the annealer to reduce aggregate bandwidth since total throughput can become bandwidth-limited due to congestion. This is similar to how a VLSI place-and-route tool optimises critical timing paths, even at the expense of increasing wire density.

The results from the annealing were then analysed to extract a Rent's exponent and a hop-distribution. A recursive partitioning was used to determine the Rent's exponent, and as can be seen by the fit in figure 4.6a, there is some evidence of Rent's-like behaviour. However the exponent is very high at 0.89, which usually suggests that it is close to a random graph. Random graphs generally exhibit poor locality and do not obey Rent's rule behaviour. Nevertheless, looking at the hop-length distributions in figure 4.6b we see evidence for some locality, though not of the Rent's variety. For a Rent's



(a) Recursive bi-partitioning of a mapping for the semi-random task-graph.

(b) Bandwidth Dist. for the semi-random task-graph taken over 100 simulated annealing mappings.

Figure 4.6: Rentian best-fit and resultant hop-distribution

distribution we may expect the hop-length distribution to approximate a power-law at low values, leading to a linear relationship on a Log-Log scale. In figure 4.6b we instead see an approximately linear relationship on a Log-Linear scale, thus indicating that it may actually be closer to an exponential distribution.

An exponential distribution has a slower initial fall-off in frequency compared to a Rent's distribution and so there is more of an emphasis for lengths of several hops, but a much smaller frequency for the single-hop, as can be seen in figure 4.2. This distribution's exponent is approximately -0.47.

Comparison of distributions

The extracted Rent's and exponential model's exponents now allow for some meaningful comparison between these distribution models, and with the random distribution.

In fact, in figure 4.2 we can already observe the effect of traffic models on a design decision. In both the exponential and random models, the first hop accounts for a significant portion of traffic, whereas this is not so for the random model. Thus having a separate neighbour-to-neighbour communication path is most valuable for the Rent's model, a little less valuable for the exponential model, but not valuable for the random model.

4.4 A fault-tolerant router design

Locality is of key importance to fault-tolerance analysis and design, as the greater the distance that packets need to travel, the more likely it is to encounter a fault. As we shall see the type of locality model can affect the fault-tolerance properties of different router designs. In this section we propose a fault-tolerant dimension-ordered router based on what we've learnt about the distribution of traffic types in Section 4.4.1. This design

Table 4.2: Suggested modes of operation for a fault-tolerant router

Operation Mode	Turn-off/Ignore	Fault-type Handled
Normal	none	fault-free
Through & Turn traffic only	Storage, switching and arbitration logic for Injection and Termination, and PE	faulty PE or Injection / Termination logic
Through traffic only	All storage, switching and arbitration router logic, and PE	faulty PE or bulk of router logic

serves as one of the test-cases, demonstrating how different traffic distribution models affect the analysis of designs.

Implications of traffic distributions

There are a number of important components within each tile. Existing tile-based architectures typically have a Processing Element (PE) of some sort (whether a processor, programmable-logic or custom IP block) that takes up the bulk of the tile area. Then there is the router logic which we can break down into various elements such as storage, injection, termination, arbitration, switching and signalling (including error-correction).

Packet-switched routers are typically dominated by storage, arbitration and switching needs. If the probability of faults is approximately proportional to the area taken up, then the most likely faulty component is the PE, followed by these other components.

Let us consider what happens if a tile has a faulty PE but a working router. The task that would have been allocated to the PE must be remapped elsewhere. Since no tasks are mapped to the faulty-PE tile, there are no *Originating* or *Terminating* packets for the tile and any logic that handles these types of packets is superfluous. Indeed, the fault-tolerant router should be designed to ignore these requests since such PE faults may be *Byzantine* in nature.

The dominance of *Through* traffic in faulty-PE tiles has interesting implications for the design of routers. If the only traffic that the router had to handle were *Through* traffic, then it doesn't need to redirect traffic, so a Switch isn't needed, nor an Arbiter for managing the Switch, and with simple flow-control, the Storage FIFO could be eliminated as well. Indeed it no longer looks like a router, but instead becomes a pipelined interconnect. Surprisingly, perhaps, such an interconnect is sufficient to handle the bulk of traffic that the router would need to handle. Of course, actually routing traffic is important too, but this requires all the switching, storage and arbitration logic in the router. Altogether this suggests that a layered approach should be taken to router fault-tolerant design which is summarised in Table 4.2.

The three modes of operation vary from supporting only basic *Through* traffic, to supporting both *Through* and *Turning* traffic, and to fault-free operation supporting all types of traffic. It should be noted that one may still need to route around faulty tiles

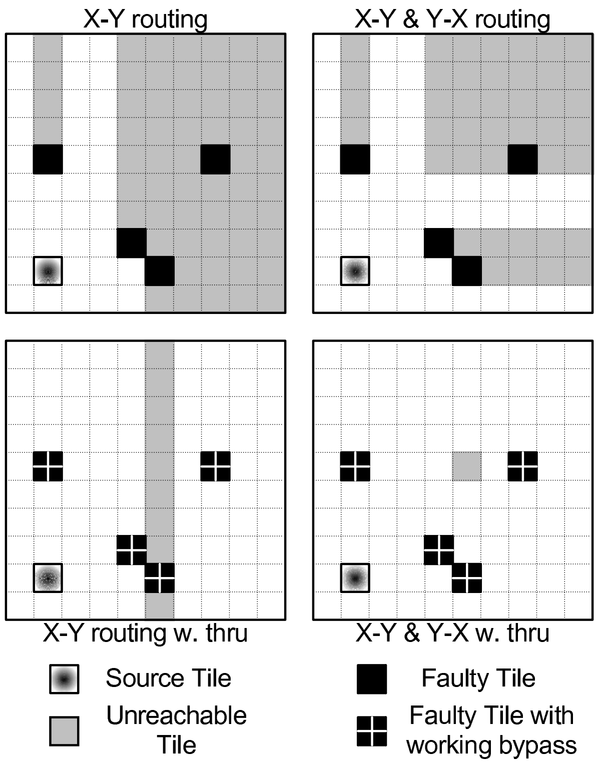


Figure 4.7: Plot of tiles unreachable from a source tile due to faults

if the links between tiles are also faulty. However, error-correction, whether applied at every tile, or every N hops, or end-to-end, may be used in conjunction with this strategy to further reduce the probability of failure. Previous work has also shown how a gracefully degrading NoC router [63] can maintain some functionality with some faults in the arbiters, switches and allocation, but it does not consider an even deeper fault-tolerance mode that lets traffic go through by effectively bypassing the router altogether.

Figure 4.7 illustrates how a through-mode can improve fault tolerance compared to regular dimension-ordered routing. Here four different routing strategies are shown and how they affect a node's reachability. Grey nodes indicate an unreachable node from the source. We can see that XY has worse reachability than an adaptable XY-YX routing algorithm, with the whole left side of the array unreachable. The redundancy in paths offered to the XY-YX routing algorithms allow the system-level mapping to statically choose a fault-free path. Note that we do not use any adaptive routing here, and so packets are still guaranteed to arrive in-order. Recall that we have assumed that the type of fault is statically covered, with the system-level providing for some adaptability to detect and cover the fault. The through-mode routers fare much better, especially compared to the regular XY routing algorithm. The one node that is unreachable by the XY-YX-thru router is because both possible turn points have faults there.

Design @90nm	Dimensions(μm)
Original router	497x495.6
With through-mode	510.2x509.6
area increase	5.6%

Table 4.3: Original & through-mode areas. These results were produced by collaborator Banerjee and not the author.

Area estimation

Let us now assume we add a through-only mode for the XY and XY-YX routers. In order to provide through-mode support, additional wrapper logic needs to be placed around the existing router. This results in additional area, and consequently more places for faults to occur. Thus an assessment of such a router needs estimates of the area penalty imposed by through-mode support.

To gain a first-order estimate of areas, a Virtual-Channel (VC) wormhole router [76] was modified to include the bypass muxing and logic, as well as the additional buffering needed for a single flit per direction. For flow-control, the input flit for each port is stored and immediately sent to the output of the adjacent port on the following cycle. A credit-based flow control is used based on the free FIFO entries available at the destination router's input port per virtual channel. The sent packet's VC is used to decrement the credit before forwarding this information back to the previous router. This implicit credit reservation ensures that the faulty router is always guaranteed to be able to send a flit, if it has one waiting. At this point the router acts as a pipelined interconnect, with some flow control.

In this way, only a single flit per port needs to be stored for this path, rather than one flit per VC per port. Also, full throughput can still be obtained for each VC without interruption as long as the FIFO size is larger than the round-trip latency of credit and flit propagation. This means that a router with FIFO depth of four can still theoretically source traffic through a single faulty router at full throughput. However, this round-trip latency means that the source router may be stalled for longer, and thus may reach saturation with less traffic than if without the faulty router(s). If two or more neighbouring routers are acting in through-mode, and the FIFOs are not large enough to hide the credit-latency, then this can also cause a latency impact for congested VC channels.

For simplicity, the configuration of router mode was assumed to be static and setup by a scan-chain of two configuration registers per router. The full control logic, however, was not implemented, as the goal was merely to obtain reasonable area estimates. The author asked Banerjee for assistance in estimating this area, and Banerjee was entirely responsible for producing the area estimation. The results after synthesis, and placement are summarised in Table 4.3. The author notes that a potential future optimisation in area should be possible by removing the dedicated through-mode flit storage and using part of the regular router FIFO storage instead.

4.4.1 Router-activity distributions

Let us examine a simple dimension-ordered router. We can identify four main types of traffic that result in different router activity:

- *Originating* – A tile may inject a new packet
- *Terminating* – A tile may be the final destination for the packet and thus remove it
- *Turning* – A tile may switch from X-routing to Y-routing or vice-versa
- *Through* – A tile may let traffic pass through without changing its direction

Let us look at the distributions of bandwidths from router to router. Here, *Turning* and *Through* packets take up bandwidth on both the incoming and outgoing edges of the router, whereas the *Originating* and *Terminating* packets take up bandwidth on only one edge. So we will assign a weight of a half on *Originating* and *Terminating* steps.

For a route of Manhattan hop length L, it is easy to show that the average number of *Through* hops is:

$$L - 2 + \frac{1}{L}$$

The fractional term accounts for the additional *Through* hops for the routes without any turns. Similarly, the average number of *Turn* hops is:

$$1 - \frac{1}{L}$$

The distributions for the expected router bandwidth with Rent's exponents ranging from 0.4 to 0.7 are illustrated in figure 4.8 for 64 to 16K tiles. The relative proportion of *Turning* traffic stays fairly constant for each value of p. This is fairly unsurprising since we have deliberately assumed a minimal number of turns. It is interesting to note that for small p there is very little *Through* or *Turning* traffic. This is because, as noted in Section 4.3.1, the traffic mainly consists of neighbour-to-neighbour communication, thus accounting for the dominance of *Originating* and *Terminating* packets. For large p, however, *Through* traffic rapidly dominates with exponential increases in tile-size. The proportion of *Turn* traffic, however, does not grow. This means that at large scales, a dimension-ordered router is predominantly letting traffic through from one side to the other. We should note that for the uniform random model, we would expect that *Through* traffic would dominate at all these scales, whereas in this model it only dominates at larger scales with higher Rent's exponents. Under this Rent's model, each router is relatively busy terminating and injecting packets, especially at lower Rent's exponents, instead of forwarding packets.

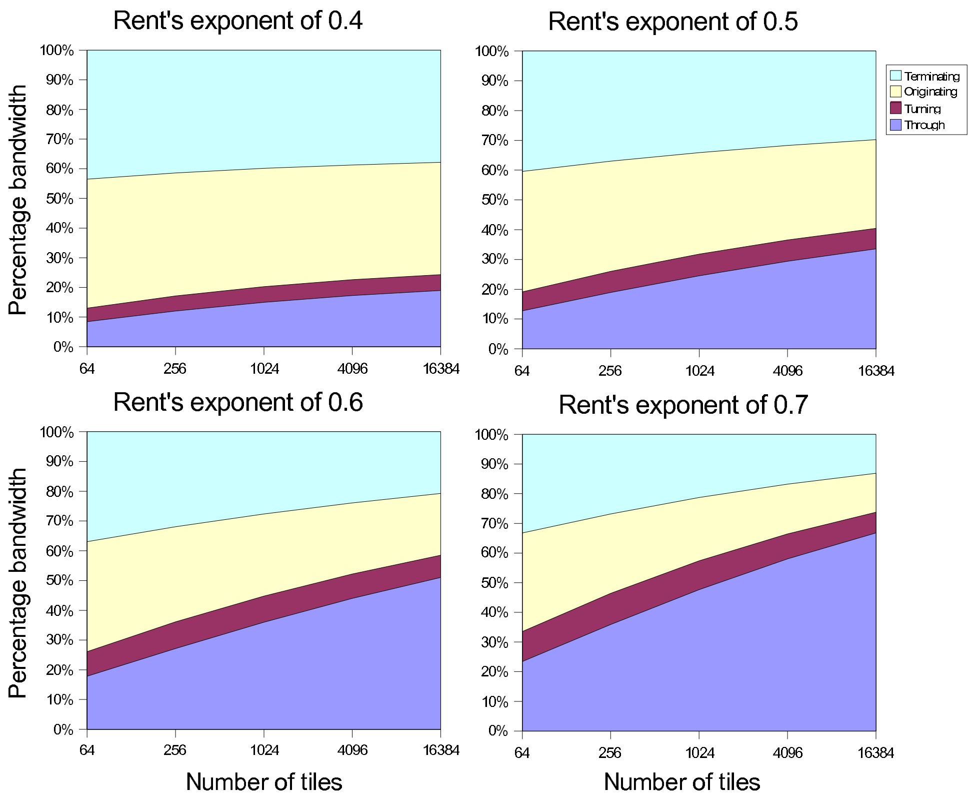


Figure 4.8: Expected distribution of router traffic for a working tile, for systems with Rent's exponent ranging from 0.4 to 0.7

4.4.2 Fault tolerance analyses

It is desirable to minimise the impact of faults on the system level performance, power and yield/cost. While there are a lot of different types of fault, whether static manufacturing faults, electromigration, signal integrity and timing faults due to parametric variation, for our purposes we shall assume that there is a predetermined static cover of these faults while the device is turned on. We are interested in comparing, to a first-order approximation, the impact of faults for some NoC fault-tolerance approaches.

We are not as interested in whether or not a task graph can feasibly be mapped to an array of faulty tiles and routers. This is because even with simple XY routing, there is a trivially findable subset of tiles that are capable of talking to every other tile in this subset. This subset may be relatively small, but it means that a task graph can still feasibly be mapped. Importantly, however, its performance may be severely crippled as a result, thus we are more interested in the system-level impact than mappability.

4.4.2.1 Unreachability analysis

When a task graph is mapped to an array of tiles, each task is assigned to a particular tile. The logical links between tasks then need to be routed from a physical source tile to a physical destination tile. Obviously if the logical link resides entirely in the same physical tile, then no such routing is needed. We shall ignore these and focus on the actually routed logical links, which we shall refer to as *task-links*. In the presence of router faults, some of these task-links may not have a reachable destination from the source. We shall refer to such links here as *unreachable task-links*.

One way of estimating the impact of faults at the system-level is to determine the fraction of task-links that are unreachable. Each unreachable task-link would require the remapping of either the source or destination task to another tile. If there are a large number of these, we would expect the incremental remapping process to be computationally expensive, and that the resultant mapping would incur penalties in the system's objective function, whether it be in power or performance.

As was shown earlier in figure 4.7, faults in the various routers can result in significantly many unreachable nodes. The through-mode routers can also result in unreachability, and of course when the through-mode path fails, then the fault behaviour resembles the non-through-mode counterpart. The through-mode wrapper adds additional logic which we must add to the probability of failure. Although it is possible for the bypass path to be faulty whilst the core logic is not, for now we will use a pessimistic bound and assume that if any of the wrapper is faulty, then the whole router is considered to be faulty. Given the small area overhead, this assumption should only have a negligible effect on the final results.

Let us now analyse the fraction of task-links that are likely to be unreachable for these dimension-ordered routers. We shall assume that all the task-links that have been mapped are between pairs of functioning tiles. Tasks that are mapped to non-functioning tiles or tiles with non-functioning routers will need to be remapped regardless of the behaviour of the router. As this is independent of the choice of router, this influence on task-links and remapping is not considered in this analysis.

Let f_{norm} be the fault rate for the normal router. We shall not estimate this value but instead use it as a parameter with which we can explore fault-rate sensitivity.

For an XY router, consider a path of m hops. Then the probability of the task-link being unreachable is:

$$f_{XY}(m) = 1 - (1 - f_{norm})^{m-1}.$$

For the XY-YX router, we need to separately consider turning and non-turning paths.

The probability that an m hop path has no turn is simply:

$$p_{noturn}(m) = \frac{1}{m}.$$

While turning paths have redundancy, non-turning paths do not and so need to be factored into the fault calculation:

$$f_{XY2}(m) = \frac{1}{m} f_{XY}(m) + \frac{m-1}{m} f_{XY}(m)^2 .$$

Let us now examine the Through-Mode routers.

Let θ be the fractional area overhead of the wrapper logic.

Then for a simple Poisson fault distribution [54], the probability of a through-mode failure is estimated by:

$$f_{thru} = 1 - (1 - f_{norm})^{\theta} .$$

And the probability of a turn failure is given by:

$$f_{turn} = 1 - (1 - f_{norm})^{\theta+1} .$$

Then the probability that a path of m hops fails for an XY-router with through-mode is:

$$\begin{aligned} f_{XYT}(m) &= \frac{1}{m}\left(1 - (1 - f_{thru})^{m-1}\right) + \\ &\quad \frac{m-1}{m}\left(1 - (1 - f_{turn})(1 - f_{thru})^{m-2}\right) . \end{aligned}$$

Finally, for an XY-YX router with through-mode, we have:

$$\begin{aligned} f_{XY2T}(m) &= \frac{1}{m}\left(1 - (1 - f_{thru})^{m-1}\right) + \\ &\quad \frac{m-1}{m}\left(1 - (1 - f_{turn})(1 - f_{thru})^{m-2}\right)^2 \end{aligned}$$

For small f_{norm}we can approximate these to:

$$\begin{aligned} f_{XY}(m) &\approx (m-1) f_{norm} \\ f_{XY}(m) &\approx \frac{m-1}{m} f_{norm} + \frac{m-1}{m}(m-1)^2 f_{norm}^2 \\ f_{XYT}(m) &\approx \frac{m-1}{m}\theta f_{norm} + \\ &\quad \frac{m-1}{m}(1 + (m-1)\theta) f_{norm} \\ f_{XY2T}(m) &\approx \frac{m-1}{m}\theta f_{norm} + \\ &\quad \frac{m-1}{m}(1 + (m-1)\theta)^2 f_{norm}^2 \end{aligned}$$

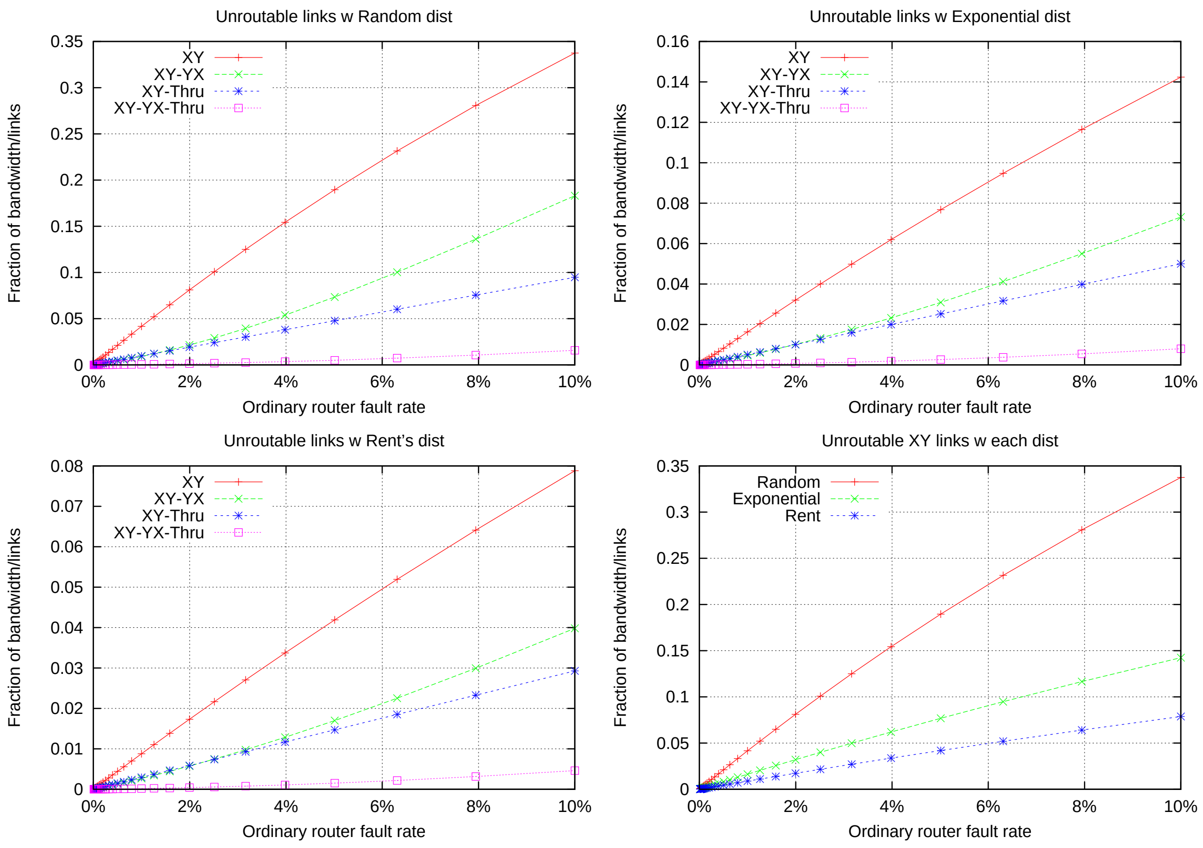


Figure 4.9: Plots of task-link unreachability for three hop-length distributions

Here we can see that the XY-Through router has faults dominated by the second term. In the XY-YX-Through router, parts of the this term are squared, leading to a very large reduction for low values of m. As hop-frequencies tend to zero as m grows, then it is the low-hop faults that are the most important here.

Now we are in a position to use hop-distributions to determine the expected fraction of task-links that are unreachable and that thus need task-remapping. We do so by merely taking the weighted sum over the hop-distribution.

Figure 4.9 shows the results for the three distributions: a Rent's rule-based, an exponential, and a uniform random distribution, as well as a comparison between the three on just the XY-router. As can be seen, the distribution makes a significant impact on the expected fraction of unreachable task-links. Taking the example of the XY-router, which consistently performs the worst, the uniform random distribution results in about about a 4x increase compared to the Rent's distribution. We can also see that relative comparisons can also be affected. For example, the Random distribution suggests that XY-with-through is approximately twice as good as the XY-YX routing for large fault-rates, whereas in the other distributions this relative gap is certainly not as large.

Looking at the more realistic Rent's and exponential distributions, we see that the absolute frequency of unreachable task-links induced by faults is considerably smaller, even

Router	Rent	Exp	Rand
XY	0.88%	1.6%	4.2%
XY2	0.27%	0.46%	0.91%
XYT	0.29%	0.50%	0.96%
XY2T	0.017%	0.029%	0.052%

Table 4.4: Unreachable task-links due to router faults, at a 1% fault rate for non-through-mode routers.

with large fault-rates in the regular router. Looking at Table 4.4, we see that at a 1% fault rate in the non-through router, the XY-YX-through-mode router has almost two orders of magnitude less impact than that of the plain XY router. However, it is interesting to note that even the XY router is expected to impact less than 1% of mapped links for the Rent's Distribution. This suggests that even a 1% fault rate in the XY routers may potentially be accommodated by system-level task remapping, although the same may not hold for the Random distribution at 4.2%.

Induced congestion analysis

We can also use the hop-distribution to calculate the impact of a single fault on the congestion of surrounding tiles. Let us compare this for our Through-Mode fault model to another approach which involves turning upon encountering faults [26, 79]. Lets refer to this strategy as Turn-Mode fault-tolerance. For paths along a single dimension, one turn at a fault is insufficient, and so a second turn is forced immediately after the first one, to route around the fault. Alternate models use fault-rings or convex fault-regions which direct traffic around it [25], which can potentially result in even more congestion, since all the traffic encountering faults, are forced to route around a ring or fixed path surrounding the faults.

For the Through-Mode model, it can be shown that the expected increase in traffic at a node is given by:

$$\begin{aligned}
R(x,y) &= \begin{cases} R_1 & x=0\wedge y=0, \\ R_2(|x|+|y|) & x\neq 0\wedge y\neq 0, \\ R_3(|x|+|y|) & otherwise \end{cases} \\
R_1 &= -\sum_{k=1}^{2L}\left(1-\frac{1}{k}\right)F(k) \\
R_2(d) &= \frac{F(d)}{4d}+\sum_{k=d+1}^{2L}\frac{F(k)}{2k} \\
R_3(d) &= -\sum_{k=d+1}^{2L}\left(\frac{k-d-1}{2k}\right)F(k)
\end{aligned}$$

Where a value of one corresponds to all the traffic that would have been handled by the faulty router (including terminating and originating traffic). For the dimension-ordered

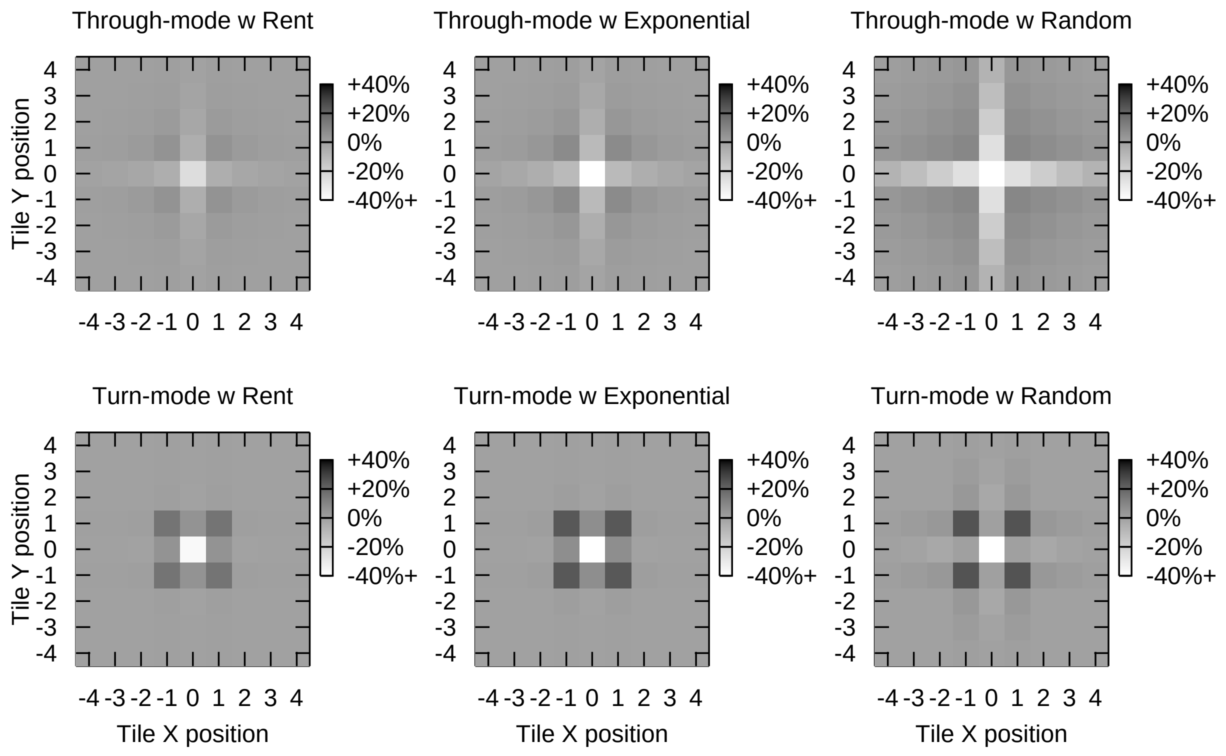


Figure 4.10: Collateral effects of a fault on router activity. The fault is located at the middle, and the shade indicates the change in activity of surrounding routers due to re-routing of paths that should have gone through the faulty router.

approach, a numerical solution was run by computing the difference between routes with the fault and without the fault, weighted by the hop-distribution. These two are compared in figure 4.10. Here, the shades correspond to the percentage change in total router activity compared to the average router activity. It does not differentiate between the different router directions. The lighter regions correspond to a reduction in traffic, and naturally the faulty through-mode router handles less traffic. However, this plot does not count the traffic that should have originated and terminated from the router, as this can no longer be handled by either and must be addressed by a system-level remapping. The intensity of the faulty turn-mode router corresponds to this traffic, and so is different between the three distributions.

We note that for the turn-mode routers, all three distributions result in more router activity immediately surrounding the fault, but the impact is quite localised. The through-mode router however has less marked router activity around the fault but results in slightly increased router activity over a larger area. It also results in reduced router activity along the axes of the faulty router. This is because when switching a task-link's routing from XY to YX, all the intermediate routers along the X axis also experience less traffic.

We can also see significant differences between the three distributions. The random distribution results in marked congestion around the Turn-mode router and under-utilisation in routers along the axis of the Through-mode router. The exponential distribution appears to be somewhere in the middle between the random and Rent's type distributions.

Both the spatial distribution and quantity of congestion impact the system's ability to compensate by remapping. The impact of the faulty Through-mode router on the surrounding tiles extends considerably far in the Random model (9% even at the corners of the array), and this may make such a router less desirable than the localised congestion of the Turn-mode router. In the Rent's distributions, however, the influence is far less (at most 4.5% around the fault, dropping rapidly to just 0.05% at the corners of the array), making it more congestion-friendly than the Turn-mode router (which is 15% around the fault). This means that under the Rent's distribution, the Through-mode router is clearly superior, whereas this is not so under the Random distribution. Thus the choice of distribution makes a significant impact on the selection of router designs.

4.5 Conclusions

In this chapter, in noting the similarities between task mapping on CMP with place-and-route on VLSI, a Rent's Rule connectivity model was explored. In addition, two derivations were given for a bandwidth version of Rent's Rule that can be applied to NoC. The conditions under which Rent's Rule is applicable, particularly for the System-level mapping of tasks in CMP architectures, was elaborated on. It was also noted that the Rent's bandwidth-exponent can vary from one application to next, especially for CMP architectures. Assuming the law to hold, one can estimate the future requirements of NoC with scaling, and this was shown to be sensitive to the Rent's bandwidth-exponent of the application. However, it was shown that unlike the unacceptable scaling trends of uniform random and transpose traffic models, a Rentian scaling of NoC bandwidth fares considerably better, in line with normal VLSI trends.

It was then shown how Rent's rule results in a distribution of hop-lengths. Two other hop-lengths were also examined, one representative of uniformly random traffic, and an exponential distribution that appears more representative for a semi-random graph mapped using simulated annealing.

Under Rent's rule, the distribution of traffic types that a dimension-ordered router has to handle, was analysed. This showed that the dominant router activity was in routing packets from one side and forwarding it to the opposite side. A through-mode that bypasses the router logic was thus explored as a fault-tolerance mechanism. For a modest area increase, it was qualitatively shown that this would allow improved reachability, however a quantitative analysis was desirable to assess its system-level impact.

Two analyses were introduced that made use of the hop-distribution, one that calculates the number of unreachable task-links caused by faults, and another that calculates the expected congestion impact due to a fault. The different hop-length distributions were shown to produce markedly different results in these analyses, with the Rent's distribution performing much better than the uniform random distribution. When comparing the congestion impact of faults in the through-mode router versus a turn-based fault-tolerant router, it was shown that the choice of distribution made a significant impact. Under the Rent's distribution, the through-mode router was clearly better, but this was not so under the random distribution.

Chapter Five

Experimental evidence for Rent's rule

This chapter examines evidence for Rent's rule already applying in software. Although arguments were made in Chapter 2 as to why one might expect such locality, in this chapter multiple techniques are used to detect the presence of Rentian locality, and establish how significant it is. This is actually quite a difficult task as the size of even simulated many-core systems are quite small, and the Rentian behaviour is more likely to be pronounced at larger scales.

In an ideal testing case one would already have a thousand cores in a message-passing system, running software optimised for locality with which to evaluate Rentian predictions. It should be noted that for VLSI, Rent's rule was first observed empirically by Rent at IBM in the 1960s, but it wasn't until 1971 that Landman and Russo [64] showed that this held in a self-similar manner for the internal partitions of a 13,000 node logic circuit. Then in 1981 Donath showed how this could predict length distributions [40]. By this stage digital circuits already had an appreciable size with which to fully test Rentian predictions.

Despite the lack of an ideal platform, this chapter tackles the quantification with two main approaches. Firstly it looks at simulated shared-memory many-core CMP systems with a variety of benchmarks. This is an approach first taken by Heirman et al. [52] who claimed to see such topological Rentian behaviour. This chapter first examine Heirman's results, and then expands on them by performing more rigorous tests for *physical* Rentian behaviour (versus Heirman's *topological* Rentian tests) in a 32-core simulated CMP system. Secondly, this chapter examines the more fundamental inter-connectivity of instructions in single-threaded programs to determine whether they exhibit the fractal connectivity that would be a basis for Rentian scaling. It employs a graph version of *box-counting* [95], a technique from the Physics/Network Science community, which can allow one to measure the fractal dimensionality of the data-dependency graph. It then directly examines the topological and physical Rentian scaling of such graphs – by basically treating the executed instructions much like logic gates in a circuit, and seeing if such a circuit would obey Rent's rule.

5.1 Other work: Heirman et al.

After our original paper [45] arguing for the existence of a bandwidth version of Rent's rule in CMP, illuminating work by Heirman et al. [52] examined this hypothesis on a cycle-accurate Virtutech Simics [73] simulator. Their setup involved 16, 32 and 64 core shared-memory CMPs using the SPLASH-2 [1] suite of parallel benchmarks.

We had originally expected that Rentian behaviour would become clearer only with a larger numbers of cores, and mainly on a message-passing architecture, where applications were written with some effort to minimise communication costs. Nonetheless, rather surprisingly, Heirman concluded that even with the implicit communication via shared memory on CMPs with as few as 16 nodes, there was evidence of Rentian scaling in bandwidth. What is remarkable is that this used actual NoC traffic, running under a distributed directory-based cache-coherency architecture, including non-coherence misses and other (conflict, capacity and cold) misses. In private correspondence with the author, it was revealed that locality of mapping from address space to core was maintained by allocating memory on a first-touch basis in blocks of one VM page of 8KB. In this way, the node that first writes to an unallocated page hosts this page, by mapping the virtual page to a free physical page with an address that puts it on this node. This allows some locality to be preserved even in a distributed directory-based cache coherence scheme where address-ranges are simply interleaved amongst the physical cores. However, it is not clear how practical such a scheme is on a general-purpose commercial CMP, especially if there are load-balancing concerns on a highly-contested memory page. We speculate that a more intelligent memory manager could allocate pages based on both physical locality and load-balancing concerns.

Unfortunately, as shall be shown in section 5.2, some of Heirman's conclusions of Rentian scaling for these benchmarks may have been a little premature. The appearance of power-law scaling was not rigorously tested against other potential hypotheses. Indeed, Heirman himself remarked [52] that the optimality of a particular partitioning would change over the duration of the algorithm, with some locality seen at one stage being non-local at another stage, particularly with the FFT (Fast Fourier Transform) benchmark. Given that the FFT is known to be an algorithm with poor locality [43], it was surprising that they extracted relatively low Rent's exponents for the FFT benchmarks (ranging from 0.59 to 0.74), whereas one might theoretically expect an exponent closer to unity (see Chapter 8).

Heirman et al. also showed that Rentian exponents could vary at different phases of the benchmark's computation. This is important to elaborate on, as we certainly do not claim that the Rent's parameters (and consequently the amount of communication or locality) would be uniform throughout any benchmark. They also showed that the optimal partitioning for one phase of the benchmark may not be as optimal for another phase of the benchmark. They created a metric that they called 'suitability' that measured the optimality of imposing a single partitioning that is optimal at one time, to all others times. Some benchmarks, such as *barnes*, exhibited a high 'suitability' throughout its execution, whereas others such as *fft4M* (a 4-million point FFT) and *cholesky*, had poorer 'suitabilities'.

5.2 Evidence in simulated CMP experiments

In our own investigations into CMP, we wanted to improve the robustness of our methodology compared to previous work, by comparing the Rentian model to other candidate null-hypotheses. Also, whereas Heirman concentrated at the NoC level on the topological Rentian scaling, we were primarily interested in the *physical*[1] Rentian scaling of data flows at the software-level and the corresponding predictions of hop-length distributions.

For this we utilised some existing traces from Barrow-Williams et al. [8]. They employed the Virtutech's Simics simulator platform to construct cycle-accurate traces of SPLASH-2 [1] and Parsec [6] benchmarks for a simulated 32-core x86 CMP. The simulated cores were each in-order with single issue pipelines much like Intel's Larrabee CMP project [93] and each had a private L1 cache and a large shared L2 cache. The traces for each core logged all reads and writes to memory addresses at a particular cycle time. In this way, one could directly determine the amount of implicit communication between cores via the shared memory system. For the SPLASH-2 benchmark, the threads were mapped very simply – with one thread allocated per core. Thus the flows can also be thought of as inter-thread data-flows as well as inter-core flows. This is slightly different from Heirman's work which examined all NoC flows, including cache directory/coherence traffic, as this only examines the fundamental algorithm data-flows, and is at a word-level granularity rather than cache-line level. These traces covered the entire benchmark's execution, including any initialisation phases.

We explored the presence of physical Rentian scaling in these traces by employing simulated annealing to embed the cores so as to minimise aggregate communication costs. Incidentally, this also allows one to choose multiple NoC topologies to explore the effect of topology on communication costs, which we also examined, but we do not report these results here.

For our 2D mesh, the cores were embedded in an 6x6 arrangement with empty cores at the four corners - the area of which, on a real CMP, could be utilised for other functionality.

[1]Please see Chapter 3 for further discussion on the differences between topological and physical Rentian scaling.

5.2.1 Caution: the uniform-traffic model

One needs to be particularly careful of the uniform distribution at small scales such as the 32 nodes here, because on a log-log plot, it can appear to scale as a power-law distribution with Rent's exponent $p < 1$. This is because for N_{tot} total cores, N cores being considered, k constant and B external-bandwidth for the N nodes, it scales according to:

$$B = k.N\left(N_{tot} - N\right),$$

so in the log-log domain:

$$\log B \quad = \quad \log k + \log N + \log\left(N_{tot} - N\right),$$

yielding:

$$\frac{\partial \log B}{\partial \log N} = 1 - \frac{N}{N_{tot} - N}.$$

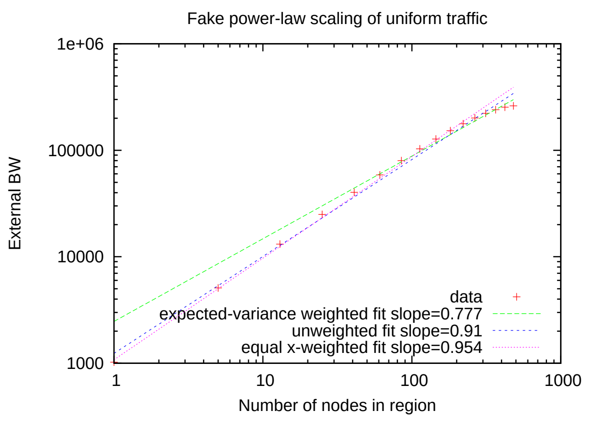
This is quite linear in behaviour for $N \ll N_{tot}$. Furthermore, an extremal fit made with two points at $N = N_{tot}/2$ and $N = 1$ would yield a slope of:

$$\frac{\Delta \log B}{\Delta \log N} = 1 - \frac{\log 2}{\log N_{tot} - \log 2}.$$

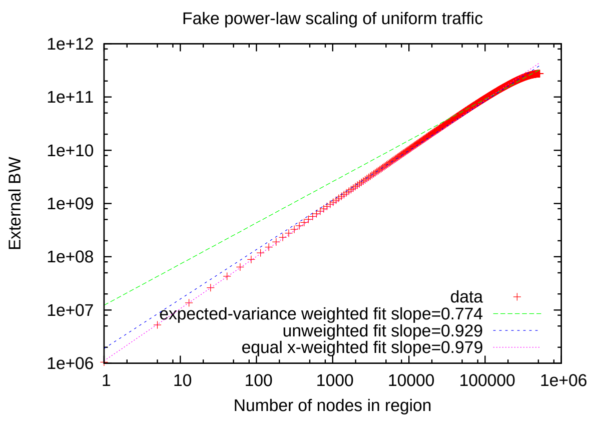
For $N_{tot} = 32$, this yields a minimum power-law slope of 0.75. By including more interim points, the best-fit slope starts to increase, but there are only a small number of interim points for $N_{tot} = 32$. For 1024 points, this minimum slope is 0.89. The picture is clearer in figure 5.1 with $N_{tot} = 1024$. A network of this size has a best-fit unweighted slope of 0.91, agreeing with the minimum prediction, however weighing the data-points to adjust for the smaller variance of larger regions, leads to a reduced slope of 0.774. One notes that along the x-axis there are far more data-points at higher values than at lower values, so another approach is to equally weigh each portion of the log-scaled x-axis so that each portion has a proportional influence on the fit, leading to an equal x-weighted slope of 0.954. For $N_{tot} = 32$, the *canneal* benchmark which is uniform, has a best-fit weighted slope of just 0.796 (see figure 5.2).

The reason why the uniform distribution poses a problem is that it actually does behave as an approximate power-law distribution, but with exponent $p \sim 1$. Indeed, a uniform distribution is actually a limiting case of the spatio-temporal Rentian model[2], with $p \rightarrow 1$. The biggest danger lies in the sensitivity of p, because a model with $p = 0.8$ has very different locality properties to one with $p = 1$. This makes the uniform model a special case that should be explicitly tested against. To guard against this with so few nodes, it means we need to include so-called Region II effects (see Background Chapter 3) by fitting against a proper Rentian model, rather than just a simple power-law.

[2]Please see Chapter 7 for more details of this model



(a) $N_{tot} = 1024$ nodes



(b) $N_{tot} = 2^{20}$ nodes (1Mi nodes)

Figure 5.1: Example of 'fake' power-law appearance in scaling with uniform traffic for 1024 and 1Mi nodes

5.2.2 Methodology

First, we employed a hand-crafted simulated annealing algorithm to find a good mapping of cores. All it can do is shuffle the mapping of the 32 cores in space. If a core needs to communicate with another core h hops away with bandwidth B it contributes hB to the aggregate bandwidth (and to NoC dynamic energy consumption as per Banerjee et al. [7]). The simulated annealing tries to minimise the aggregate bandwidth thus optimising for locality. This objective function is as good as any other to define 'optimal locality' (see the Chapter 6 for a discussion of *cost-universality*).

Each node was taken as a seed node, and then all the nodes surrounding the seed node up to a distance L were used to form a boxed region. The bandwidth of communication from inside the box to outside the box was plotted against the number of nodes within the box. This allows one to determine the physical rather than purely topological Rentian exponent. We fit the results in the log-log domain using a range of possible functions, but with two free parameters each, and recorded their RMS error. Both the Rentian and uniform models have two free parameters. For the Rentian model this is the average bandwidth and the Rent's exponent, so for the uniform model we proposed two generalised versions. One is the uniform distribution with total number of nodes allowed to vary as a free parameter, and the other extends the uniform model retaining information about the total number of nodes, but adding a free parameter to make it a full second-order polynomial. The first model, importantly, has an actual hop-length distribution associated with it, which we make use of for comparing predictions.

The non-linear least-squares fits were done using a Levenberg-Marquardt algorithm adapted from Price et al. [88]. The candidate fitting models were:

1. A simple power-law of form: $\lambda x^{-\mu}$, which is what is typically used for fitting Rent's exponents

2. A uniform distribution of form: $\lambda x(\mu - x)$. This essentially models a region of b nodes communicating with each other, where the bandwidth between each pair of nodes is uniform.
3. A modified uniform distribution of form: $\lambda x(n - x) + \mu$, where n is already fixed at the number of nodes in the system.
4. An exponential of form: $\lambda e^{\mu x}$
5. A logarithmic of form: $\lambda \log(x) + \mu$
6. Polynomials of form: $(\lambda x + \mu)^k$ for $k = 1..3$
7. A Spatio-Temporal Rentian distribution with average bandwidth a, and Rent's exponent b, with a topology of n nodes given. (This model is described in Chapter 7 and Appendix E)

As another Rentian model to compare to, we would have liked to have used Christie and Stroobandt's equally optimised and partitioned Rentian model [29] as well, but could not determine how to adapt it to a system with nodes that are not a power of four.

For each pair of communicating cores, we assumed that the variation of bandwidth was normally distributed.

For average bandwidth b of communication between each pair of communicating nodes, and standard deviation σ, we can define a relative variation of:

$$\epsilon = \sigma / b,$$

and for T independent links of communication going from within the box to outside the box, we would expect a total variance of:

$$\sigma_N^2 = T\sigma^2,$$

with a relative variation of:

$$\epsilon_N = \epsilon / \sqrt{T}.$$

In the logarithmic domain, this translates to a range of values of:

$$\begin{aligned} \log b_N &= \log(Tb(1 \pm \epsilon_N)) \\ &= \log T + \log b + \log(1 \pm \epsilon_N). \end{aligned}$$

For small ϵ_N, we have:

$$\begin{aligned} |\log(1 \pm \epsilon_N)| &\approx \epsilon_N \\ &= \epsilon/\sqrt{T} \\ &\approx \epsilon\sqrt{\frac{b}{b_N}}. \end{aligned}$$

Thus we assume that the standard-deviation in log-space of a box with N nodes scales according to $1/\sqrt{b_N}$. We use this to scale the sum-of-squares non-linear regression by weighting according to relative expected variance.

5.2.3 Results

In the VLSI domain, the appropriateness of a Rentian model is examined by comparing how well it predicts the wire-length distribution and average wire-length, based on fits of the Rentian parameters on examples and benchmark circuits. Our equivalent here in the CMP domain is to examine how well it predicts the hop-length distribution and aggregate bandwidth (the total distance-weighted bandwidth) for the 2-D mesh NoC topology.

Figure 5.2 shows a small selection of benchmark results, deliberately chosen to showcase a variety of behaviour – for a comprehensive set please consult Appendix A. The left-hand-side consist of physical Rentian fits and the right-hand side consist of the resultant predicted and actual hop-length distributions.

Just focusing on the Rentian fits, the RMS error for each fit function can be seen in figure 5.3. We note that because there is variation in the bandwidth for any given number of nodes (cores), the minimal achievable RMS error is constrained by the variation of the data itself. Thus we can remove the variance in data to obtain another comparison of RMS fit seen in figure 5.4. Both variances are important as a high data-variance can also indicate that the model is generally a poor fit, whereas we would also like to compare the relative scale of RMS error between the candidate fit functions without including data-variance.

From these we see that many of the alternative candidate functions are poor fits. The log-law and the power-law distributions did well for many benchmarks, however the uniform and Spatio-Temporal Rentian distributions were consistently the best fitting. Remarkably, this makes the uniform distribution the best null-hypothesis out of all the candidate fit functions, including the regular power-law fit. For the generalised uniform and Spatio-Temporal Rentian distributions, these also have associated hop-length distributions, thus allowing another form of cross-evaluation.

Given the Rentian parameter fits, the hop-length predictions were compared to the actual hop-length distribution, and returning back to figure 5.2 we can see these in the right-hand-side plots. Here, while *freqmine* subjectively has an almost equally poor Rentian prediction compared to the uniform model, *water_nsq* is very close, and *canneal*

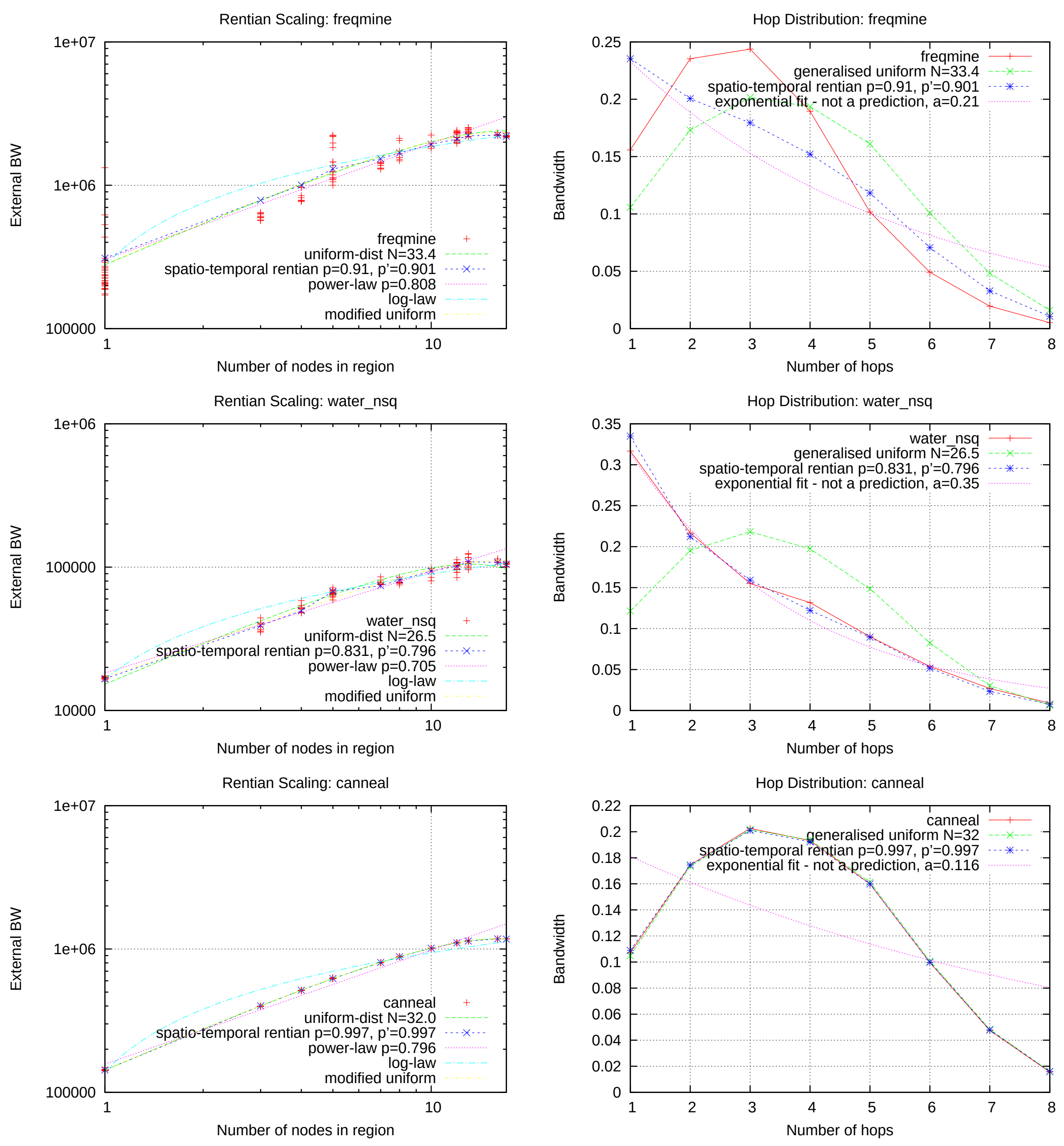


Figure 5.2: A small selection of Rentian fits for benchmarks running on a 32-node CMP simulator, and their resultant predicted hop-distributions. While *freqmine* has an almost equally poor Rentian prediction compared to the uniform model, *water_nsq* is very close, and *canneal* has a limiting-case behaviour of uniform traffic in the Rentian model ($p \approx 1$ implying no locality). For a comprehensive set please see Appendix A.

has a limiting-case behaviour of uniform traffic in the Rentian model ($p \approx 1$ implying no locality). More objectively, we can determine the RMS error of these hop-length predictions and these are shown for the entire benchmark in figure 5.5. Although the Spatio-Temporal Rentian model has a smaller RMS error in almost all the benchmarks compared to the generalised uniform, for many it is quite comparable. For *ferret*, both models have very high RMS errors, whilst for some like *canneal*, *fft* and *swaptions*, both models have very low errors. The latter case corresponds to the limiting Rentian

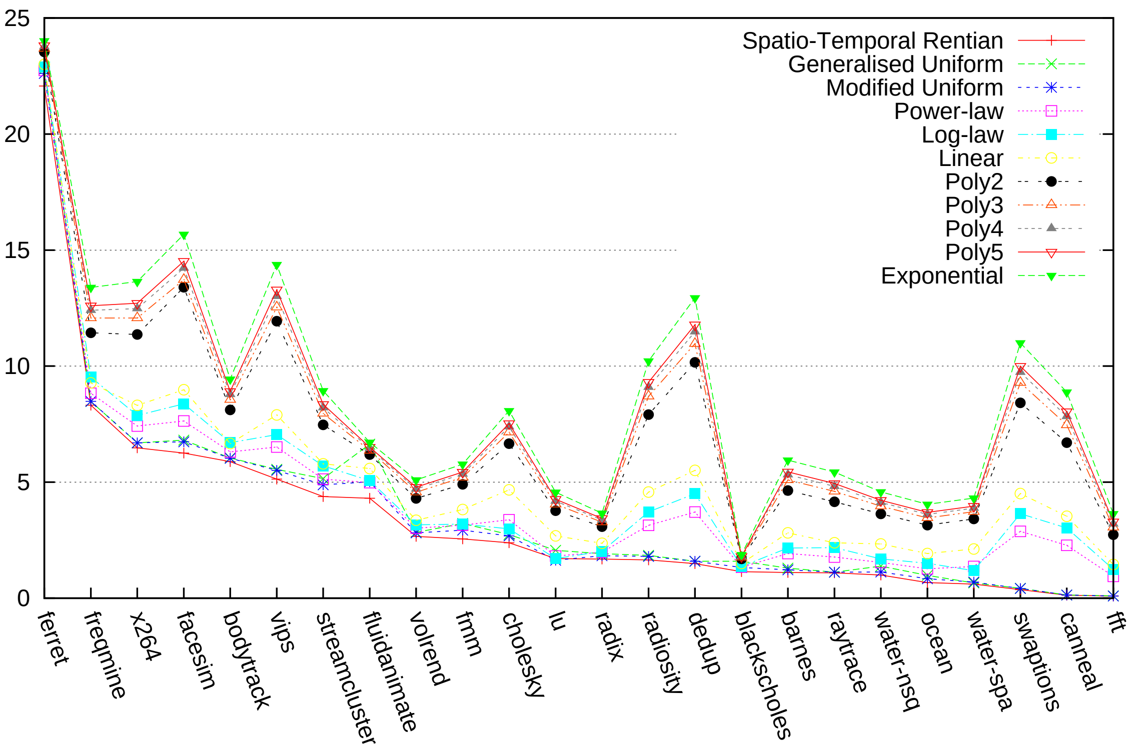


Figure 5.3: RMS error (including data's variance) of BW versus node fits

model of uniform traffic (i.e. $p \approx 1$). As an interesting point for comparison, we also added a direct exponential fit for the hop-distribution and measured the RMS error from that. Note that unlike the uniform and Spatio-Temporal Rentian models, this is not even a prediction, but merely a least-squares fit. We note that the predictions from the Spatio-Temporal Rentian model still manage to outperform the exponential fits for most benchmarks.

We can also compare the predictions for aggregate bandwidth. Table 5.1 shows the predicted versus actual aggregate bandwidth for the benchmark applications. Aggregate bandwidth here corresponds to the total distance-weighted cost of communication. For the Rentian prediction it takes the hop-length distribution and weighs it by the hop-distance and then multiplies it by the number of nodes and Rent's parameter for average bandwidth per core (also from the Rentian fit). As can be seen, for most of the benchmarks, the estimated aggregate bandwidth is also quite close (within $\pm 5\%$) to the actual value. When it comes to predicting communication energy consumption, there is dynamic energy consumed that is proportional to the link distance covered [7], as well as a component based on router utilisation. Thus accurate predictions of both the aggregate bandwidth and hop-distribution should allow prediction of NoC energy consumption as well. We should note that in a Rentian model, although most of the traffic may be local, the energy consumption is most affected by the tail of the distribution, since costs are weighted according to traversal distance.

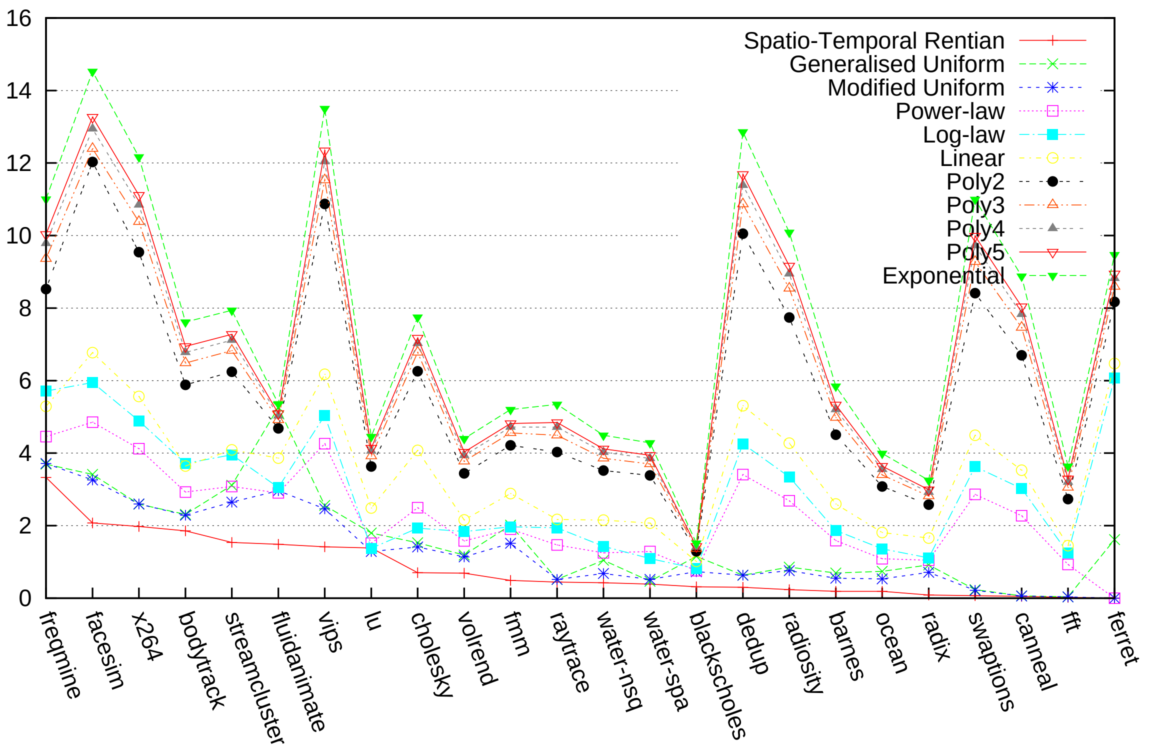


Figure 5.4: RMS error (excluding data's variance) of BW versus node fits

5.2.4 Discussion

For a model with only two free parameters it is quite remarkable how close the actual versus predicted hop-distributions are. After-all it is not the hop-distribution itself that is being fit, but rather the Rent's exponent and average bandwidth. We should further note that the average bandwidth has no bearing on the hop-*distribution* since it is normalised out. Thus there is really only one free parameter for the hop-distribution, namely the Rent's exponent that is determined separately.

It is also important to note that these results do not *necessarily* rule out Rentian behaviour on some benchmarks – instead they show that for some it is hard to differentiate between the Rentian model and an alternative one, making it more unlikely. Benchmarks where it is safe to say it is unlikely to be Rentian are *ferret*, *freqmine*, *water.spa*, and *x264*. Note that there are special cases *dedup*, *ferret* and *x264* which involve more than one thread per core, where we would expect the unconstrained (semi-random) mapping of threads to introduce greater non-locality. Indeed, Barrow-Williams et al. [8] demonstrated that there is greater locality in these benchmarks when examining the thread-to-thread communication rather than the core-to-core communication, so that it is possible that a constrained communication-optimised thread mapping would lead to Rentian behaviour.

The benchmarks which appear to be uniform-traffic also have Rentian locality of $p \approx 1$. These are *dedup*, *canneal*, *fft*, and *swaptions* (of which *dedup* is a special case as dis-

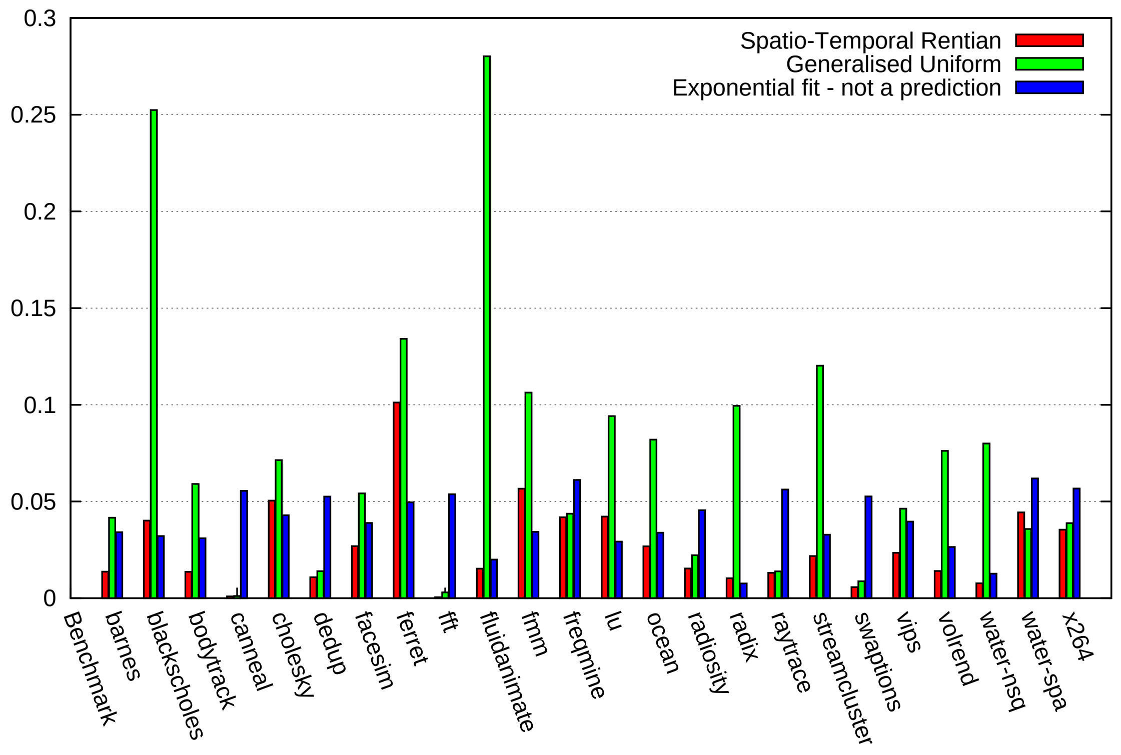


Figure 5.5: Hop-distribution RMS prediction errors (based on Bandwidth vs. Nodes scaling fits)

cussed above). Benchmarks which appear to follow the Rentian model with $p < 1$ are *barnes*, *blackscholes, bodytrack, facesim*, *fluidanimate*, *fmm*, *streamcluster*, *radix*, *water.nsq, volrend.* Those benchmarks which may be Rentian but which have almost equally good predictions as the generalised uniform models are: *cholesky*, *raytrace*, *radiosity* and *vips*.

The best-fit spatio-temporal and spatial Rent's exponents are listed in table 5.2 along with a judgement of whether it is likely to be Rentian. The criteria chosen was that the hop-length distribution RMS error should be less than 0.05 and should outperform the uniform model by at least 2:1. Also, where the normalised RMS error of the Rentian fit was at the higher end (3.0 or more), or if the aggregate BW prediction error was more than 5%, it was downgraded in category.

The spatial Rent's exponents are calculated from the spatio-temporal ones by $p_{spatial} = \max\left\{2 - \frac{1}{p_{ST}}, \frac{1}{2}\right\}$, where the spatial rent's exponent are necessarily at least 0.5 for a 2-dimensional spatial topology. For more details of the spatio-temporal model please consult Chapter 7.

Looking through the exponents, the uniform-random benchmarks clearly stand out with near unity Rent's exponents. We also note the special case of *fluidanimate*, where the spatio-temporal Rent's exponent of about zero implies that it is one-dimensional in its communication structure, like a linear dependence chain.

Benchmark	Actual Aggregate BW	Rentian predicted	% Rentian error	Modified Uniform Predicted	% Uniform error
barnes	1746786	1748633	0.11%	1897779	8.64%
blackscholes	11884	11458	-3.58%	18971	59.63%
bodytrack	3656176	3893234	6.48%	4283584	17.16%
canneal	8442467	8454134	0.14%	8474101	0.37%
cholesky	5274672	5266977	-0.15%	6300917	19.46%
dedup*	36443085	37009511	1.55%	37986386	4.23%
facesim	56118910	59181378	5.46%	65678526	17.03%
ferret*	11188999	13236903	18.30%	19684363	75.93%
fft	239193	239379	0.08%	240576	0.58%
fluidanimate	2354296	2168705	-7.88%	4173220	77.26%
fmm	1274182	1260237	-1.09%	1643574	28.99%
freqmine	14312190	15458636	8.01%	16494845	15.25%
lu	808769	778992	-3.68%	967681	19.65%
ocean	421679	419407	-0.54%	471679	11.86%
radiosity	14858580	14744439	-0.77%	15464948	4.08%
radix	186723	184412	-1.24%	223855	19.89%
raytrace	1022003	1051469	2.88%	1078492	5.53%
streamcluster	6193909	6354654	2.60%	7703215	24.37%
swaptions	20044115	20125571	0.41%	20450646	2.03%
vips	42144523	43722867	3.75%	47894675	13.64%
volrend	463263	487414	5.21%	554583	19.71%
water.nsq	753212	721876	-4.16%	843360	11.97%
water.spa	447880	460296	2.77%	507438	13.30%
x264*	23880716	25617844	7.27%	27584094	15.51%

Table 5.1: Predicted vs. actual aggregate bandwidth. Asterisks denote benchmarks with more threads than cores, and unconstrained thread mapping to cores.

5.3 Evidence in dynamic data-dependency-graphs

5.3.1 Earlier work on fractal properties in software

Some previous work [31] has shown that in an object-oriented system, the static network of classes where edges represent relationships between the classes, exhibits fractal behaviour, and that the dimensionality may even be a good metric as to the complexity of the software system. There has been some speculation as to whether program behaviour exhibits fractal properties [77], however as far as we are aware, prior to our first paper [45], there had been no work demonstrating that the dynamic graph of operations in software *actually* exhibits fractal behaviour. The dimensionality of such fractal behaviour would be of interest in understanding the intrinsic complexity of communication within the software.

At different scales, the dimensionality is a measure of the complexity of communication

Benchmark	Likely to be Rentian?	ST $\hat{p}$	Spatial $\hat{p}$
barnes	Yes	0.91	0.90
blackscholes	Yes, but with borderline RMS error	0.36	0.50 (min)
bodytrack	Yes, but high variance	0.89	0.88
canneal	Uniform ($p \approx 1$)	1.00	1.00
cholesky	Unclear (hop dist RMS higher)	0.81	0.76
dedup*	Uniform ($p \approx 1$)	0.97	0.97
facesim	Perhaps, but high variance	0.88	0.87
ferret*	No (unconstrained thread mapping)	0.51	0.50 (min)
fft	Uniform ($p \approx 1$)	0.99	0.99
fluidanimate	Perhaps, but high variance, and with $p = 0$	-0.03	0.50 (min)
fmm	Unclear (hop dist RMS higher)	0.71	0.58
freqmine	No	0.91	0.90
lu	Unclear (hop dist RMS higher)	0.76	0.69
ocean	Yes	0.87	0.85
radiosity	Unclear (almost uniform?)	0.94	0.94
radix	Yes	0.79	0.73
raytrace	Unclear (almost uniform?)	0.97	0.97
streamcluster	Yes	0.78	0.72
swaptions	Uniform ($p \approx 1$)	0.98	0.98
vips	Unclear (high variance, low RMS hop dist ratio)	0.90	0.89
volrend	Yes	0.86	0.83
water.nsq	Yes	0.83	0.80
water.spa	No	0.89	0.88
x264*	No (unconstrained thread mapping)	0.91	0.91

Table 5.2: Categorisation of benchmarks, and their estimated Rent's exponents. Asterisks denote benchmarks with more threads than cores, and unconstrained thread mapping to cores.

between operations, between blocks of operations, functions and higher level assemblies of code. In a CMP-setting this fractal dimensionality places constraints on the communication complexity between cores regardless of the level of the software mapping.

There have also been clues as to the fractal nature of communication if we look in the temporal domain. A number of observations have been made [108] that the time-evolution of traffic appears to exhibit long-range self-similar traffic that can be characterised by a Hurst parameter. However, since this also involves an interaction of software with the network, this does not comprise definitive evidence that the graph underlying software communication is fractal in nature.

5.3.2 Fractal behaviour in software

We were primarily interested in the intrinsic structure of communication of executing software instructions and what it means for CMPs. Thus we avoided explicitly structured multi-threaded applications, and instead decided to use the dynamic data-dependency graphs of single-threaded applications. Much like VLSI networks consist of logic gates

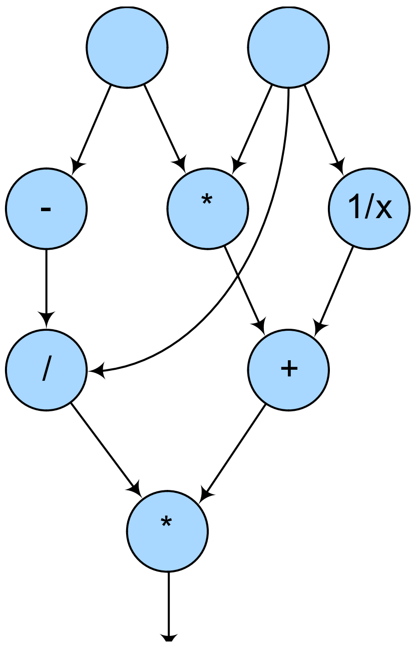


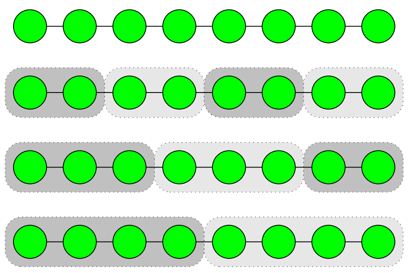
Figure 5.6: Example of a Data-Dependency Graph

connected via wires, we can also view the data-flow in software as instructions connected via registers/memory/NoC. These graphs can be mapped to both multicore or single-core architectures, so we are interested in the properties and limitations imposed by them for communication. By extracting such a data-dependency graph in a benchmark application, one can determine whether fractal connectivity properties are present that would imply Rentian scaling.

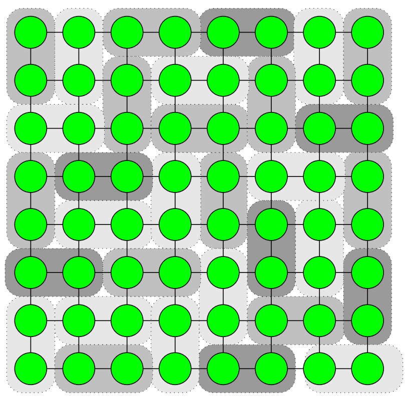
In figure 5.6 we see an example of such a graph. with nodes representing instructions and edges representing word-sized communication. Typically instructions will have two inputs (or more for complex instructions) and one output that could fan out to multiple other instructions. Such abstracted graphs of computation capture information flow between executed instructions, and thus optimistically contain all the control information in advance. This is equivalent to a perfect oracle expanding the program graph into exactly the graph that needs to be executed without any speculative operations. While this is fine for certain statically-fixed algorithm graphs, such as the fixed butterfly-graphs in FFTs, this is unrealistic for algorithms with highly non-predictable graph expansions. Nonetheless, assuming this implicit ‘perfect oracle’, provides at least a minimal bound of communication required for executing without it. We address this assumption in section 5.4, where we instead examine the connectivity of actual instructions without the need for a perfect oracle.

5.3.3 Box-counting dimension

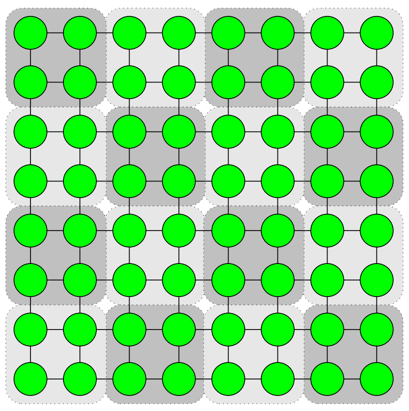
Box-counting is a methodology used for analysing fractal dimensionality. It involves a length or ‘yardstick’ of box that tiles the set. For example to measure the dimensionality



(a) Box-counting of 1D graph (lengths 1-3)



(b) Box-counting of 2D graph (length 1)



(c) Box-counting of 2D graph (length 2)

Figure 5.7: Box-counting of example 1D and 2D graphs.

of a region, one can cover it with boxes of a certain side-length. As the length of the boxes shrinks, the number of boxes scales asymptotically according to some exponent. That exponent corresponds to the box-counting dimension. For a two-dimensional object, like a filled circle, the number of boxes will scale according to $\Theta\left(L^{-2}\right)$ for box side-length L, however for fractal objects the exponent will be a non-integer – i.e. fractal dimensional.

Song et al. extended the idea of box-counting to the network domain by coming up with an analogue to 'box-length'. In their definition, a box of length L contains nodes which are all at most $(L-1)$ hops away from one another. The box count is then given by the minimal number of boxes of length L that can tile the entire network. Note that this may include boxes that are of smaller 'lengths', what matters is that they are tiled and cover the entire network. Song et al. [95] showed that the scaling behaviour wasn't as sensitive to finding the actual minimum, which is computationally very demanding (NP), and instead that the exponent could be extracted even with randomly chosen nodes as seed nodes. Figure 5.7 shows an example of how box-counting works on a linear chain (a) with four boxes of length two, three of length three and two of length four. With large chains, the asymptotic scaling behaviour would be of $\Theta\left(L^{-1}\right)$. In figure 5.7(b) and (c) we see it applied to a 2-D graph with lengths two and three respectively. Note that this serves to illustrate the general technique of box-counting. In practise, the incremental box-counting of Concas et al. [31] is used, which is more efficient by merging candidate boxes together, whilst being observed to still preserve the scaling exponent.

5.3.4 Methodology

We constructed a tool using Pin [72] to instrument x86 applications running under Linux. This was used to construct the dynamic data dependency graphs of operations over the trace. In order to ensure that the initialisation steps were skipped and the important part measured, we introduced a trigger within each benchmark application to start capture.

A suite was constructed, mainly comprising of the MiBench benchmark suite [46], plus a few additional algorithms. Together this comprised a wide variety of algorithm types including multimedia, scientific computing, graph methods, encryption, compression, and networking domains.

To look for and measure the possible fractal behaviour, we utilised Concas et al.'s [31] approach of incremental box-counting. In their approach, a box of length L is one that contains nodes that are at most $(L - 1)$ hops away from every other node in the box. The algorithm tiles the graph with boxes of length L and then counts how many there are. It starts with every node belonging to its own box. For each box length iteration, it randomly selects seed boxes and greedily merges with neighbouring boxes so long as the resulting box has maximal length L, and until there are no more box merging opportunities. The box length is then incremented and the previous box allocation is used as an input for the next iteration. To give an example, in a 3D mesh, the number of nodes in the boxes will tend to grow by the cube of L, and thus the number of boxes will be tend to shrink according to L^3. By plotting the number of boxes to box length on a Log-Log scale, one can then see if such behaviour is indeed exhibited, and if so, what the dimensionality is.

When merging two candidate boxes, ensuring that all the internal nodes of the two boxes are at most $(L - 1)$ hops away from every other node can become computationally expensive. Because of this, we only used 100,000 graph nodes for our analysis. Even achieving this required some optimisations of the former algorithm [31], as calculating or storing a non-sparse 100k x 100k matrix of distances between nodes was not practical. We did this by utilising the properties of the fractal graph, namely that for each box, the boundary nodes should be small compared to the number of internal nodes. As we were only interesting in calculating the maximal distances between the nodes of merging boxes, we noted that there are internal nodes which are discardable. For two internal nodes X and Y in a box B, if X has greater or equal distance than Y to each of B's boundary nodes, then X dominates Y, and we can discard Y if we keep X. This reduces the storage and computational requirements considerably. These optimisations do not affect the correctness of the algorithm. For graphs that do not exhibit fractal properties, it is possible that the storage and computational requirements are not tractable. Indeed, we discovered that the graphs of some benchmarks could not be analysed with even 10,000 node graphs, whereas others had no problem scaling to millions of nodes.

5.3.5 Results

We start by examining the results for a known non-fractal dimensional graph. Looking at Fig 5.8, we see that a uniform-random graph, of the form of Erdös-Rènyi, exhibits a rapidly accelerating decrease in box counts with box-length, and is certainly not fractal [91]. This is because most nodes are only a short distance from other nodes, and so upon merging nodes into boxes, the connectivity of the boxes is very high. We could not run a large random graph to completion, due to the performance issues discussed in section 5.3.4, so only a 1,500 node random graph was used here for illustration.

To validate the dimensionality extraction, we also tried a linear graph (consisting of a simple uni-directional chain), also seen in Fig 5.8, noting that it does exhibit dimensionality very close to one. There are some artifacts at the very lowest and largest box sizes, and this has to do with the random samplings and isolation. Ideally, the box count should be the minimum possible number of maximally-packed boxes for a particular side-length. So for a one-dimensional graph of ten nodes, a side-length of two should yield five boxes. However, random seed selection means that some pairs will be merged on either side of a lone node, orphaning that node, and increasing the box count. With incremental merging, this isolating behaviour tends to be preserved, until large box-lengths where there are very few boxes left over. This behaviour is seen through most of the plots, and in order to extract a linear fit for the central portion, we ignore box lengths lower than four and box counts lower than sixteen, as these appear to be the values at which these artifacts appear for most plots. We apply a least-squares fit over this region to find the slope. Many benchmarks exhibit very clear fractal behaviour.

The results for the benchmark suite are presented in Appendix B, although we present some select ones here. For the *lame* benchmark (figure 5.9b) that performs MP3 encoding, we observe a fairly good fractal relationship over the region with dimensionality of about 2.6. A similar behaviour and dimensionality is seen for the HTML typesetting benchmark *typeset* (figure 5.9a), while *rijndael* (figure 5.9c) has a dimensionality very close to three. Several highly sequential benchmarks such as *CRC32*, *adpcmenc* and *bitcount* (figure 5.9f) were observed to be quite linear in their behaviour. Whereas we note that *qsort* (figure 5.9e) does not appear fractal in its scaling behaviour. This makes sense since a binary tree is not a fixed dimensional graph – with n entries it can only be embedded with edge length $O(1)$ in a hypercube of dimension $O(\log n)$. The *tiff2rgba* benchmark (figure 5.9d) appears to have different scales exhibiting different near-linear fits, possibly hinting at multi-fractal behaviour.

A couple of benchmark graphs did not complete box-counting analysis, because of very high inter-connectivity. We were initially surprised that *tiff2bw*, a benchmark that converts an image from colour into black and white, behaved like this, but a closer look revealed that a lookup table (LUT) was being used for the conversion, and that this was causing every pixel conversion to reference these values, resulting in very high connectivity. In practise, a CMP implementation should create local copies of the LUT or

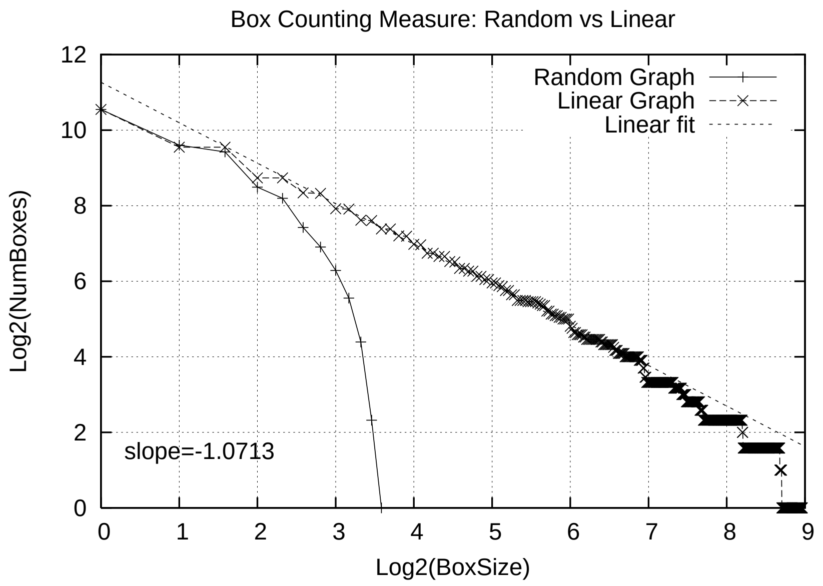


Figure 5.8: Box-counting measure comparing a Random Erdös-Rènyi graph to a linear graph, each with 1500 nodes

substitute them with in-place calculations, thereby eliminating these large hubs, however we did not manually or automatically transform the graphs in this way – potential future work that we believe may be worth investigating. This only highlights that an algorithm's communication concerns motivate us to alter them for the CMP domain. Analysis of the dynamic data dependency graph may be one way to pinpoint suitable targets.

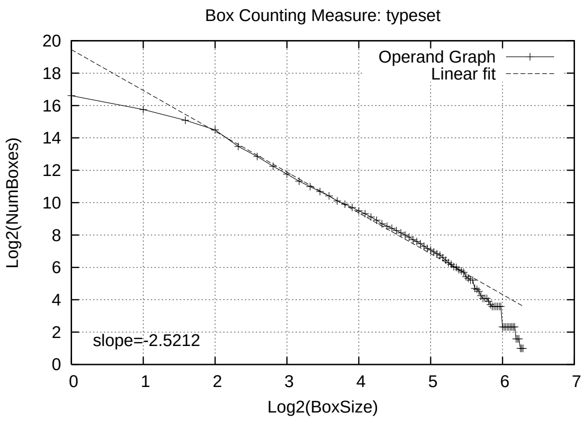
5.3.6 Dimensionality of communication vs. parallelism

It should be emphasised that low dimensionality does not necessarily indicate lack of parallelism, and nor does high dimensionality necessarily indicate lots of available parallelism. Indeed high dimensionality may indicate that communication will be a bottleneck constraining parallel speed-up.

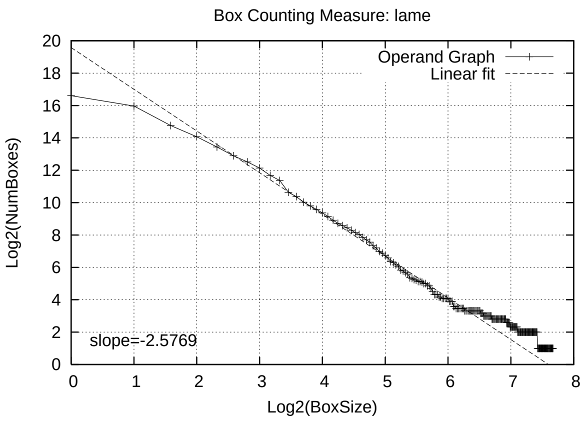
To give two examples: if a node launches ten strictly sequential graphs, the resultant graph would still be 1D, but there would be an available parallelism of ten. Conversely, if a 3D graph of operations were linked together such that there is a sequential dependency from one to the next, then the graph would still not provide an opportunity for parallelism, even though the dimensionality of communication is three – it can only be embedded within the one dimension of time. This is because data dependencies must observe temporal ordering, whereas they are unconstrained in the spatial dimensions.

5.4 Rentian analysis of dynamic-data-dependency graphs

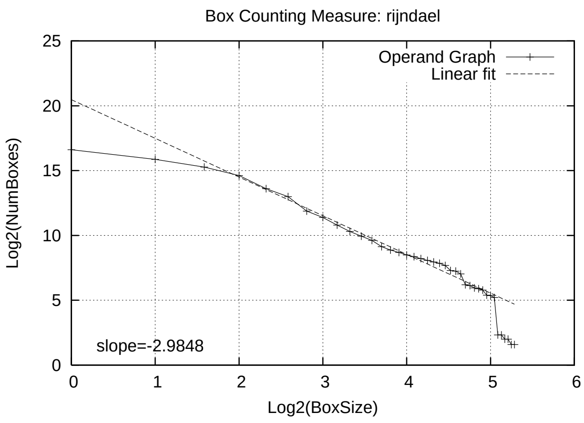
We can take the dynamic data-dependence-graphs of Section 5.3 and apply the tools of topological (partitioning) and physical (placement) Rentian analysis. We can then



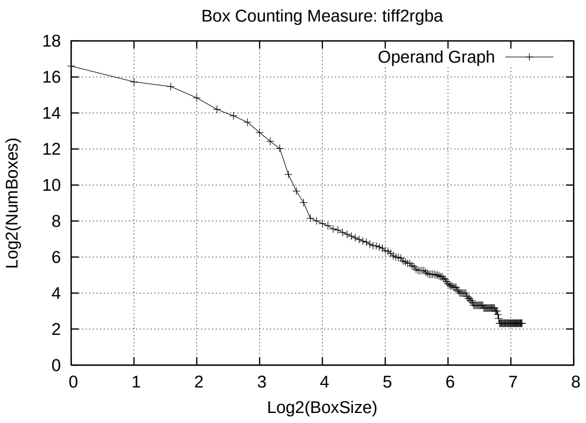
(a) Box-counting measure of the Typeset benchmark



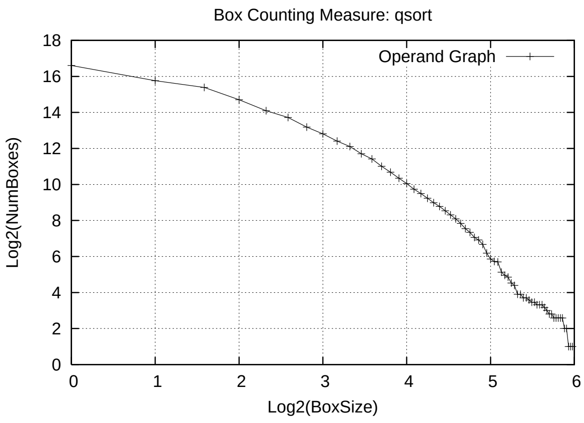
(b) Box-counting measure of the Lame benchmark



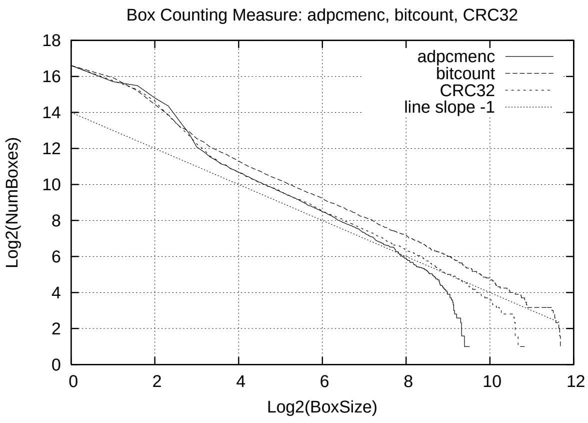
(c) Box-counting measure of the Rijndael (aes) benchmark



(d) Box counting measure of the tiff2rgba benchmark may suggests possible multi-fractal behaviour, whereby the dimensionalities are heterogeneous at different scales.



(e) Box counting of the quick-sort benchmark doesn't appear fractal



(f) Near-unity dimensionality of adpcmenc, bitcount and CRC benchmarks

Figure 5.9: A selection of box-counting plots from the MiBench benchmark suite. For a complete set, please refer to Appendix B.

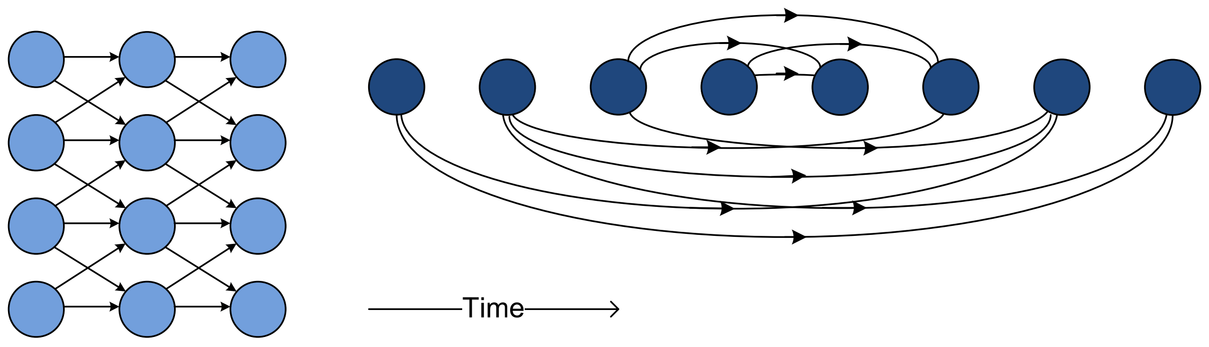


Figure 5.10: Temporal Interconnect

examine the scaling of nodes to edges for Rentian behaviour as well as predictions for distance-distributions, but in the temporal domain.

5.4.1 Temporal communication and approximate distributions

When mapping operand graphs, we are mapping them both in space and in time. Not only is there a spatial distance, but also a temporal distance, and much like longer spatial distances consume more resources, so too do longer temporal distances. In figure 5.10 we see an example graph on the left that has a natural spatial embedding for four cores, with short spatial and temporal communication lengths between them. A layer of the example graph is embedded into a single core on the right. The embedding eliminates spatial communication but comes at the cost of increased temporal-distances. Every temporal link requires memory for storage, a limited on-chip resource which is used up for the duration (length) of the link. Rather than provide a uniform performance cost per memory access, the memory hierarchy is a statistical attempt at minimising costs. For short range communication, register files are used, followed by L1, L2 and L3 cache respectively, and then external memory or even virtual memory. With greater temporal distance, there is a greater resource and performance cost, thus it is also important here to minimise temporal communication distances.

In our communication-centric paradigm[3] it helps to re-examine what services memory is providing. On the one extreme it acts as a wire in time, directly connecting data from one time to another, and on another extreme it can be used to logically act as a lookup table, possibly replacing whole functions. Both extremes are accommodated by the view of memory as acting as a switch that routes data from one moment in time to the next. Thus the concepts of memory and the NoC may not be so fundamentally different after all. We can view memory as routing primarily in time, and the NoC as routing primarily in space, but really they act together to route data in space-time.

Under the constraint of minimising distances, it might be reasonable to expect Rentian approximate power-law distributions in the temporal distances. We were pleased to find

[3]see Chapter 2 for more details

that Hartstein et al. [49], did indeed observe a power-law distribution for the inter-access times of cache-lines, much to their surprise, as they had expected an exponential distribution. They constructed analytical cache models based on this distribution and showed that it could reasonably model and predict cache behaviour (upon varying cache architecture). They also showed that by assuming a power-law, they could explain the empirically observed but unsatisfactorily explained scaling law for cache misses, which they relate to the power-law exponent of cache line accesses for the application. However, they did not have a satisfactory explanation as to why such a power-law behaviour was present in the first place.

We should note, however, that their results are based on the distance of cache line accesses, whereas our analysis looks at the individual links of operands between executed instructions. The power-law distribution of operand-distances does not automatically follow from such a distribution in cache lines, or vice-versa, as a cache line combines many words of data and thus also involves address-locality within the cache line as well. It would be interesting to establish that power-law temporal-distance distributions do exist at the operand scale as well.

Recently, Clauset et al. [30] examined phenomenon previously claimed to be governed by a power-law *probability distribution*, and demonstrated that with a more robust statistical analysis, that many of these were not likely to have been drawn from a perfect power-law distribution afterall. They recommended the use of Maximum Likelihood Estimation (MLE) for parameter estimation and then the use of the Kolmogorov-Smirnov test and likelihood ratios to determine the goodness-of-fit against other probability distributions. For our distance distributions, we do not make the claim that there is an underlying power-law distribution, from which the data-set represents independently drawn samples. Indeed, because of the nature of repeated loop execution, there are many links with 'magic' lengths that violate any independence. This can result in a certain large distance L being favoured by orders of magnitude compared to distance $L+1$, even though at each scale (with logarithmic binning) the trend is roughly power-law in nature. Thus a strict application of the Kolmogorov-Smirnov test will rule out any smoothly-varying candidate distribution such as the power-law, exponential, log-normal, etc. However, Clauset et al. also point out that the importance of whether or not it *perfectly* fits a power-law distribution depends on *how* the distribution is used. They distinguish, for example, between the scientist trying to construct models of how a perfect power-law emerges, versus the engineer who needs a reasonable approximate model with which to make decisions. In our case, we are less concerned with whether or not it perfectly fits, indeed we already know that it cannot fit perfectly, however this should not stop us from applying it as a simple and useful approximate model. For similar arguments, we would expect the distance distribution of VLSI circuits to also fail Clauset's criteria, and yet Rentian wire-length distribution models yield useful predictions.

5.4.2 Physical (embedded) analysis

Instead of trying to find an optimal embedding for the instructions by simulated annealing, we can simply use the instruction sequence as an existing embedding. This also avoids the problem of needing an 'oracle' for the graph, since we are simply taking what is dynamically generated by executing the instructions as-is.

Plots for the entire MiBench benchmark suite are shown in Appendix C, however a selection chosen for their diverse behaviour is also presented in figure 5.11. They show the scaling behaviour for 2 million instructions of their data-dependency graphs. The meaning of 'placement' here is the actual instruction execution position in time – so the first instruction has position one and the last has instruction position two-million. The communication graph is then segmented approximately equally in time into eight regions starting from segment 1, and ending at segment 8, and these are labelled separately within each plot. The Rentian scaling properties of 'terminals' (cut hyper-edges) to nodes (instructions) is shown on the left. The scaling of the distance distribution of accesses is shown on the right hand side.

All plots are in log-log form for power-law fitting. Distance distribution frequencies are logarithmically binned. A shallower distance distribution slope implies less locality, as it means there is more communication at further distances. For Rentian distributions, there is a limiting value in the slope of -1 which corresponds to infinitely large dimensionality in the communication graph's connectivity. One-dimensional graphs (serial communication) are a limiting-case, with distance distribution slope of -2 before the model breaks down allowing steeper slopes. For the Rentian plots on the left, one should note that there are different sizes in the sample points. Here, the size of the cross corresponds to the degeneracy (number of identical samples) at that sample point. We should note that all fits are over the entire data-range, and therefore may include Region II effects that may alter the values of the slopes.

The selection of benchmarks in figure 5.11 was deliberately chosen to showcase a wide variety of different behaviours, however many of benchmarks typically exhibited approximately Rentian behaviour, as seen in Appendix C. Looking at *adpcmenc*, we can see that it has a fairly flat Rentian scaling exponent (close to zero), making it look like a linear-chain in its communication structure, with only on the order of tens of words of data being communicated even at large scales of the hierarchy. The corresponding distance distribution starts off looking approximately power-law but is truncated very early at a link distance of under a hundred instructions, and with frequencies higher than one. This also indicates a simple sequential structure with dependencies between stages of at most 100 cycles. In comparison, *typeset* has a steeper Rentian scaling exponent and is not truncated early in its distance distribution. We can also see some variation in exponent for each of the eight regions of time. In the *fft* benchmark we can see several separate heterogeneous scaling behaviours. Importantly, the steepness of the *fft* slopes on the left-hand scaling graph gets steeper with the number of nodes instead of being

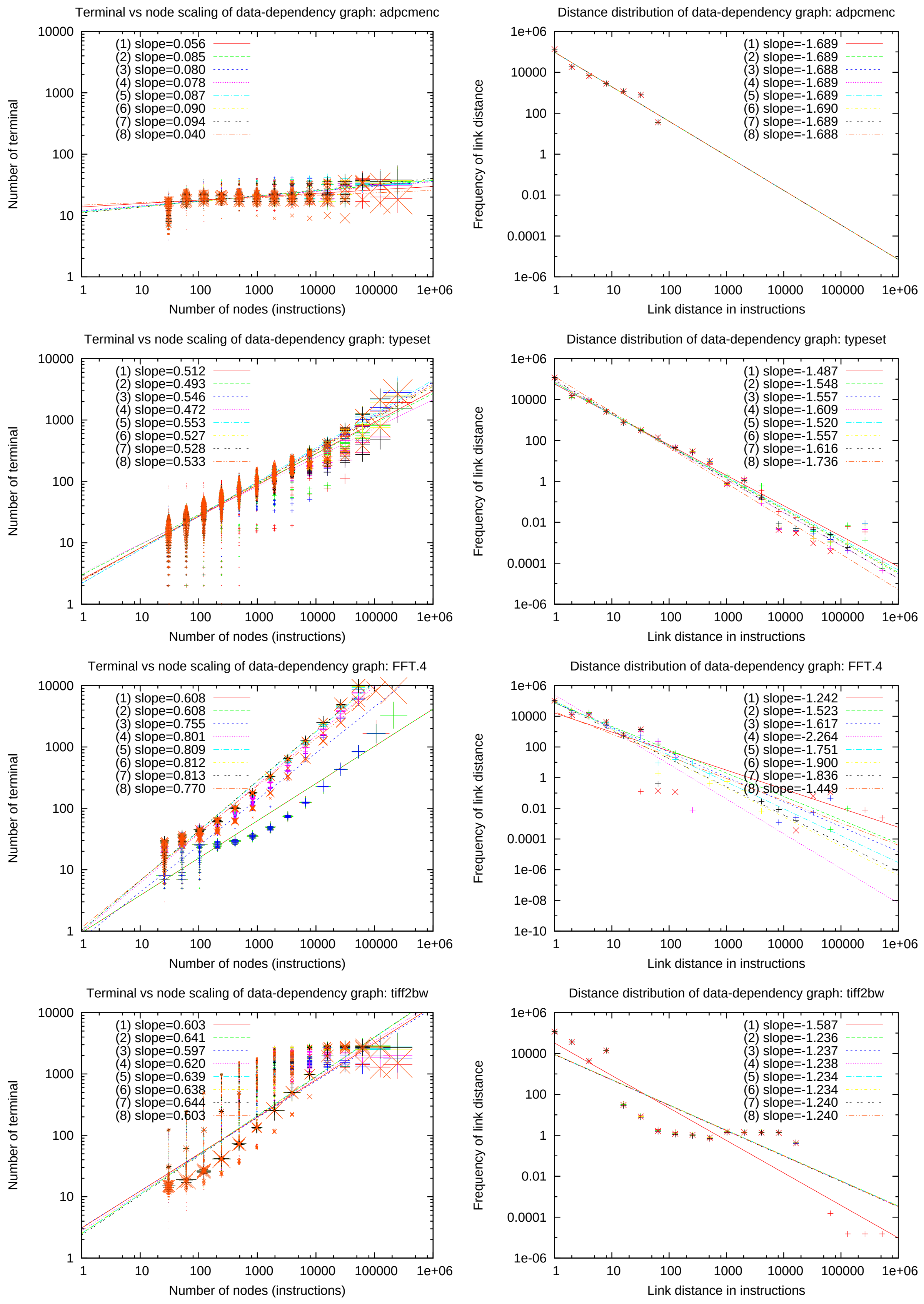


Figure 5.11: A small selection of Rentian fits and their temporal-distance distributions. For a complete set please refer to Appendix C.

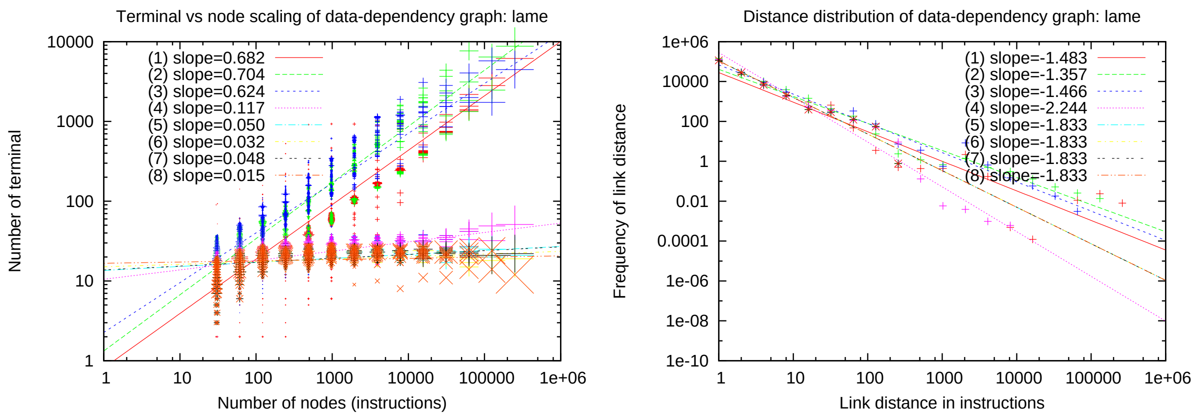


Figure 5.12: Heterogeneity in the *lame* benchmark across different times. Two main Rentian scaling distributions can be seen with corresponding effects on their distance distributions.

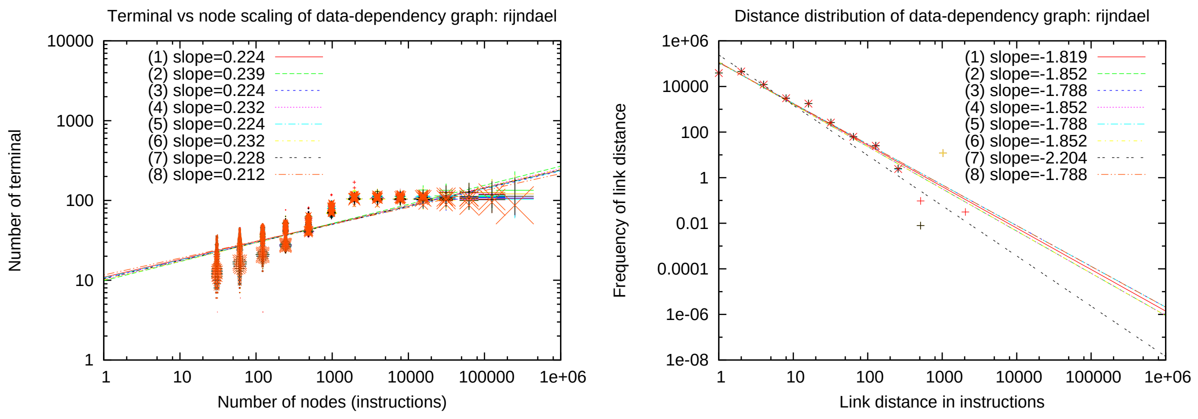


Figure 5.13: Abrupt change in scaling behaviour for *rijndael*

approximately linear in the log-log domain. This indicates it is non-Rentian in its scaling behaviour, and indeed we would expect this for the *fft*[4]. The corresponding distance-distributions are also more scattered, with poor locality for some of the regions. Finally in the *tiff2bw* benchmark, we see a clear violation of approximate power-law distance distributions, as well as poor Rentian scaling.

Taking a close look a the *lame* benchmark, shown in figure 5.12, we can clearly see two heterogeneous phases of execution – an almost linear phase like in *adpcmenc*, as well as a more complex phase with higher Rent's exponent and lower associated distance distribution slopes. The presence of heterogeneity is not, in itself, surprising, as Heirman et al. [52] has already identified how the Rent's parameters can change with different phases of a program's execution. What is remarkable is that we can identify these structures at all. Taking a step back, one may ask why the data-points in this figure aren't simply scattered and indeed why they aren't merely random, instead of exhibiting differentiable structures, particularly Rentian structure?

Looking at *rijndael* in figure 5.13, we see that after about a few thousand instructions,

[4]See Chapter 8 for more details

Algorithm 5.1 Example of one-dimensional communication structure, but with a higher-dimensional substructure (in matrixMultiply).

```
R = matrixIdentity
while (exponent != 0) do
   if (exponent & 1)
      R = matrixMultiply(R, X)
   fi
   X = matrixMultiply(X, X)
   exponent = exponent >> 1
endwhile
```

the slope abruptly changes to be flat. This indicates a near one-dimensional communication structure. It helps to understand how this might occur with an example. Suppose we had a fixed size matrix multiplication and repeated the operation back to back, say for exponentiation. Then at the higher-level scale this consists of forwarding matrix multiplication results onto the next matrix multiplication operation as seen in Algorithm 5.1. This results in only linear communication needs at this level of the hierarchy, which is different from the scaling behaviour *within* the matrix multiplication. We would expect larger matrix multiplies to push out the transition point to the right (i.e. more instructions before linear scaling behaviour takes over).

For a Rentian network that is optimally embedded in one dimension (in this case 'time'), one would expect the exponents to be related. For Rent's exponent p and distance distribution exponent $-\mu$ we would expect $p + \mu \approx 2$. A plot across the benchmark segments are shown in figure 5.14. The ten percent error bars are shown for the sum, and it does appear that most benchmarks are within 10% of the Rentian model. Benchmarks with truncated distributions (marked here by circles), indicate a likely Rent's exponent of zero at larger scales, and are unlikely to obey this relationship in exponents. As a form of evidence, the plot here is certainly suggestive of this Rentian relationship, but it is not definitive – a tighter fit to the expected curve would be much more desirable. In practise, this depends on the optimality of the algorithm's embedding of instructions so as to minimise temporal communication. Unlike in VLSI, whereby a placement tool can freely shuffle gates around for its optimisation goals, algorithm designers are not necessarily motivated to optimise an implementation's temporal communication and compilers currently optimise at the fine-grain scale of tens to hundreds of instructions. Unless compilers (or programmers) can better work at optimising at a higher level, this may pose a limitation on the temporal Rentian model's predictive ability. However, given existing scaling trends, we believe that there will be increasing motivation to optimise at such a level. Distance distribution exponents below the line, should move up to the line at larger scales. Exponents above the line are sub-optimal and should be structured to move towards the line.

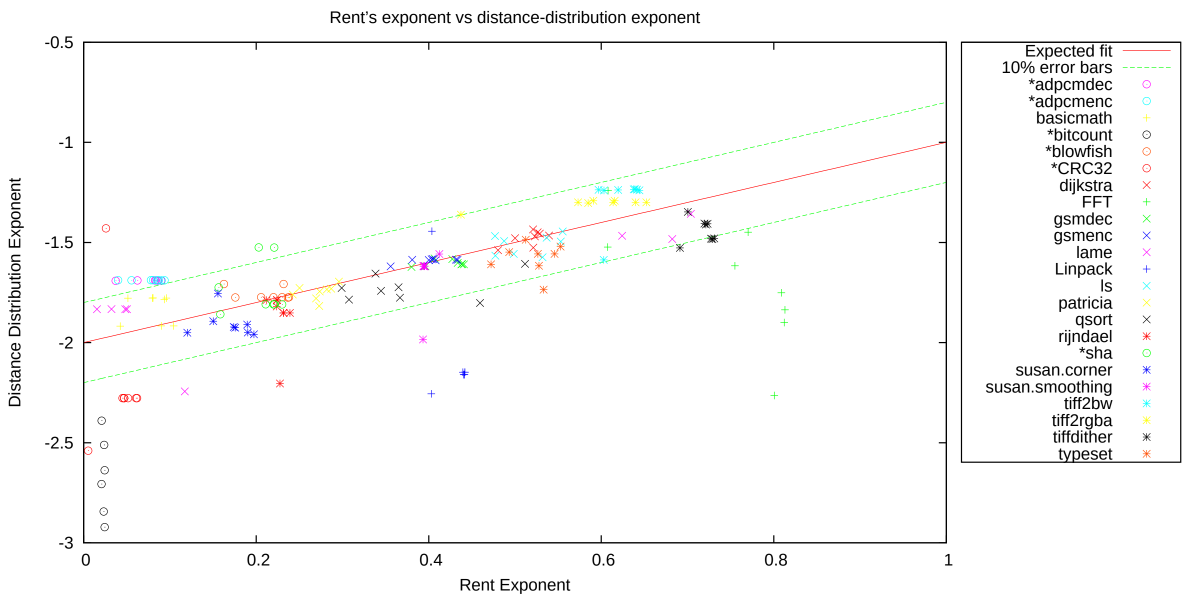


Figure 5.14: Relating the Rent's exponent to the distance distribution exponent. For a Rentian network that is optimally embedded in 1-dimension (in this case 'time'), then we expect the Rent's exponent p and distance distribution exponent $-\mu$ to be related by $p + \mu \approx 2$. The ten percent error bars are shown for the sum.

5.4.3 Topological analysis

The topological approach treats the graph in the abstract, much like the fractal box-counting analysis earlier. A selection of plots for MiBench benchmark applications are shown in figure 5.15, with the entire benchmark suite results available in Appendix D. They show the scaling behaviour for 2 million instructions of their data-dependency graphs. The 'partitioning' here uses *hMetis* [62] which tries to find a min-cut bisection of the communication graph using hyper-edges. This is done recursively for sixteen levels forming 65,536 partitions. The top three levels of bisection form eight regions which are labelled segment 1 to segment 8, which although tend to correspond to eight near-contiguous temporal segments, may not necessarily do so. These labels are shown within each plot. The Rentian scaling properties of 'terminals' (cut hyper-edges) to nodes (instructions) is shown on the left. The scaling of the distance distribution of accesses is shown on the right hand side.

The distance distribution here, as in the physical analysis in section 5.4.2, uses the actual instruction time distances but utilising the partitioning found by *hMetis* rather than partitioning by time. We should perhaps note that the min-cut problem is actually NP-complete, and *hMetis* is using heuristic approaches and annealing to achieve its result. As such *hMetis* is not guaranteed to find a minimal partitioning. One would hope that it could do better than the existing embedding of instructions in time from the previous chapter, but this is not necessarily the case.

All plots are in log-log form for power-law fitting, similarly to section 5.4.2, with identical plot attributes. We should again note that all fits are over the entire data-range, and therefore may include Region II effects that may alter the values of the slopes.

Again, for many of these plots we do see evidence of Rentian scaling. Comparing the selection in figure 5.15 to the previous physical Rentian analysis, we can note some differences. Although *typeset* is not affected much, the spread of slopes and distance distributions is markedly narrower (reduced from a range of size 0.081 to 0.020), along with the distance distribution slopes (from a range of 0.25 down to 0.094). For *tiff2bw* the behaviour is also clearer, with less variance). This could be because the tool is free to choose the partitioning and so can separate out other behaviour. For the *fft* we note a change in behaviour - it now appears to be Rentian with near constant slopes. However, one has to remember that there is an implicit context at work here – the topological partitioning is done recursively in a tree-like manner, and without regard to temporal ordering dependencies. The Rentian-like scaling property seen on the left-hand-side, then, is for communication that is embedded in a high-dimensional binary tree substrate rather than, say, a 2-D mesh with uni-directional time. This highlights the importance of also doing a physical Rentian analysis.

Looking in figure 5.15 at the benchmark *lame* for 1 million cycles, we see that there is a wide scattering of points, and it does not follow a simple Rentian scaling. However, if we look closer, we see that there are actually several Rentian scaling structures being presented at once. Again we find for *lame* a heterogeneous system with modules that have different Rentian parameters (i.e. multi-fractal) in its behaviour. Indeed we can use the top-level partitioning of *hMetis* to separate out these modules, as shown in figure 5.15. In the *lame* benchmark we see that whereas in the physical Rentian analysis we could see only two heterogeneous regimes, in the topological one there appears to be three (region 7 in black is separate from the other two main scaling trends in red and yellow). The partitioning tool appears to have found a communication structure that was not exposed in the physical Rentian analysis, possibly because this structure was embedded (by the programmer) inside a different algorithm's loop-body. This analysis suggests that it could be perhaps be separated into another spatial region.

5.5 Conclusion

Using multiple methodologies, this chapter examined and tested existing software for evidence of Rentian scaling. In addition to Heirman's work on characterising Rentian behaviour for CMP systems, this chapter more rigorously examined evidence for it by comparing against other null-hypotheses in the physically placed domain. Across the SPLASH-2 and Parsec benchmarks, good agreement with Rentian behaviour was observed on a wide variety of benchmarks. It becomes easier to distinguish between Rentian scaling and other null hypotheses as the size of the system increases, and with only 32 nodes this makes it a challenge. For a number of benchmarks, there wasn't sufficient evidence of Rentian behaviour compared to other null hypotheses, and in some cases the Rentian model was rather unlikely. However, even for those benchmarks deemed

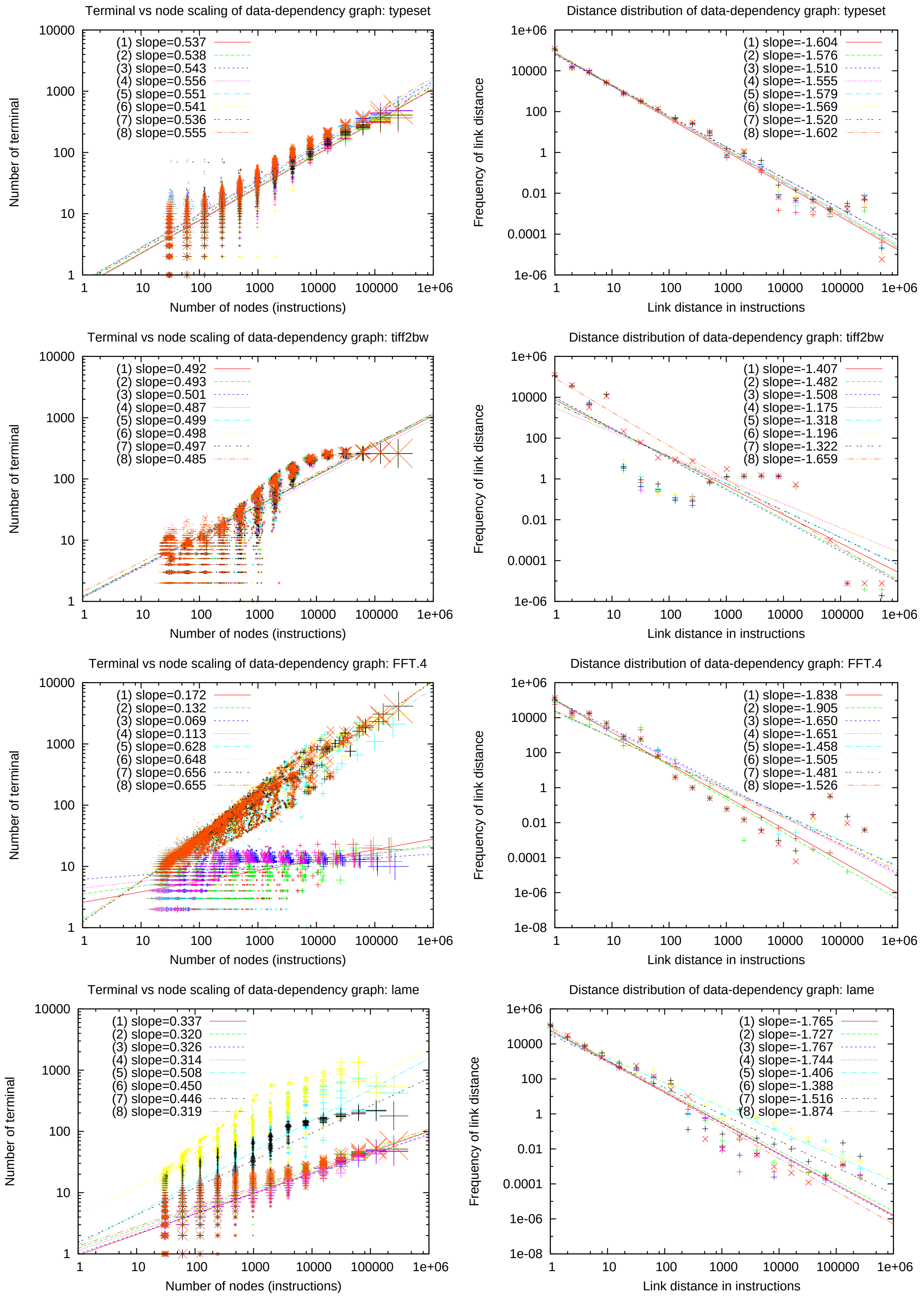


Figure 5.15: Selection of benchmarks chosen for comparison with the physical analysis.

unlikely to be Rentian, the spatio-temporal Rentian model was observed to be the best-fitting model for the growth of bandwidth versus nodes in almost all the benchmark applications. Moreover, the Rentian model had remarkably good predictions for the resultant hop-length distributions and aggregate bandwidth for most benchmark applications. This indicates that the simple model is indeed useful for predicting and characterising the hop-distribution using only one free parameter, and the aggregate bandwidth with only two free parameters. This lends further credence to the Rentian model as a characterisation and modelling tool.

To investigate the more fundamental communication structure of software, dynamic data-dependence graphs were extracted from single-threaded benchmark applications. With several techniques – box-counting from the Network Science community, and physical and topological Rentian analysis, it was seen that many benchmark applications appear to exhibit fractal scaling properties, and the temporal-distance distributions were also seen to be approximately power-law. Instead of random or non-power-law distributed scaling structures for terminals versus nodes, clear heterogeneous Rentian structures were clearly visible in the analysis corresponding to different phases of the program execution, or to different min-cut partitioning found by an automated tool. Furthermore, for the physical Rentian analysis, the distance distribution exponent and Rent's exponent, were predicted by theory to sum to two. We saw that most benchmarks were within 10% of this predicted behaviour in exponent sum.

This all indicates that the Rentian model, although not universal, is a simple, practical model that is useful for characterisation and applies to a wide variety of algorithms.

Chapter Six

6

Generalising Rent's rule

Although in Chapter 4 it was examined how Rent's rule could be generalised to the NoC domain of bandwidth, the argument focused on the interchangeability of hardware blocks with software blocks. This implies individual software tasks being specifically localised on certain cores, and acting more as a software circuit – like the PicoChip architecture [85]. More generally, however, it is not just software *tasks* that are being mapped in space, but also their individual *instructions* are embedded both in space and in time.

One would desire, then, a more complete model that unifies the concept of NoC and memory into a single Spatio-Temporal Rentian model for the CMP domain. This is where there is both a spatial component, say an $N \times N$ 2-D mesh, as well as a temporal component of memory and execution time – extending many orders of magnitude further into time, but only in one direction – forward.

Existing Rentian models from the VLSI domain deal primarily with planar logic that can be connected in any direction, rather than software which has constraints in the temporal dimension. Some extension to 3D stacked die also exist [60, 116] but they capture neither the unbounded extent nor the uni-directionality of the temporal dimension. Nor do they give an adequate account of how the relative costs of moving in one dimension versus another affect the embedding, as moving information purely in time can be much cheaper than moving it in space (and time). Furthermore, we would also like a model to support a 3-D stacked CMP (say an $N \times N \times M$ mesh) along with the fourth dimension of time.

In order to achieve this goal, one actually needs to better understand Rent's rule, and generalise its principles much farther than existing theory. To do so, in this chapter it is first proven that there is an equivalence between asymptotically power-law tailed distance distributions and asymptotic Rentian behaviour. Moreover, this proof is done in a general d-dimensional vector space, rather than just the 2-D Manhattan metric of VLSI. Previously Donath and others [40] have shown that Rent's rule leads to approximate power-law distance distributions for VLSI, but not the other way around, and not with this generality. This asymptotic equivalence is quite remarkable in itself. It is then proven that the problem of finding optimal distributions for mappings in a d-dimensional domain is equivalent to an optimisation problem in a configuration-space. It is shown that the particular power-law tails for the d-dimensional embedding, produce power-law tails in the equivalent configuration-space with an invariant exponent, independent of the original embedding dimensionality. Leveraging this invariant power-law tail exponent allows generalisation to arbitrary embedding domains, including for software running on a CMP, and perhaps even physical human-interaction networks as will be illustrated.

6.1 The principle of cost universality

It would be useful to be able to meaningfully compare two length distributions $f(l)$ and $g(l)$ for optimality of communication. Here, the concept of a length distribution is general. For example, the length distribution in a manufacturing context may correspond to actual physical wire length, and for a network-on-chip may correspond to what proportion of traffic goes a particular distance. The total cost incurred by a distribution is equal to the sum of the cost of each link. If link cost varies by distance we can utilise a cost function $c(l)$ and the total cost C over a length distribution f, up to maximum length L_{max}, as:

$$C(f) = \int\limits_{0}^{L_{max}} f(l)\, c(l)\, dl.$$

We allow f to also be a discrete probability distribution, by way of Kronecker delta functions.

Definition 6.1. We define the relation $f \succ_c g$ to mean f is more optimal than g over cost function c i.e. $C(f) < C(g)$, where $C(f) = \int_0^{L_{max}} f(l)\, c(l)\, dl$.

It is reasonable to assume that the cost function $c(l)$ strictly increases with distance (or hop count in the case of topological networks). This monotonicity, in practice, is not an important restriction to represent real monotonic cost functions, as it still allows the rate of monotonic increase to be arbitrarily small.

Under certain conditions shown below, f is more optimal than g regardless of the specifics of the cost function as long as cost is strictly increasing. We remove the subscript on the optimality relation to denote that it is independent of cost function. i.e:

$$f \succ g.$$

Definition 6.2. We define the relation $f \succ g$ to mean f is more optimal than g over all strictly increasing cost functions c iff $\forall c \in C^1 \wedge \forall l \geq 0,\ 0 \leq c(l) < \infty \wedge c'(l) > 0 :\ f \succ_c g$.

Lemma 6.3. *For probability distributions f and g with cumulative distributions F and G respectively, if $\forall l \in [0, L_{max}]$, $G(l) \geq F(l)$, where L_{max} is the maximum length of the system, then $f \succ g$.*

Proof. First we can examine the difference in costs incurred for two probability distributions as:

$$C(f) - C(g) = \int_0^{L_{max}} (f(l) - g(l))\, c(l)\, dl.$$

As by definition $F(0) = G(0) = 0$ and $F(L_{max}) = G(L_{max}) = 1$, then integrating by parts we have:

$$\begin{aligned} C(f) - C(g) &= \left[(F(l) - G(l))\, c(l)\right]_0^{L_{max}} - \int_0^{L_{max}} (F(l) - G(l))\, c'(l)\, dl \\ &= 0 - \int_0^{L_{max}} (F(l) - G(l))\, c'(l)\, dl \\ &= \int_0^{L_{max}} (G(l) - F(l))\, c'(l)\, dl. \end{aligned}$$

This is an integral of a positive function, therefore the result is positive. □

Definition 6.4. A family of distribution functions $\{f_k\}$ is said to be *'cost universal'* if any two members can be strictly ordered as $f_j \succ f_k$ or $f_k \succ f_j$, independent of cost-function.

Lemma 6.5. *The family of distributions $\{f_\mu\}$ with common head-distribution and continuous power-law tail distribution from some starting length l_{tail} and with exponent $\mu > 1$ is 'cost universal'.*

Proof. Let G and H be arbitrary members with power-law exponents μ_g and μ_h respectively where $\mu_h > \mu_g > 1$, then:

$$G(l) = \begin{cases} B(l_{tail}) + \lambda \left(1 - \left(\frac{l}{l_{tail}}\right)^{1-\mu_g}\right) & , l \geq l_{tail} \\ B(l) & , otherwise \end{cases}$$

for some normalisation parameter λ, and similarly for H, for some common c.d.f. head distribution B (including $B \equiv 0$).

We know that $\forall x, y \in \mathbb{R}^+$:

$$\log x > \log y \Rightarrow x > y$$

So that for $l \geq l_{tail}$:

$$\begin{aligned}
H(l) - G(l) > 0 &\Leftrightarrow \left(\frac{l}{l_{tail}}\right)^{1-\mu_g} - \left(\frac{l}{l_{tail}}\right)^{1-\mu_h} > 0 \\
\log\left(\left(\frac{l}{l_{tail}}\right)^{1-\mu_g}\right) - \log\left(\left(\frac{l}{l_{tail}}\right)^{1-\mu_h}\right) &= ((1-\mu_g) - (1-\mu_h)) \log\left(\frac{l}{l_{tail}}\right) \\
&= (\mu_h - \mu_g) \log\left(\frac{l}{l_{tail}}\right) \\
&> 0
\end{aligned}$$

$$\Rightarrow H(l) > G(l)$$

As any two members can be ordered irrespective of any strictly monotonically increasing cost-function, the set is *cost-universal*. □

A similar result can be shown for discrete power-law distributions.

6.2 Proving asymptotic equivalence of Rent's rule and power-law distance distributions

Let us start with a normed vector space M in d dimensions with distance metric $d(X, Y)$. A d-dimensional object Z, which resides in the vector space, is convex iff for any two points P_1 and P_2 inside Z, every point along the geodesic from P_1 to P_2 is also in Z. Let us choose an arbitrary convex shape Z, such that the longest geodesic inside Z is of exactly unit length. Let V_Z be the d-dimensional volume of Z. For an example, see figure 6.1 for a two dimensional Euclidean vector space.

Suppose we have an edge E of fixed length L which can be anywhere homogeneously in the d-dimensional vector space, and in any orientation. Let each position and orientation be uniformly probable, but with lengths obeying a prior distribution. Let us denote the start of the edge by P_1 and the end of the edge by P_2, then for a fixed start point the configuration space[1] of possible end points is given by $B_L = m(P_1 \mid d(P_2, P_1) = L,\ P_2)$, where $m()$ is a function that measures the hyper-volume of the set of points satisfying the constraints. Let $m_Z(P) \equiv m(P \in Z,\ P_2 \in M \mid d(P_2, P) = L,) = V_Z.B_L$ is the $2d-1$ dimensional volume of configuration space of possibilities for P being inside of Z, for all

[1]Concept originally from Physics where configurations of objects correspond to a position in the high-dimensional space. For example the configuration phase space of a particle is the 6-dimensional position corresponding to its 3-dimensional spatial position and 3-dimensional momentum.

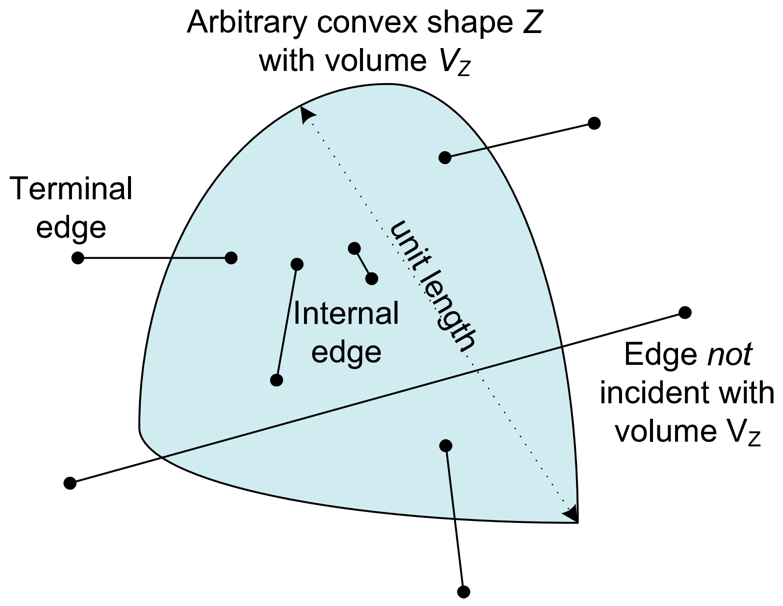

Arbitrary convex shape Z with volume V_Z
Terminal edge
unit length
Internal edge
Edge *not* incident with volume V_Z

Figure 6.1: Example of an arbitrary convex shape in 2-D

values of P_2. So $m_Z(P_1) = V_Z.B_L$ and likewise, the configuration volume of possibilities for P_2 being inside of Z, irrespective of P_1, is $m_Z(P_2) = V_Z.B_L$. However the configuration volume for both P_1 and P_2 being in Z simultaneously are further constrained by the edge length L, as:

$$m_Z(P_1,\, P_2) \equiv m(P_1 \in Z,\, P_2 \in Z \mid d(P_1, P_2) = L)$$

We note that the configuration volume of orientations B_L, grows with L, whereas it is desirable that each orientation is equally probable with total probability equal to one. So we contract the space to normalise as:

$$InternalEdges_Z(L) = \frac{1}{B_L} m_Z(P_1,\, P_2)$$

Let us denote the dimensionless, volume-renormalised function:

$$\phi_Z(L) \equiv \frac{1}{V_Z B_L} m_Z(P_1,\, P_2)$$

We note that all edges incident with the *volume* of Z must have at least one point on the inside (either P_1 or P_2), but must not double count edges when both points are inside (P_1 and P_2). These *incident* edges include both internal edges and terminals.

$$\begin{aligned} IncidentEdges_Z(L) &\equiv \frac{1}{B_L}(m_Z(P_1) + m_Z(P_2) - m_Z(P_1,\, P_2)) \\ &= 2V_Z - V_Z\phi_Z(L) \\ &= V_Z(2 - \phi_Z(L)) \end{aligned}$$

Whereas terminals are those edges which are incident only with the surface of Z and thus must further exclude all internal edges:

$$\begin{aligned} Terminals_Z(L) &\equiv \frac{1}{B_L}\left(m_Z(P_1)+m_Z(P_2)-2m_Z(P_1,\,P_2)\right) \\ &= 2V_Z-2m_Z(P_1,\,P_2|L) \\ &= 2V_Z(1-\phi_Z(L)) \end{aligned}$$

For convenience, let us define $\theta(x) = (1-\phi_Z(x))$. An intuitive way of thinking about $\phi(x)$ is to think of a stick of length x and to count all the possible combination of position and orientation of that stick that manage to fit inside shape Z. Each combination of position and orientation is equally likely, and we normalise so that at infinitesimal length, where all possible orientations and positions in Z fits inside shape Z, then $\phi(0) = 1$. As we shall see shortly, the function $\theta(x)$ has some interesting properties that we can employ in later proofs.

6.2.1 Lemmas proving properties of $\theta(L)$

In this section, we wish to prove particular properties of $\theta(L)$ that will be essential in later proofs:

1. $\theta(0) = 0$
2. $\theta(L) = 1,\, L > 1$
3. $0 < L_1 < L_2 \Rightarrow \theta(L_1) \leq \theta(L_2)$
4. $\lim_{L\to 0^+}\frac{\theta(L)}{L} = \Upsilon$, which is a finite positive constant
5. $\exists\delta > 0, s.t.\theta(L) \in C^1(0,\delta)$, where $C^1(0,\delta)$ is the set of all functions with continuous derivative in the interval $(0,\delta)$.

We start by remapping the product space of $Z\times Z$ into the space of $Z\times S\times\Lambda$ where $S = \{\mathbf{x} \mid |\mathbf{x}| = 1\}$ denotes a set of orientations, and Λ a set of edge lengths. In particular:

$$(X,Y) \to \left(X, \frac{Y-X}{|Y-X|}, |Y-X|\right)$$

Let: $\eta = \frac{1}{|S||V_Z|}$

Let $\vec{\mathbf{N_B}}(Y)$ be the normal vector for $Y \in \partial Z$, where ∂Z is the the boundary of Z.

Let us also define the identity function:

$$\delta_Z(X,\,\epsilon) = \begin{cases} 1 & if\, X \in Z \\ 1-\frac{1}{\epsilon}|X-P(X)| & if\, 0 < |X-P(X)| < \epsilon \\ 0 & otherwise \end{cases}$$

where $P(X)$ is the projection of X onto the closest point on the surface ∂Z.

Let:

$$\begin{aligned}\delta_Z(X) &\equiv \lim_{\epsilon\to 0}\delta_Z(X,\,\epsilon)\\ &= \begin{cases}1 & if\,X\in Z\\ 0 & otherwise\end{cases}\end{aligned}$$

We can write:

$$\begin{aligned}\phi(L) &= \lim_{\epsilon_1,\epsilon_2\to 0}\eta\int\limits_S\int\limits_M \delta_Z(\mathbf{X},\,\epsilon_1)\,\delta_Z(\mathbf{X}+L\vec{\mathbf{r}},\,\epsilon_2)\,d\mathbf{X}d\vec{\mathbf{r}}\\ &= \eta\int\limits_S\int\limits_Z \delta_Z(\mathbf{X}+L\vec{\mathbf{r}})\,d\mathbf{X}d\vec{\mathbf{r}}\end{aligned}$$

Let $\vec{\mathbf{F}}(\mathbf{X},\,\vec{\mathbf{r}},\,\epsilon)\equiv\vec{\mathbf{r}}\delta_Z(\mathbf{X}+L\mathbf{r},\vec{}\epsilon)$ is a continuous vector field.

By the Gauss–Ostrogradsky theorem:

$$\begin{aligned}\int\limits_Z \nabla\bullet\vec{\mathbf{F}}(\mathbf{X},\,\vec{\mathbf{r}},\,\epsilon)\,d\mathbf{X} &= \int\limits_{\partial Z}\vec{\mathbf{F}}(\mathbf{X},\,\vec{\mathbf{r}},\,\epsilon)\bullet\vec{N_B}(\mathbf{X})\,d\mathbf{X}\\ &= \int\limits_{\partial Z}\delta_Z(\mathbf{X}+L\mathbf{r},\vec{}\epsilon)\,\vec{\mathbf{r}}\bullet\vec{N_B}(\mathbf{X})\,d\mathbf{X}\end{aligned}$$

But we also have that:

$$\begin{aligned}\int\limits_Z \nabla\bullet\vec{\mathbf{F}}(\mathbf{X},\,\vec{\mathbf{r}},\,\epsilon)\,d\mathbf{X} &= \int\limits_Z \vec{\mathbf{r}}\bullet\nabla\delta_Z(\mathbf{X}+L\mathbf{r},\vec{}\epsilon)\,d\mathbf{X}\\ &= \int\limits_Z \nabla_{\vec{\mathbf{r}}}\delta_Z(\mathbf{X}+L\mathbf{r},\vec{}\epsilon)\,d\mathbf{X}\end{aligned}$$

Then:

$$\begin{aligned}\phi'(L) &= \lim_{\epsilon_1,\epsilon_2\to 0}\eta\int\limits_S\int\limits_M \delta_Z(\mathbf{X},\,\epsilon_1)\,\frac{\partial}{\partial L}\left[\delta_Z(\mathbf{X}+L\vec{\mathbf{r}},\,\epsilon_2)\right]d\mathbf{X}d\vec{\mathbf{r}}\\ &= \lim_{\epsilon_2\to 0}\eta\int\limits_S\int\limits_Z \nabla_{\vec{\mathbf{r}}}\delta_Z(\mathbf{X}+L\vec{\mathbf{r}},\,\epsilon_2)\,d\mathbf{X}d\vec{\mathbf{r}}\\ &= \lim_{\epsilon_2\to 0}\eta\int\limits_S\int\limits_{\partial Z} \delta_Z(\mathbf{X}+L\mathbf{r},\vec{}\epsilon_2)\,\vec{\mathbf{r}}\bullet\vec{N_B}(\mathbf{X})\,d\mathbf{X}d\vec{\mathbf{r}}\\ &= \eta\int\limits_S\int\limits_{\partial Z} \delta_Z(\mathbf{X}+L\vec{\mathbf{r}})\,\vec{\mathbf{r}}\bullet\vec{N_B}(\mathbf{X})\,d\mathbf{X}d\vec{\mathbf{r}}\end{aligned}$$

Let:

$$T\left(\mathbf{Y},L,\epsilon\right) \equiv \left\{\vec{\mathbf{x}} \mid \left|\vec{\mathbf{x}}\right| = 1,\ \mathbf{Y}+L\vec{\mathbf{x}}-\epsilon\left(\mathbf{Y}+L\vec{\mathbf{x}}-P\left(\mathbf{Y}+L\vec{\mathbf{x}}\right)\right)\in\partial Z\right\}$$

and:

$$T\left(\mathbf{Y},L\right) \equiv \left\{\vec{\mathbf{x}} \mid \left|\vec{\mathbf{x}}\right| = 1,\ \mathbf{Y}+L\vec{\mathbf{x}}\in\partial Z\right\}$$

Then:

$$\begin{aligned}
\phi''\left(L\right) &= \lim_{\epsilon_2\to 0}\eta\int\limits_{S}\int\limits_{\partial Z}\frac{\partial}{\partial L}\left[\delta_Z\left(\mathbf{X}+L\mathbf{r},\vec{\epsilon}_2\right)\right]\vec{\mathbf{r}}\bullet\vec{N_B}\left(\mathbf{X}\right)d\mathbf{X}d\vec{\mathbf{r}}\\
&= \lim_{\epsilon_2\to 0}\eta\int\limits_{\partial Z}\int\limits_{S}\nabla_{\vec{\mathbf{r}}}\delta_Z\left(\mathbf{X}+L\vec{\mathbf{r}},\ \epsilon_2\right)\vec{\mathbf{r}}\bullet\vec{N_B}\left(\mathbf{X}\right)d\vec{\mathbf{r}}d\mathbf{X}\\
&= \lim_{\epsilon_2\to 0}\eta\int\limits_{\partial Z}\int\limits_{0}^{\epsilon_2}\int\limits_{T(\mathbf{X},L,\epsilon)}-\frac{1}{\epsilon_2}\left|\mathbf{X}+L\vec{\mathbf{r}}-P\left(\mathbf{X}+L\vec{\mathbf{r}}\right)\right|\vec{\mathbf{r}}\bullet\vec{N_B}\left(\mathbf{X}\right)d\vec{\mathbf{r}}d\epsilon d\mathbf{X}\\
&= \eta\int\limits_{\partial Z}\int\limits_{T(\mathbf{X},L)}\left(-\vec{\mathbf{r}}\bullet\vec{N_B}\left(\mathbf{X}+L\vec{\mathbf{r}}\right)\right)\left(\vec{\mathbf{r}}\bullet\vec{N_B}\left(\mathbf{X}\right)\right)d\vec{\mathbf{r}}d\mathbf{X}
\end{aligned}$$

The following lemma merely proves that a stick of length zero has a configuration volume of terminals that fills the entire shape Z.

Lemma 6.6. *That* $\theta\left(0\right)=0$

Proof. By evaluation:

$$\begin{aligned}
\theta\left(0\right) &= 1-\phi\left(0\right)\\
&= 1-\eta\int\limits_{S}\int\limits_{Z}\delta_Z\left(\mathbf{X}\right)d\mathbf{X}d\vec{\mathbf{r}}\\
&= 1-\frac{1}{\left|S\right|\left|V_Z\right|}\int\limits_{S}\left|V_Z\right|d\vec{\mathbf{r}}\\
&= 1-\frac{1}{\left|S\right|\left|V_Z\right|}\left|S\right|\left|V_Z\right|\\
&= 0
\end{aligned}$$

□

The following lemma merely proves that a stick of greater than unit length has no configuration volume of terminals.

Lemma 6.7. *That* $\theta(L) = 1$ *for* $L > 1$

Proof. We note that for $L > 1$, $\phi(L) = 0$ trivially as by definition there are no lengths that can fit in Z over unit length (the maximum length), so $\theta(L) = 1 - \phi(L) = 1$. $\square$

The following lemma merely proves that as longer sticks have at best the same configuration volume of terminals than shorter sticks.

Lemma 6.8. *That* $0 < L_1 < L_2 \Rightarrow \theta(L_1) \leq \theta(L_2)$

Proof. Following on from Lemma 6.7, this is equivalent to establishing that $\forall L \in (0,1)$, $\theta'(L) \geq 0$. Given:

$$\theta'(L) \;=\; -\phi'(L) = -\eta \int\limits_{S} \int\limits_{\partial Z} \delta_Z(\mathbf{Y} + L\vec{\mathbf{r}}) \left(\vec{N_B}(\mathbf{Y}) \bullet \vec{\mathbf{r}}\right) d\mathbf{Y} d\vec{\mathbf{r}}$$

We know that as $\mathbf{Y}$ is at the boundary of Z, then by convexity of Z, all points:

$$\left\{\mathbf{Y} + L\vec{\mathbf{r}} \mid \vec{N_B}(\mathbf{Y}) \bullet \vec{\mathbf{r}} > 0\right\}$$

must lie *outside* Z. Therefore restricting $\mathbf{X} = \mathbf{Y} + \mathbf{L}\tilde{\mathbf{r}}$ to those points *inside* Z, we have:

$$\forall \mathbf{X} \in Z, \mathbf{Y} \in \partial Z : \left(\vec{N_B}(\mathbf{Y}) \bullet \overrightarrow{(\mathbf{X} - \mathbf{Y})}\right) \leq 0.$$

This similarly implies that within the integral bounds for $L > 0$, that:

$$-\delta_Z(\mathbf{Y} + L\vec{\mathbf{r}}) \left(\vec{N_B}(\mathbf{Y}) \bullet \vec{\mathbf{r}}\right) \geq 0.$$

Integration over the bounds proves for $L > 0$: $\theta'(L) \geq 0$, yielding our desired result:

$$0 < L_1 < L_2 \Rightarrow \theta(L_1) \leq \theta(L_2)$$

$\square$

The following lemma proves that the configuration volume of terminals varies linearly with the length at infinitesimal scales.

Lemma 6.9. *That* $\lim_{L \to 0^+} \frac{\theta(L)}{L} = \Upsilon$, *for some* $\Upsilon > 0$

Proof. First let us establish that $0 < \theta(0^+) < \infty$. We do this by taking the limit as L approaches zero.

$$\begin{aligned} \phi'(0) &= \lim_{L \to 0^+} \;\; \eta \int\limits_{\partial Z} \int\limits_{S} -\delta_Z(\mathbf{Y} + L\vec{\mathbf{r}}) \left(\vec{N_B}(\mathbf{Y}) \bullet \vec{\mathbf{r}}\right) d\vec{\mathbf{r}} d\mathbf{Y} \\ &= \;\; -\eta \int\limits_{\partial Z} \int\limits_{W(Y)} \left(\vec{N_B}(\mathbf{Y}) \bullet \vec{\mathbf{r}}\right) d\vec{\mathbf{r}} d\mathbf{Y} \end{aligned}$$

Where:

$$W(\mathbf{Y}) = \{\vec{\mathbf{x}} \mid N_B(\mathbf{Y}) \bullet \vec{\mathbf{x}} > 0,\ |\vec{\mathbf{x}}| = 1\}$$

This is because as L approaches zero, the delta region is positive only for vectors of $\mathbf{r}$ leading to the interior of Z.

If for all unit vectors $\vec{\mathbf{u}}$ we have:

$$\kappa = \int\limits_{\mathbf{r}\in\{\vec{x}||\vec{x}|=1,\, \vec{\mathbf{u}}\bullet\vec{x}>0\}} (\vec{\mathbf{u}} \bullet \vec{\mathbf{r}})\, d\vec{\mathbf{r}}$$

We know that $\kappa > 0$ as we are guaranteed by the non-empty integral region to only be taking a positive value.

Then $\phi'(0) = \frac{-\kappa}{|S||V_Z|}\int_{\partial Z} d\mathbf{Y} = \frac{-\kappa|\partial Z|}{|S||V_Z|}$, and $\theta'(L) = -\phi'(L)$

Thus establishing that:

$$0 < \theta'\left(0^+\right) = \frac{\kappa\,|\partial Z|}{|S|\,|V_Z|} < \infty$$

We then define $\Upsilon \equiv \frac{\kappa|\partial Z|}{|S||V_Z|} = \theta'(0^+)$. □

The following lemma proves continuity properties, namely that there aren't any discontinuities in the configuration volume of terminals, or its differential, in some finite positive region of lengths near zero.

Lemma 6.10. *That* $\exists\delta > 0,\ s.t.\ \theta(L) \in C^1(0,\delta)$

Proof. This is equivalent to proving that $|\phi''(L)| < \infty$ for some region $L \in (0,\epsilon)$. That is, the double differential is finite, and hence that $\theta'(L)$ is continuous.

$$\begin{aligned}
|\phi''(L)| &= \left|\eta\int\limits_{\partial Z}\int\limits_{T(\mathbf{Y},L)} (-N_B(\mathbf{Y}+L\vec{\mathbf{r}})\bullet\vec{\mathbf{r}})(N_B(\mathbf{Y})\bullet\vec{\mathbf{r}})\, d\vec{\mathbf{r}}d\mathbf{Y}\right| \\
&\le \eta\int\limits_{\partial Z}\int\limits_{T(\mathbf{Y},L)} |(-N_B(\mathbf{Y}+L\mathbf{r})\bullet\vec{\mathbf{r}})(N_B(\mathbf{Y})\bullet\vec{\mathbf{r}})|\, d\vec{\mathbf{r}}d\mathbf{Y} \\
&\le \eta\int\limits_{\partial Z}\int\limits_{T(\mathbf{Y},L)} d\vec{\mathbf{r}}d\mathbf{Y} \\
&= \eta\int\limits_{\partial Z} |T(\mathbf{Y},L)|\, d\mathbf{Y}
\end{aligned}$$

But $|T(\mathbf{Y},L)|$ is ordinarily the length of the $(d-2)$-dimensional curve along the boundary ∂Z that is distance L from $\mathbf{Y}$. However, in certain circumstances this can blow up to a $(d-1)$-dimensional surface. For example, consider a 3D cone, the tip of the cone is equidistant from an entire 2D surface instead of just 1D curves. As we are dealing with a restricted region of maximum length δ, we can choose a small δ such that we do not have this problem. We do this by noting that any finite $(d-1)$-dimensional sub-region of the convex boundary ∂Z that has constant curvature, that curvature must be finite, otherwise V_Z would not be convex. Let the minimum such radius of curvature in all constant-curvature segments of ∂Z be given by ρ_{min}, then we merely need to choose $0 < \delta < \rho_{min}$. We are then guaranteed that only $(d-2)$-dimensional curves are encountered at each point of $\mathbf{Y} \in \partial Z$. This thus yields a finite $\phi''(L)$ in the region $L \in (0, \delta)$. Combined with previous results (Lemmas 6.6, 6.8, 6.9), this implies that: $\exists \delta > 0,\ s.t.\, \theta(L) \in C^1(0, \delta)$. □

6.2.2 Rent's rule from continuous power-law tails

The following theorem proves that a continuous power-law tailed length distribution with exponent between 1 and 2, asymptotically follows Rent's rule with a related exponent.

Theorem 6.11. *That for a length-distribution with a continuous power-law tail of exponent* $1 < \mu < 2$*, the scaling of terminals crossing arbitrary shape* Z *in arbitrary* d*-dimensional normed vector space* M*, asymptotically grows according to Rent's rule with Rent's exponent* $p = 1 - \frac{\mu-1}{d}$.

Proof. We know that if we were to scale the shape Z by s in $\mathbb{R}^d$, it would result in the configuration space increasing by s^d:

$$T_Z(L, s) = 2V_Z s^d \theta_Z\left(\frac{L}{s}\right)$$

Then for a global probability distribution of lengths $h(L)$ we have number of terminals $T(s)$ given by:

$$\begin{aligned}
T(s) &\equiv \int_0^\infty h(L)\, T(L,s)\, dL \\
&= 2V_Z s^d \int_0^\infty h(L)\, \theta\left(\frac{L}{s}\right) dL \qquad (6.1)\\
&= 2V_Z s^d \int_0^\infty s h(ys)\, \theta(y)\, dy \\
&= 2V_Z s^{d+1} \int_0^\infty h(ys)\, \theta(y)\, dy
\end{aligned}$$

For a power-law tailed distribution $h(x) = \begin{cases} \gamma(x) & , x < L_0 \\ \lambda x^{-\mu} & , x \geq L_0 \end{cases}$

with $\int_0^{L_0} \gamma(x)\,dx = \Gamma < 1$

Note that this constraint is quite loose and admits Dirac-deltas in the head of the distribution.

$$\begin{aligned}
T(s) &= 2V_Z s^{d+1}\left(\int_0^{L_0/s} \gamma(ys)\,\theta(y)\,dy + \int_{L_0/s}^{\infty} \lambda y^{-\mu} s^{-\mu}\theta(y)\,dy\right)\\
&= 2V_Z s^{d+1-\mu}\left(s^{\mu}\int_0^{L_0/s} \gamma(ys)\,\theta(y)\,dy + \lambda\int_{L_0/s}^{\infty} y^{-\mu}\theta(y)\,dy\right)\\
&= 2V_Z s^{d+1-\mu}\left(P(s,L_0) + \lambda Q\left(\frac{L_0}{s}\right)\right)
\end{aligned}$$

Where:

$$\begin{aligned}
P(s,L_0) \equiv s^{\mu}\int_0^{L_0/s} \gamma(ys)\,\theta(y)\,dy &\le s^{\mu}\int_0^{L_0/s} \gamma(ys)\,\theta\left(\frac{L_0}{s}\right)dy\\
&= s^{\mu-1}\theta\left(\frac{L_0}{s}\right)\int_0^{L_0} \gamma(x)\,dx\\
&= s^{\mu-1}\theta\left(\frac{L_0}{s}\right)\Gamma
\end{aligned}$$

Using Lemma 6.8 (positive monotonicity of $\theta(y)$) in the first inequality step.

And we also define:

$$Q(x) \equiv \int_x^{\infty} y^{-\mu}\theta(y)\,dy$$

We know that $\lim_{x\to 0^+}\frac{\theta(x)}{x} = const$. Therefore:

$$\lim_{s\to\infty}\left|\frac{P(s,L_0)}{s^{\mu-2}}\right| \le \lim_{s\to\infty}\left|\frac{\theta\left(\frac{L_0}{s}\right)}{\left(\frac{L_0}{s}\right)}L_0\Gamma\right| = \theta'(0)\,L_0\Gamma = const \tag{6.2}$$

But then for $\mu < 2$, we have $\lim_{s\to\infty} P(s,L_0) = 0$

By Lemma 6.10, we know that $\theta(y)$ is $C^1(0,\delta)$ for some $\delta > 0$, and by Lemma 6.9, we also know that $lim_{y\to 0}\frac{\theta(y)}{y} = \Upsilon$.

Let us denote $E(y) = \theta'(y) - \Upsilon$. So $\forall\epsilon > 0$, $\exists\delta' < \delta$ s.t. $\forall y \in (0,\delta') : |E(y)| < \epsilon$.

Now:

$$\begin{aligned} Q(0) - Q(\delta') &= \int_0^{\delta'} y^{-\mu}\theta(y)\,dy \\ &= \left.\frac{y^{1-\mu}}{1-\mu}\theta(y)\right|_0^{\delta'} - \int_0^{\delta'} \frac{y^{1-\mu}}{1-\mu}\theta'(y)\,dy \\ &= \frac{\delta'^{1-\mu}}{1-\mu}\theta(\delta') + \int_0^{\delta'} \frac{y^{1-\mu}}{\mu-1}(\Upsilon + E(y))\,dy \end{aligned}$$

Now each of these terms is finite provided $1 < \mu < 2$, resulting in $Q(0) - Q(\delta')$ being finite. As $E(y) < \epsilon$:

$$\begin{aligned} |Q(0) - Q(\delta')| &= \left|\frac{\delta'^{1-\mu}}{1-\mu}\theta(\delta') + \int_0^{\delta'} \frac{y^{1-\mu}}{\mu-1}(\Upsilon + E(y))\,dy\right| \\ &\leq \left|\frac{\delta'^{1-\mu}}{1-\mu}\theta(\delta')\right| + \left|\int_0^{\delta'} \frac{y^{1-\mu}}{\mu-1}(\Upsilon + E(y))\,dy\right| \\ &\leq \left|\frac{\delta'^{1-\mu}}{1-\mu}\theta(\delta')\right| + \left|\int_0^{\delta'} \frac{y^{1-\mu}}{\mu-1}(\Upsilon + \epsilon)\,dy\right| \\ &= \left|\frac{\delta'^{2-\mu}}{1-\mu}\left(\frac{\theta(\delta')}{\delta'}\right)\right| + \left|(\Upsilon+\epsilon)\frac{\delta'^{2-\mu}}{(\mu-1)(2-\mu)}\right| \end{aligned}$$

Noting that $\lim_{\delta\to 0}\frac{\theta(\delta)}{\delta} = \Upsilon$, then each of these terms converges to zero:

$$\lim_{\delta\to 0}|Q(0) - Q(\delta)| = 0$$

Thus for:

$$T(s) = 2V_Z s^{d+1-\mu}\left(P(s,\, L_0) + \lambda Q\left(\frac{L_0}{s}\right)\right)$$

For $L_0 \ll s$, we have $P(s,\, L_0) \to 0$, and $Q\left(\frac{L_0}{s}\right) \to Q(0)$ which is a constant we denote ζ:

$$\begin{aligned} T(s) &= 2V_Z\zeta s^{d+1-\mu}\left(\lambda + O\left(\left(\frac{L_0}{s}\right)^{2-\mu}\right)\right) \\ &\approx 2V_Z\zeta\lambda s^{d+1-\mu} \end{aligned}$$

Where $O\left(\left(\frac{L_0}{s}\right)^{2-\mu}\right)$ is the order of the error term.

Or asymptotically:

$$\lim_{s\to\infty}\frac{T(s)}{s^{d+1-\mu}}=2V_Z\zeta\lambda$$

This is the same form as Rent's Rule. Let us compare with the usual parameters of Rent's Rule:

$$T=kN^p=k\left(V_Zs^d\right)^p$$

Now equating exponents:

$$\begin{aligned}p.d &= d+1-\mu\\ \mu &= (1-p)\,d+1\\ p &= 1-\frac{\mu-1}{d}\end{aligned}\qquad(6.3)$$

and constant:

$$k=2\zeta\lambda V_Z^{1-p}$$

As the scaling derivation is valid for $1<\mu<2$, these relationships are likewise only valid for:

$$1-\frac{1}{d}<p<1$$

□

6.2.3 Rent's rule from discrete power-law tails

The following theorem proves that a discrete power-law tailed length distribution with exponent between 1 and 2, asymptotically follows Rent's rule with a related exponent.

Theorem 6.12. *That for a length-distribution with a discrete power-law tail of exponent $1<\mu<2$, the scaling of terminals crossing arbitrary convex shape Z in arbitrary d-dimensional normed vector space M, asymptotically grows according to Rent's rule with Rent's exponent $p=1-\frac{\mu-1}{d}$.*

Proof. Here we have $h(x)=\begin{cases}\gamma(x) & ,\ x<L_0\\ \sum_k\lambda x^{-\mu}\delta(x-\varphi k) & ,\ x\geq L_0\end{cases}$

For some discrete distribution spacing φ.

We have:

$$T(s)=2V_Zs^d\int\limits_0^\infty h(L)\,\theta\left(\frac{L}{s}\right)dL$$

We know by definition of Dirac-delta that:

$$\int_{\left(k-\frac{1}{2}\right)\varphi}^{\left(k+\frac{1}{2}\right)\varphi} f\left(x\right)\delta\left(x-\varphi k\right)g\left(x\right)dx=\frac{1}{\varphi}f\left(\varphi k\right)g\left(\varphi k\right)$$

Suppose $\forall x,\ f'\left(x\right)\leq 0$, and $g'\left(x\right)\geq 0$, then let:

$$f_{max}\equiv\sup_{x\in\left(\varphi\left(k-\frac{1}{2}\right),\varphi\left(k+\frac{1}{2}\right)\right)}\left\{f\left(x\right)\right\}=f\left(\varphi\left(k-\frac{1}{2}\right)\right)$$

$$g_{max}\equiv\sup_{x\in\left(\varphi\left(k-\frac{1}{2}\right),\varphi\left(k+\frac{1}{2}\right)\right)}\left\{g\left(x\right)\right\}=g\left(\varphi\left(k+\frac{1}{2}\right)\right)$$

This allows us to remove the Dirac-deltas with a simpler integral bound:

$$\begin{aligned}\int_{\left(k-\frac{1}{2}\right)\varphi}^{\left(k+\frac{1}{2}\right)\varphi} f\left(x\right)\delta\left(x-\varphi k\right)g\left(x\right)dx &\leq \int_{\left(k-\frac{1}{2}\right)\varphi}^{\left(k+\frac{1}{2}\right)\varphi} f_{max}\delta\left(x-\varphi k\right)g_{max}dx\\ &= f_{max}g_{max}\\ &= \frac{1}{\varphi}\int_{\left(k-\frac{1}{2}\right)\varphi}^{\left(k+\frac{1}{2}\right)\varphi} f_{max}g_{max}dx\\ &\leq \frac{1}{\varphi}\int_{\left(k-\frac{1}{2}\right)\varphi}^{\left(k+\frac{1}{2}\right)\varphi} f\left(x-\varphi\right)g\left(x+\varphi\right)dx\end{aligned}$$

In particular for $f\left(x\right)=x^{-\mu}$ and $g\left(x\right)=\theta\left(\frac{x}{s}\right)$, over the entire tail of the integral we then have:

$$\begin{aligned}W\left(s,L_0\right) &\equiv \frac{1}{s}\int_{L_0}^{\infty}h\left(x\right)\theta\left(\frac{x}{s}\right)dx\\ &= \frac{1}{s}\int_{L_0}^{\infty}\sum_{k}\lambda x^{-\mu}\delta\left(x-\varphi k\right)\theta\left(\frac{x}{s}\right)dx\\ &\leq \frac{1}{s}\int_{L_0}^{\infty}\lambda\left(x-\varphi\right)^{-\mu}\theta\left(\frac{x+\varphi}{s}\right)dx\end{aligned}$$

Let $ys=x-\varphi$, then:

$$\begin{aligned} W(s, L_0) &\leq \int\limits_{(L_0-\varphi)/s}^{\infty} \lambda (ys)^{-\mu} \theta\left(y + \frac{2\varphi}{s}\right) dy \\ &= s^{-\mu}\lambda \int\limits_{(L_0-\varphi)/s}^{\infty} y^{-\mu}\theta\left(y + \frac{2\varphi}{s}\right) dy \end{aligned}$$

We can also similarly say, using the minimum values in the range, that:

$$W(s, L_0) \geq s^{-\mu}\lambda \int\limits_{(L_0+\varphi)/s}^{\infty} y^{-\mu}\theta\left(y - \frac{2\varphi}{s}\right) dy$$

So that:

$$\int\limits_{(L_0+\varphi)/s}^{\infty} y^{-\mu}\theta\left(y - \frac{2\varphi}{s}\right) dy \leq \frac{s^{\mu}}{\lambda} W(s, L_0) \leq \int\limits_{(L_0-\varphi)/s}^{\infty} y^{-\mu}\theta\left(y + \frac{2\varphi}{s}\right) dy \qquad (6.4)$$

Let

$$Q(s, L_0) \equiv \frac{s^{\mu}}{\lambda} W(s, L_0)$$

Then we have:

$$\begin{aligned} T(s) &= 2V_Z s^{d+1} \left(\int\limits_{0}^{L_0/s} \gamma(ys)\theta(y)\,dy + W(s, L_0) \right) \\ &= 2V_Z s^{d+1-\mu} \left(P(s, L_0) + \lambda Q(s, L_0) \right) \end{aligned}$$

We can treat $P(s, L_0)$, the terminal contribution from the head distribution γ in an identical way to the continuous version in theorem 6.11. So for $\mu < 2$, we have $\lim_{s\to\infty} P(s, L_0) = 0$

We have from eqn 6.4:

$$\int\limits_{(L_0+\varphi)/s}^{\infty} y^{-\mu}\theta\left(y - \frac{2\varphi}{s}\right) dy \leq Q(s, L_0) \leq \int\limits_{(L_0-\varphi)/s}^{\infty} y^{-\mu}\theta\left(y + \frac{2\varphi}{s}\right) dy \qquad (6.5)$$

We want to know the behaviour of $\lim_{s\to\infty} Q(s, L_0)$. To do this, let us generalise the form above.

For $x \geq 2\varphi/s$ we know by positive monotonicity of $\theta(x)$ that:

$$\theta\left(y - \frac{2\varphi}{s_1(x)}\right) \leq \theta(x) \leq \theta\left(y + \frac{2\varphi}{s_2(x)}\right)$$

For any $s_1(x) > 0$ and $s_2(x) > 0$. Thus:

$$\int\limits_{x}^{\infty} y^{-\mu}\theta\left(y - \frac{2\varphi}{s_1(x)}\right) dy \leq \int\limits_{x}^{\infty} y^{-\mu}\theta(x)\,dy \leq \int\limits_{x}^{\infty} y^{-\mu}\theta\left(y + \frac{2\varphi}{s_2(x)}\right) dy$$

Equating bounds with eqn 6.5 we have:

$$\begin{aligned}\frac{(L_0+\varphi)}{s_1(x)} &= x \\ \therefore s_1(x) &= \frac{(L_0+\varphi)}{x}\end{aligned}$$

Similarly:

$$s_2(x) = \frac{(L_0-\varphi)}{x}$$

This means that:

$$\lim_{s\to\infty} \int\limits_{(L_0+\varphi)/s}^{\infty} y^{-\mu}\theta\left(y-\frac{2\varphi}{s}\right)dy = \lim_{x\to 0^+}\int\limits_{x}^{\infty} y^{-\mu}\theta\left(y-\frac{2\varphi}{L_0+\varphi}x\right)dy$$

and

$$\lim_{s\to\infty} \int\limits_{(L_0-\varphi)/s}^{\infty} y^{-\mu}\theta\left(y+\frac{2\varphi}{s}\right)dy = \lim_{x\to 0^+}\int\limits_{x}^{\infty} y^{-\mu}\theta\left(y+\frac{2\varphi}{L_0-\varphi}x\right)dy$$

Now, we want to show that: $\lim_{x\to 0^+}\left|\int_x^\infty y^{-\mu}\theta\left(y-\frac{2\varphi}{s_1(x)}\right)dy - \int_x^\infty y^{-\mu}\theta(x)\,dy\right| = 0$

And that: $\lim_{x\to 0^+}\left|\int_x^\infty y^{-\mu}\theta\left(y+\frac{2\varphi}{s_2(x)}\right)dy - \int_x^\infty y^{-\mu}\theta(x)\,dy\right| = 0$

These limits are two particular cases $\alpha = -\frac{2\varphi}{L_0+\varphi}$ and $\alpha = \frac{2\varphi}{L_0-\varphi}$ of the general relationship, which we define:

$$\begin{aligned}
\Delta(\alpha,\mu) &\equiv \left|\lim_{x\to0^+}\int_x^\infty y^{-\mu}\left(\theta(y+\alpha x)-\theta(y)\right)dy\right| \\
&= \left|\lim_{x\to0^+}\int_0^\infty y^{-\mu}\left(\theta(y+\alpha x)-\theta(y)\right)dy-\lim_{x\to0^+}\int_0^x y^{-\mu}\left(\theta(y+\alpha x)-\theta(y)\right)dy\right| \\
&\le \left|\int_0^\infty y^{-\mu}\left(\lim_{x\to0^+}\left(\frac{\theta(y+\alpha x)-\theta(y)}{x}\right)x\right)dy\right| \\
&\quad+\left|\lim_{x\to0^+}\int_0^x y^{-\mu}\left(\theta(y+\alpha x)-\theta(y)\right)dy\right| \\
&= \left|\int_0^\infty y^{-\mu}\left(\alpha\lim_{x\to0^+}\theta'(y)\,x\right)dy\right|+\left|\lim_{x\to0^+}\int_0^x y^{-\mu}\left(\theta(y+\alpha x)-\theta(y)\right)dy\right| \\
&= 0+\left|\lim_{x\to0^+}\int_0^x y^{-\mu}\left(\theta(y+\alpha x)-\theta(y)\right)dy\right| \\
&\le \left|\lim_{x\to0^+}\int_0^x y^{-\mu}\left(\alpha\sup_{y\in(0,x)}\{\theta'(y)\}\,x\right)dy\right| \\
&= \left|\lim_{x\to0^+}\alpha\frac{x^{1-\mu}}{\mu-1}x\sup_{y\in(0,x)}\{\theta'(y)\}\right| \\
&\le \left|\alpha\lim_{x\to0^+}\frac{x^{2-\mu}}{\mu-1}\sup_{y\in(0,\delta)}\{\theta'(y)\}\right|,\ \forall\delta>x
\end{aligned}$$

For $1<\mu<2$: this has limit 0 as we know by Lemma 6.10 that $\theta'(y)$ is continuous and finite in some region $(0,\delta)$.

$$\begin{aligned}
\therefore\lim_{s\to\infty}Q(s,L_0) &= \int_0^\infty y^{-\mu}\theta(y)\,dy \\
&= Q(0)
\end{aligned}$$

Again, for $L_0\ll s$, we have $P(s,\,L_0)\to0$, and $Q(s,\,L_0)\to Q(0)$ which is a constant we denote ζ:

$$\begin{aligned}
T(s) &= 2V_Z\zeta s^{d+1-\mu}\left(\lambda+O\left(\left(\frac{L_0}{s}\right)^{2-\mu}\right)\right) \\
&\approx 2V_Z\zeta\lambda s^{d+1-\mu}
\end{aligned}$$

Where $O\left(\left(\frac{L_0}{s}\right)^{2-\mu}\right)$ is the order of the error term.

Or asymptotically:

$$\lim_{s\to\infty}\frac{T(s)}{s^{d+1-\mu}}=2V_Z\zeta\lambda$$

Thus provided $L_0>\varphi$, the discrete version is also finite and converges to the same constant factor as the continuous solution's. In general, we can extend the head distribution to be the same as the tail distribution, which allows us to choose L_0 arbitrarily, so this constraint is unimportant. □

Note that this discrete length distribution is not the same as saying there is a discrete grid of points that edges must be 'snapped' to. Continuous placement and orientations are allowed for the discrete lengths in this formalism. However it is also possible to show that the discrete grid version converges to the continuous case in the limit of very large scales, by showing that the discrete version of $\theta(x,s)$ converges to the continuous version at large scales.

6.2.4 Rent's rule from finite distribution lengths

The following theorem proves that a finite, discrete length distribution also follows Rent's rule but only with a special, fixed Rent's exponent.

Theorem 6.13. *That for a finite-length discrete distribution (where largest length is at most $L_{max}<\infty$), the scaling of terminals crossing arbitrary convex shape Z in arbitrary d-dimensional normed vector space M, asymptotically grows according to Rent's rule with Rent's exponent $p=1-\frac{1}{d}$.*

Proof. We start with $h(x)=\delta(L_1-x)$, where δ is the Dirac delta function and $L_1\le L_{max}$. Then:

$$\begin{aligned}T(s) &= 2V_Zs^d\int\limits_{L_0}^{\infty}\delta(L_1-L)\,\theta\left(\frac{L}{s}\right)dL\\ &= 2V_Zs^d\theta\left(\frac{L_1}{s}\right)\end{aligned}$$

In particular for $s\gg L_1$

$$\begin{aligned}T(s) &\approxeq 2V_Zs^d\Upsilon\frac{L_1}{s}\\ &= 2V_Zs^{d-1}\Upsilon L_1\end{aligned}$$

Or rather:

$$\lim_{s\to\infty}\frac{T(s)}{s^{d-1}}=2V_Z\Upsilon L_1$$

Thus equating to Rent's equation:

$$\begin{aligned} pd &= d-1 \\ p &= 1-\frac{1}{d} \end{aligned}$$

Which is the expected Rent's exponent for a d-dimensional network embedded in d dimensions, with fixed interconnection lengths. By linearity of T w.r.t h, any finite combination of fixed lengths produces a finite sum of these integrals, and thus also produces this limiting case Rent's exponent. □

Looking again at equation 6.3, we should note here that there is a phase-transition that occurs for power-law tailed distributions with exponent μ:

$$p = 1 - \frac{\mu - 1}{d}.$$

As the power-law exponent μ approaches the limit of 2 for the power-law domain, the Rent's exponent converges to $1-\frac{1}{d}$, which is the case of a finite length distribution. Indeed it can be shown that for $\mu > 2$ that the Rent's exponent still converges to a limiting case of $1-\frac{1}{d}$, which is an attribute actually observed in VLSI designs [109].

6.2.5 Uniqueness of exact solutions

The following theorem shows that for a particular (exact) terminal distribution function, there is at most one exact distance distribution that generates it.

We have from eqn 6.1:

$$T(s) \equiv 2V_Z s^d \int_0^\infty h(L)\,\theta\left(\frac{L}{s}\right) dL$$

Theorem 6.14. *For any given $T(s)$ there is at most one exact distance distribution solution $h(L)$.*

Proof. For a given $T(s)$, let us suppose there exist two distance distribution solutions: $h_1(x)$, $h_2(x)$. where $h_1 \neq h_2$. Then there exists a non-zero function:

$$g(x) = h_1(x) - h_2(x)$$

If g is not zero, then it must have a region where it is always positive or always negative. Let the very first domain be (a, b) from the y-axis. If b is $+\infty$ then we choose another value for b where $a < b < \infty$.

If g satisfies:

$$\forall s > 0 : \int_0^\infty g(L)\,\theta\left(\frac{L}{s}\right) dL = 0 \tag{6.6}$$

Then $-g$ also satisfies this, so without loss of generality we can assume that g is positive in the domain (a, b):

$$\forall x \in (a, b): \; g(x) \geq 0$$

and

$$\int_a^b g(x)\, dx > 0$$

This constraint allows us to also handle Dirac-deltas inside the region.

As it is the first domain we can extend that:

$$\forall x \in (0, a): \; g(x) = 0$$

Now $\theta(y)$ is a strictly monotonically increasing function for $y \in (0, 1)$, and $\forall y > 1 :$ $f(y) = 1$

Let $\eta(y) = \theta(2y) - \theta(y)$.

Then:

$$\eta(0) = 0$$

$$\forall y \in (0, 1): \; \eta(y) > 0$$

and

$$\forall y \geq 1: \; \eta(y) = 0$$

This is because $\forall y \in (0, 1), \delta > 0: \; \theta(y) < \theta(y + \delta)$

If eqn 6.6 is true for all positive s, then it is true for $s_1 = \frac{2}{b}$ and $s_2 = \frac{1}{b}$, and:

$$\int_0^\infty g(L)\, \theta\left(\frac{L}{s_1}\right) dL - \int_0^\infty g(L)\, \theta\left(\frac{L}{s_2}\right) dL = 0 - 0 = 0$$

$$\begin{aligned} \Rightarrow \int_0^\infty g(L)\, \eta\left(\frac{L}{b}\right) dL &= 0 \\ \Rightarrow \int_a^b g(L)\, \eta\left(\frac{L}{b}\right) dL &= 0 \end{aligned}$$

But we know $\eta\left(\frac{L}{b}\right)$ is strictly positive in the region (a, b) and $g(L)$ is also positive, which results in a non-zero integral, thus is a contradiction. $\square$

We must be careful about interpreting what this means, as this uniqueness-property is for exact $T(s)$. In general, for *asymptotic* Rentian behaviour of fixed Rentian parameters, there are many possible specific $T(s)$ and corresponding $h(L)$ with specific head and tail distributions. However, it is the tail of the length distribution that is most important for determining Rentian behaviour, and it is also the tail of the length distribution that dominates in cost (for a linearly or super-linearly growing cost function). This means that the minimum power-law tail exponent is favoured as the size of the system grows, dominating over any deviation due to the head distribution.

6.2.6 Internal Edge Length Distribution

Given a global edge distribution, for a given region size we can determine the expected number and distribution of internal edge lengths. This is merely:

$$I(L) = V_Z s^d \phi\left(\frac{L}{s}\right) h(L)$$

Thus there is a cut-off according to $\phi\left(\frac{L}{s}\right)$ of the power-law tail comprising $h(L)$. For simple shapes, such as boxes, this cut-off is polynomial. Regardless of the shape, it is also strict at $L = s$ as $\phi(1) = 0$. In Network Science, exponential cut-offs in the power-law distribution are commonly 'reported' [30], however this is certainly not the behaviour here.

6.3 Fractal embeddings into arbitrary spaces

The problem with the classical Rent's Rule is that it deals with a homogeneous distribution of resources and costs. We suggest that there is something more fundamental than Rent's Rule. That Rent's rule is actually a homogeneous instance of a more general law governing embeddings of fractal dimensional graphs. Our proof of the equivalence of power-law tailed length distributions and Rent's rule in a homogeneous system leads us to consider that the particulars of the graph being embedded do not matter for Rentian behaviour to emerge – instead any graphs stochastically chosen from the same power-law tailed length distribution should exhibit similar Rentian behaviour.

6.3.1 Invariant property of embeddings

The following lemma shows that length distribution optimisation in a d-dimensional space, is equivalent to optimisation in a configuration-space, subject to maintaining the graph property constraints.

Lemma 6.15. *Finding the optimal length distribution in a continuous d-dimensional space is equivalent to finding an optimal configuration-space distribution with translated cost-function $c_V(V) = c\left(aV^{1/d}\right)$ for some constant a.*

Proof. The optimal length distribution $f(l)$ is such that the total cost C is minimised, under cost function $c(l)$, subject to maintaining the graph properties:

$$C = \int_0^\infty f(l)\, c(l)\, dl$$

Instead of expressing by length, we can simply express these in terms of the volume of d-dimensional space reachable:

$$V = al^d,$$

for some constant a. We identify this to also be the configuration volume of possible links reachable within cost $c(l)$ from *each* node.

By simple change of variable we have:

$$\begin{aligned} C &= \int_0^\infty f(l)\, c(l)\, dl \\ &= \int_0^\infty f(l(V))\, c(l(V))\, \frac{\partial l}{\partial V} . dV \\ &= \int_0^\infty f\left(aV^{1/d}\right) c\left(aV^{1/d}\right) \frac{a}{d} V^{1/d-1} dV \\ &= \int_0^\infty \left(\frac{a}{d} V^{1/d-1} f\left(aV^{1/d}\right)\right) c\left(aV^{1/d}\right) dV \end{aligned}$$

If we define the translated cost function to be: $c_V(V) \equiv c\left(aV^{1/d}\right)$

Then the equivalent minimisation problem is to find a distribution $f_V(V)$ subject to graph property constraints, that minimises:

$$C = \int_0^\infty f_V(V)\, c_V(V)\, dV$$

□

Lemma 6.16. *That an embedding in d-dimensional space with a power-law-tailed length distribution of exponent $\mu = (1-p)d + 1$ occurs iff there is also a power-law tailed distribution in the configuration-space for each node, with exponent $\mu_V = 2 - p$, that is independent of the dimensionality d.*

Proof. For a d-dimensional space with optimal continuous distance distribution under cost function $c(l)$ of:

$$f(l) = \begin{cases} \lambda l^{-\mu} & , l \geq l_{tail} \\ \gamma(l) & , l < l_{tail} \end{cases}$$

From Lemma 6.15, the optimisation problem is equivalent to optimising in configuration-space under translated cost function $c_V(V) = c\left(aV^{1/d}\right)$, with solution:

$$f_V(V) = \frac{a}{d} V^{1/d-1} f\left(aV^{1/d}\right)$$

then we have:

$$f_V(V) = \begin{cases} \alpha V^{1/d-1-\mu/d} & , V \geq a l_{tail}^d \\ \frac{a}{d} V^{1/d-1} \gamma\left(aV^{1/d}\right) & , V < a l_{tail}^d \end{cases}$$

for some constant α. Thus $f_V(V)$ has a negative power-law tail exponent μ_V of:

$$\begin{aligned} \mu_V &= 1 + \frac{\mu - 1}{d} \\ &= 2 - p \end{aligned}$$

This is independent of the original embedding dimension d. We can similarly show for the discrete power-law distribution. □

Among other things, this also means we can take a power-law tailed embedding in a d_o dimensional space and exponent μ_o, and we can translate the embedding into a $d_e < d_o$ dimensional space, with appropriate cost-function into another power-law tailed distribution with exponent:

$$\mu_e = \frac{d_e}{d_o}(\mu_o - 1) + 1$$

What this means is that Rentian behaviour, more fundamentally, can be described as a power-law tail in a configuration-space $f_V(V)$, and optimal embeddings into a particular space are equivalent to translating the distribution according to the minimal-cost-mapping of volume to lengths in the embedded space. This also implies that $p_e = p_o$, that is the Rent's exponent under an optimal remapping from one embedding space to another, is unchanged. If, as per prior work [99], we express p according to some intrinsic dimensionality f_D:

$$p = 1 - \frac{1}{f_D}$$

Then in the tail for some λ_V:

$$\begin{aligned} f_V(V) &= \lambda_V V^{p-2} \\ &= \lambda_V V^{-\frac{1}{f_D}-1} \end{aligned} \tag{6.7}$$

This distribution, with a cost function, is invariant under embedding. Indeed we can generalise by assuming it is invariant under any embedding space (including ones with heterogeneous costs and edge distributions). We also note that for the discrete power-law distribution, we do not need a separate head-distribution – we can just use the power-law distribution to describe a system uniformly at all scales with Rentian property, noting that the set of complete discrete power-law distributions is also 'cost-universal'. There are two slightly different generalisations – a local Rentian law, or a global Rentian law. A local Rentian law has the power-law tail surrounding each node separately, acting on the configuration space of connections to that node, so that each node has an equal distribution of connectivity. A global Rentian law operates globally, on all connections at once. It acts on the configuration space of every node to every other node, and in inhomogeneous systems can result in remote nodes having fewer connections than average. By acting on the $N \times V_{local}$ configuration spaces, the configuration space volume V_{global} is expanded by the number of nodes in the system, and needs to be normalised back by contracting by N – the number of nodes in the system. Thus the normalised configuration volume $V_{norm} = V_{global}/N$. The local Rentian law needs to be used carefully as it can give globally contradictory results. This is because node A may assign a connection probability to node B that is not the same as node B's connection probability to node A. If connections are directional, this doesn't necessarily result in a contradiction, as one direction may have a different probability to the other direction. However, it is hard to see how real embeddings operate like this if direction is unimportant. The global Rentian law, however, can lead to boundary or remote edges with fewer connections – which intuitively one could see happening as nodes with lower average connectivity get pushed out to make way for better locality for the rest of the nodes in the middle.

We can also extend the cost model to include the originating node, so that at cost zero, the only node reachable is itself. This allows us to avoid the singularity at the head of the distribution by always guaranteeing that the initial volume starts at one. A simple continuous power-law distribution can be used thereafter at all scales.

6.4 Conclusion

This chapter has proven some new fundamental results concerning Rent's rule. Previously the analytical tools for Rent's rule had been restricted to the discrete 2-D (or 3-D) lattice of VLSI with rectangular bounding region. In this chapter, it has been expanded to continuous d-dimensional vector spaces with arbitrary metric and arbitrary convex bounding shape and shown to be asymptotically equivalent to a power-law tailed distance distribution (whether continuous or discrete). Furthermore, the problem of optimally embedding a graph in a particular d-dimensional space, under a particular cost-function has been shown to be equivalent to embeddings in a dimensionless configuration-space, with an equivalent cost-function. Due to cost-universality over the family of power-law distributions, the asymptotic power-law tailed exponent in configuration-space is preserved independent of the particulars of the embedding space. By preserving this invariant, it allows one to generalise Rent's rule to new embedding spaces – including inhomogeneous ones and, importantly, the CMP domain where the time dimension extends in only one direction and is infinite, whilst the spatial dimensions are finite. Indeed, this is utilised in Chapter 7 for further analysis.

6.4.1 An intuitive example: human-interaction networks

To illustrate just how general and how far from the VLSI domain we can now take the Rentian model, let us look at an intuitive example – modelling the human interaction network in physical space. In this case, the interaction cost might be the total time spent (including the time spent earning the monetary cost) of going from one point to another. These costs are inhomogeneous with distance as there is walking, car, rail, and air transportation. We note that the distribution of people is inhomogeneous – there can be large rural regions with low population density, as well as high density city centres. Thus the volume of links between nodes (people) reachable within a given cost would likely be very inhomogeneous. Like the CMP scenario, the Rentian embedding should also be spatio-temporal, describing links both in space and in time. This would allow the expected frequency of contact between people to also be captured. For an approximately homogeneous population distribution we would then expect an approximately power-law distance distribution and approximately power-law temporal distribution in interaction times. Indeed, such spatial behaviour is observed by Brockmann et al. [17] In their study, they examined data for US bank-notes that were voluntarily logged by recipients and found power-law scaling of $\mu \approx 1.6$ spatially (in 2-D), and they suggest power-law scaling in the underlying inter-wait times as well. The asymptotic equivalence of Rent's rule and power-law distance distributions in a *homogeneous* space, that has been shown in this chapter, certainly suggests that human travel may at least *spatially* be Rentian, however it would be even more interesting to see if it is actually Rentian in a *spatio-temporal* sense.

Although Brockmann et al. produced a fitting Lévy-flight model (power-law distributed random walk) that assumes homogeneity, the generalised Rentian model could then produce far more detailed predictions by adjusting for population inhomogeneities, as well as by earning power – as that skews the cost calculation. The model may even be able to predict the effect of lowering or raising, for example, air transportation costs on the quantity of air travel, or of adding new routes. It would be very interesting to see if the human interaction network is actually Rentian in nature, and whether such predictions would be accurate.

Chapter Seven

Spatio-temporal Rentian model: CMP scaling implications

7.1 The many-core memory bandwidth wall

Assuming software in a CMP follows a spatio-temporal Rent's rule, this chapter examines the scaling implications and tradeoffs for future CMP in terms of their external I/O bandwidth, on-chip memory, on-chip communication, coarseness of inter-tile communication, and sensitivity (or lack thereof) to NoC topology.

Chapter 2 examined some of the motivations for needing locality in a CMP setting, including that parallel execution that exploits data *independence,* results in scaling problems for external I/O bandwidth. So if data *independence* is a problem, can data *inter*-dependence and its locality save the day? In the following subsections, an ideal Rentian many-core system that fully exploits the available locality is analytically examined. Perhaps surprisingly, although the scaling is considerably better, the memory-wall is still a crippling problem.

7.1.1 The spatio-temporal Rentian model

The Generalised Rentian results from Chapter 6 show that Rent's rule is asymptotically equivalent to a power-law tailed distribution in a cost-ordered configuration-space. We can choose our configuration space, then, to see how Rent's rule generally affects CMP architectures. Computation on a CMP or superscalar processor consists of both spatial as well as temporal communication costs. There is a cost for moving data in time. Physically, this comprises moving the data into and out of local storage (such as a register file) as well as the power consumption taken up by the added memory. The quantity of memory needed also increases linearly with the length of the temporal interconnect. Memory is also expensive in terms of basic manufacturing cost for the area it takes up on-chip, thus the temporal interconnect necessarily incurs an area cost as well. Additionally,

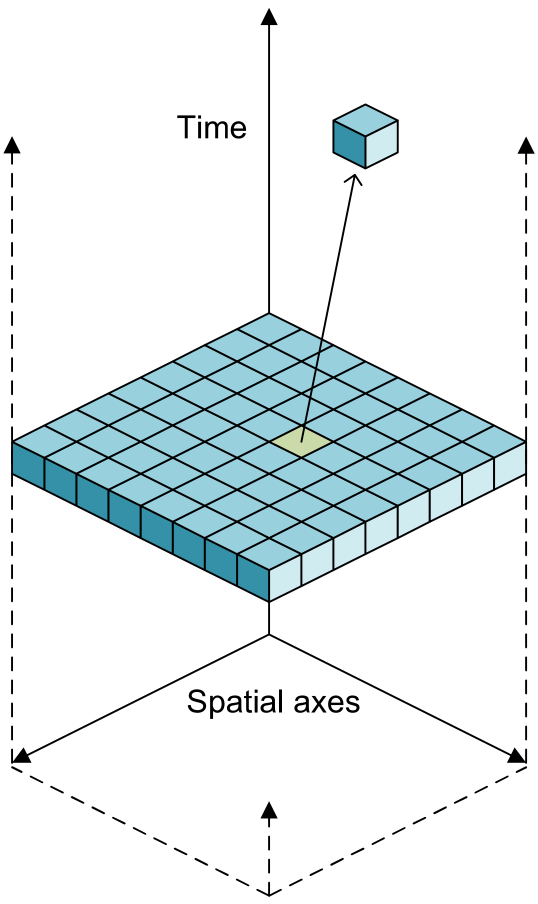


Figure 7.1: Simple Spatio-Temporal view of a 64-core tiled CMP. Information flows both spatially and temporally. The cores for one moment in time are shown here. One of these cores (green) has a communication link to a future instruction on another core.

off-chip memory accounts for a significant proportion of embedded system power cost [59] including for memory refresh.

For our purposes we will use the full power-law distribution starting at a volume of one (to represent the configuration volume of the originating tile), as it is self-similar at each scale of the hierarchy, making the analysis simpler. The asymptotic behaviour here is not affected by having alternate head distributions, instead of the complete power-law.

In figure 7.1 is an illustration of the spatio-temporal model. Here, the 64-cores (8x8) form a 3-D vertical column extending up in time, in which instructions are embedded. However, unlike placement in 2-D VLSI, instructions can only communicate with future instructions, and are constrained in how quickly they can communicate with their neighbours. In this illustration, we see an instruction embedded in one core (green) communicating with an instruction on another core at a future time-point. In this model, we are not concerned with whether this communication is explicit (e.g. message-passing based) or implicit (e.g. shared memory based). At the time of the data being produced (green core), it may not even know exactly when or where it will be consumed.

Let the cost of a unit of temporal communication be c_T and for spatial communication be c_S. Depending on the speed of spatial communication, it may take many temporal time steps before an allowable spatial step. Assuming that the communication statistics are stationary (independent of time), at least for the duration of the phase of execution being considered, we can examine just a single temporal layer of communication (seen as a 2D cross-section in figure 7.1). For this layer, there is a corresponding configuration volume of allowed edges with associated costs. We need to normalise this configuration volume to be per-node. For example, there are 64 possible links that extend exactly one unit in the temporal dimension but no units in the spatial dimension. So this is a configuration volume of 64 links with cost c_T, and a normalised configuration volume of one. Similarly there is a configuration volume of 64-links for communication exactly T steps in time, and no steps in space, with cost Tc_T. For side-length $S = 8$, there are also $4S\,(S-1) = 224$ links that extend exactly one spatial unit and, say, five temporal units with cost $5c_T + c_S$, and this has a normalised volume of $224/64 = 7/2$. Note that there may be many possible paths of equal cost for each of these spatio-temporal links, but the actual paths are abstracted away for the Rentian analysis – what matters here is whether or not a link is allowed, and how much it costs. In this fashion we can construct volumes of allowable links, and order the configuration space according to increasing cost. At smaller cost scales, the *accessible* normalised configuration volume $V\,(c)$, which is the total normalised volume with cost less than or equal to c, may grow according to cost c, in a 3-D fashion i.e. $V \sim \Theta\,(c^3)$, because for increasing cost, there is a reachable volume of links growing in both space and time. However at large costs, the accessible normalised configuration volume is constrained to grow along one dimension (time) alone, thus $V = \Theta\,(c)$.

While the Manhattan-metric is being used for a tile-based NoC topology, we also note that this analysis can easily be adapted to arbitrary NoC topologies including binary-trees, fat-trees, butterfly-networks, etc... However, if the spatial costs are largely due to physical movement of data over links (with cost proportional to spatial distance), then the accessible configuration volume is still constrained to grow at best in a 3-D manner at small cost scales, instead of the exponential growth normally due to trees or butterflies. This restriction is key – as it is not the NoC topology that matters so much as the actual physical communication costs. The result of using a more complex topology may lead to even higher embedding costs than the simple 2-D mesh.

7.1.2 Summary of Key Results

For the convenience of the reader, here is a summary of the main results from derivations in this chapter (further, less important, results are embedded in the rest of the chapter):

1. The spatial Rent's exponent p' is related to the spatio-temporal Rent's exponent $p > \frac{1}{2}$ of software by:
$$p' = 2 - \frac{1}{p}$$

2. For P cores in a 2D mesh, and sufficiently large Rentian memory of size M words *per core*, total I/O bandwidth requirements B_{tot} scale according to:

$$B_{tot} \propto M^{p'-1}P^{p'}$$

$$\left[B_{tot} \approx \left(\frac{p}{1-p}\right)\left(\frac{\eta}{p}\right)^{\frac{1}{p}} M^{p'-1}P^{p'}\right]$$

3. The asymptotic result (2) is unaffected by the communication topology of the cores (i.e. mesh, tree, etc..), *provided there is sufficient on-chip memory*

4. The asymptotic result (2) is unaffected by the fineness or coarseness of spatial communication, *provided there is sufficient on-chip memory*

5. However, result (4) may eventually be impractical as a rapidly growing amount of memory is needed *per core* to remain in this asymptotic region. The size of the 'critical region' of memory sizes $M < M_{crit}$, where spatial locality is still being exploited and asymptotic result (2) does not hold, grows for spatial communication coarseness factor ψ, and the number of cores P according to:

$$M_{crit} \approx O\left(\frac{\psi^p P^{\frac{3}{2}p}}{p\zeta\,(2-p)}\right)$$

Thus as the number of cores increases, spatial communication must become more fine-grain, with ψ scaling with order:

$$O\left(M^{\frac{1}{p}}P^{-\frac{3}{2}}\right)$$

So that to remain in the asymptotic region of (2) requires τ, the average interval of a core communicating a word of data to another core, to scale by:

$$\tau \sim \Omega\left(M^{\frac{1-p}{p}}P^{-\frac{3}{2}(1-p)}\right)$$

6. That for a real cache of size M_{cache} versus an ideal Rentian memory of size $M_{rentian}$, the effective Rentian memory scales asymptotically independent of cache architecture, according to:

$$M_{rentian} \propto M_{cache}^{p}$$

with the bandwidth scaling behaviour of:

$$B_{tot} \propto M_{cache}^{p-1}P^{p'}$$

(Note the replacement of the spatial Rentian exponent with the larger spatio-temporal version instead)

7. With area a scaling, and with I/O bandwidth scaling by (Rentian) exponent $\beta < p'$, in the external I/O-limited case, the optimal number of cores c scales according to (for a system with Rentian memory):

$$c \propto\sim a^{1-p'+\beta}$$

and for a cache-based memory system by:

$$c \propto\sim a^{(1-p+\beta)/(1-p+p')}$$

Some of these derivations are quite surprising, particularly that on-chip topology and coarse-grain communication don't affect the asymptotic scaling of I/O bandwidth provided sufficient memory is on-chip. It is also interesting that even a full exploitation of spatio-temporal locality with an idealised 'perfect memory' is still not sufficient to tackle the I/O bandwidth scaling problem, although it certainly helps to constrain the on-chip memory growth.

7.2 Rentian many-core architecture

If we treat software as exhibiting Rentian locality in space and time, then we can analyse the effect of scaling the number of cores and memory on I/O bandwidth requirements. The Tilera architecture, based on the RAW platform [111, 101, 102], is probably the closest current architecture to supporting Rentian behaviour, as spatio-temporal interconnect can be explicitly allocated by the software, and instructions can be mapped/scheduled in space-time so as to minimise this spatio-temporal communication. The following section examines the spatio-temporal model in more detail for a 2-D mesh of processors. This is the spatio-temporal Rentian model used in Chapter 5.

7.2.1 Configuration-space analysis for the 2-D mesh

Let us consider a model where moving data in time costs c_T and moving data in space-alone costs c_S. Let us define a relative cost parameter $\theta = c_S/c_T$. This means that a movement of data in one unit of space and time (because spatial data movement is not instantaneous) costs $c_S + c_T$ is equivalent to $(\theta + 1)$ movements in time-alone. We would expect that θ could be quite large in the case of CMP – so, for convenience, let us assume that θ is approximated by the nearest integer value.

We are interested, then, in the volume V of configuration space that is reachable within a cost c. Let s be the number of equivalent temporal-only steps, i.e. $s = c/c_T$, which is an integer value. So we can instead define $V(s)$ as the volume of configuration space that is reachable within s equivalent temporal-only steps. However, we also need to restrict the speed of communication. Let us simply have each spatial-step require ρ temporal-steps as well.

We wish to determine the number of valid spatio-temporal links of a given cost. Christie and Stroobandt have previously derived a result for $D(l_S, L)$ – the number of spatial-only links of length l_S in a square mesh of side-length L [29] under a Manhattan-metric. Their result assumed that two nodes A and B with link $A \leftrightarrow B$ should only count the link once. However, for our purposes because it involves communication in time, we have: $A \to B'$ and $B \to A'$ where prime denotes a future version of the node. Thus in the spatio-temporal domain, we need to double their result, and also account for temporal-only communication where $l_S = 0$. This is shown below:

$$D(l_S, L) \equiv \begin{cases} L^2 & , l_S = 0 \\ \frac{2}{3} l_S \left(l_S^2 - 1 + 6L(L - l_S)\right) & , 1 \le l_S < L \\ \frac{2}{3}(2L - l_S + 1)(2L - l_S)(2L - l_S - 1) & , L \le l_S \le 2L \\ 0 & otherwise \end{cases}$$

We can now calculate the volume of reachable (re-normalised) configuration-space with *exactly* cost sc_T as:

$$V_\Delta(s) \equiv \frac{1}{L^2} \sum_{k=0}^{\lfloor s/(\theta+\rho) \rfloor} D(k, L)$$

We notice that the relative cost of spatial communication θ and the slowness of spatial communication ρ appear together as a sum. As θ is likely to be much larger then ρ, let us merely define $\psi = \theta + \rho$ noting that $\psi \sim \theta$. Then:

$$V_\Delta(s) = \frac{1}{L^2} \sum_{k=0}^{\lfloor s/\psi \rfloor} D(k, L)$$

Let us the define $Q(r,\ L)$ to be the non-normalised volume reachable at distance r by:

$$\begin{aligned} Q(r,\ L) &\equiv \sum_{k=0}^{r} D(k, L) \\ &= \begin{cases} \frac{1}{6}(6L^2 - 2r - 4Lr + 12L^2 r - r^2 - 12Lr^2 \\ +12L^2 r^2 + 2r^3 - 8Lr^3 + r^4) & , 0 \le r < L \\ \frac{1}{6}(-4L + 4L^2 + 16L^3 - 10L^4 + 2r - 4Lr - 24L^2 r \\ +32L^3 r + r^2 + 12Lr^2 - 24L^2 r^2 - 2r^3 + 8Lr^3 - r^4) & , L \le r < 2L \\ L^4 & , r \ge 2L \end{cases} \end{aligned}$$

Then the normalised volume reachable at distance r is:

$$V_\Delta(s) = \frac{1}{L^2} Q\left(\left\lfloor \frac{s}{\psi} \right\rfloor, L\right) \tag{7.1}$$

Let V_{max} be the maximum configuration volume – essentially constrained by the problem-size or phase of execution. The total volume of configuration-space within cost sc_T (including cost 0 – just the original node itself) is then:

$$\begin{aligned}
V(s) &= \min\left\{V_{max},\, \sum_{r=0}^{s} V_{\Delta}(r)\right\} \\
&= \min\left\{V_{max},\, \frac{1}{L^2}\sum_{r=0}^{s} Q\left(\left\lfloor\frac{r}{\psi}\right\rfloor,\, L\right)\right\} \\
&= \min\left\{V_{max},\, \frac{1}{L^2}(s \bmod \psi)\, Q\left(\left\lfloor\frac{s}{\psi}\right\rfloor,\, L\right) + \frac{\psi}{L^2}\sum_{r=0}^{\lfloor s/\psi\rfloor-1} Q(r,\, L)\right\}
\end{aligned}$$

Let us define $R(q,\, L)$ to be the non-normalised volume reachable within distance less than or equal to q:

$$\begin{aligned}
R(q,\, L) &\equiv \sum_{r=0}^{q} Q(r,\, L) \\
&= \begin{cases}
\frac{1}{30}(120L - 90L^2 - 6q - 20Lq + 70L^2q - 5q^2 - 50Lq^2 & \\
+60L^2q^2 + 5q^3 - 40Lq^3 + 20L^2q^3 + 5q^4 - 10Lq^4 + q^5) & ,\, 0 \leq q < L \\
\frac{1}{30}(108L - 100L^2 + 60L^3 - 50L^4 + 12L^5 + 6q - 20Lq & \\
-60L^2q + 160L^3q - 50L^4q + 5q^2 + 30Lq^2 - 120L^2q^2 & \\
+80L^3q^2 - 5q^3 + 40Lq^3 - 40L^2q^3 - 5q^4 + 10Lq^4 - q^5) & ,\, L \leq q < 2L \\
\frac{1}{3}(12L - 12L^2 + 2L^3 + 3L^4 - 2L^5 + 3L^4q) & ,\, q \geq 2L
\end{cases}
\end{aligned}$$

Removing the floor-function, we have a reasonable continuous interpolation as:

$$V(s) \approx \min\left\{\frac{\psi}{L^2}R\left(\frac{s}{\psi},\, L\right),\, V_{max}\right\}$$

Noting that this is exact for s a multiple of ψ.

We also note that for $s \geq 2\psi L$ that $V_{\Delta}(s) = 0$, and so let $V_{crit} = V(2\psi L)$.

Then for $s \geq 2\psi L$:

$$V(s) = \min\left\{V_{crit} + (s - 2\psi L)\, L^2,\, V_{max}\right\}$$

Where $V_{crit} = \frac{1}{3}(12L^{-1} - 12 + 2L + 3L^2 + 4L^3)\,\psi$

If we normalise the analysis to have one link per node, on average, to a future instruction, then for $P = L^2$ cores, we have P links in total. We are going to use a complete continuous power-law distribution, so let us first calculate the normalisation factor η (i.e. λ_V in eqn 6.7):

$$\eta(1-p)\int_{1}^{V_{max}} v^{p-2}dv = 1$$

Where p is the spatio-temporal Rent's exponent. Thus $\eta = \frac{1}{\left(1-V_{max}^{p-1}\right)}$. If $V_{max} = \infty$, we just have $\eta = 1$. We can then determine the total expected number of links with cost sc_T as:

$$\begin{aligned} E(s) &= L^2\eta(1-p)\int_{V(s-1)}^{V(s)} v^{p-2}dv \\ &= L^2\eta\left((V(s-1))^{p-1} - (V(s))^{p-1}\right) \end{aligned}$$

7.2.2 Separating Spatial and Temporal components

Although interconnect is allocated in a spatio-temporal manner, we are interested in the total number of expected links with particular temporal length or particular spatial length.

To do this, we need to project out the spatial and temporal components from those spatio-temporal links.

For spatio-temporal cost distance sc_T there is a volume $V_\Delta(s)$ of which the proportion of links that have spatial distance $l_S \leq \frac{s}{\psi}$ is:

$$f_S(s, l_S) = \frac{D(l_S, L)}{Q\left(\left\lfloor \frac{s}{\psi} \right\rfloor, L\right)}$$

For spatio-temporal cost distance sc_T there is a volume $V_\Delta(s)$ of which the proportion of links that have temporal distance $l_T \leq s$ is:

$$f_T(s, l_T) = \frac{D\left(\left\lfloor \frac{s-l_T}{\psi} \right\rfloor, L\right)}{Q\left(\left\lfloor \frac{s}{\psi} \right\rfloor, L\right)}$$

Then the total expectation of links with spatial distance l_S within V_{max} is:

$$\begin{aligned} E_S(l_S) &= \sum_{s=\psi l_S}^{\infty} f_S(s, l_S)E(s) \\ &= L^2\eta\sum_{s=\psi l_S}^{\infty}\frac{D(l_S, L)}{Q(s, L)}\left((V(s-1))^{p-1} - (V(s))^{p-1}\right) \\ &= L^2\eta\sum_{k=l_S}^{\infty}\frac{D(l_S, L)}{Q(k, L)}\left(\left(\frac{\psi R(k-1, L)}{L^2}\right)^{p-1} - \left(\frac{\psi R(k, L)}{L^2}\right)^{p-1}\right) \\ &= \eta\psi^{p-1}L^{4-2p}\sum_{k=l_S}^{\infty}\frac{D(l_S, L)}{Q(k, L)}\left((R(k-1, L))^{p-1} - (R(k, L))^{p-1}\right) \end{aligned}$$

Of this, we can separate out the component where $k \geq 2L$, past the critical volume V_{crit} where each node can reach any other node:

$$\sum_{k=2L}^{\infty}\frac{D\left(l_S,\,L\right)}{Q\left(k,\,L\right)}\left(\left(V\left(k-1\right)\right)^{p-1}-\left(V\left(k\right)\right)^{p-1}\right) \;=\; \frac{D\left(l_S,\,L\right)}{L^4}\left(V_{crit}^{p-1}-V_{max}^{p-1}\right)$$

yielding:

$$E_S\left(l_S\right)=\frac{\eta\psi^{p-1}}{L^{2p-4}}D\left(l_S,\,L\right)\left(\frac{\left(V_{crit}^{p-1}-V_{max}^{p-1}\right)}{L^4}+\sum_{k=l_S}^{2L-1}\frac{\left(\left(R\left(k-1,\,L\right)\right)^{p-1}-\left(R\left(k,\,L\right)\right)^{p-1}\right)}{Q\left(k,\,L\right)}\right)$$

We note that upon normalising the expected number of links into a distance-distribution, that the constant terms at the front vanish. This means that the approximate distance distribution is independent of ψ – the cost of spatial-communication. Instead, the distribution is dependent on p and L alone. So the 2-D spatial Rentian distance *distribution* is independent of the coarseness of communication, although the quantity of communication can certainly vary with ψ. We also should note that as p increases, the proportion of links to other nodes, independent of spatial position, grows. This is because V_{crit}^{p-1} starts to dominate over the rest of the distance-dependent terms. Indeed, as p approaches unity, the total distribution approaches $D\left(l_S, L\right)$, which is the uniform distribution.

Similarly, the total expectation of links with temporal distance l_T is (if within V_{max}):

$$\begin{aligned}
E_T\left(l_T\right) &= \sum_{s=l_T}^{\infty} f_T\left(s,\,l_T\right)E\left(s\right)\\
&= L^2\eta\sum_{s=l_T}^{\infty}\frac{D\left(\left\lfloor\frac{s-l_T}{\psi}\right\rfloor,\,L\right)}{Q\left(\left\lfloor\frac{s}{\psi}\right\rfloor,\,L\right)}\left(\left(V\left(s-1\right)\right)^{p-1}-\left(V\left(s\right)\right)^{p-1}\right)\\
&\approx L^2\eta\sum_{k=\lfloor l_T/\psi\rfloor}^{\infty}\frac{D\left(k-\left\lfloor\frac{l_T}{\psi}\right\rfloor,\,L\right)}{Q\left(k,\,L\right)}\left(\left(V\left(k\psi\right)\right)^{p-1}-\left(V\left(\left(k+1\right)\psi\right)\right)^{p-1}\right)\\
&= \eta L^2\sum_{k=\lfloor l_T/\psi\rfloor}^{\infty}\frac{D\left(k-\left\lfloor\frac{l_T}{\psi}\right\rfloor,\,L\right)}{Q\left(k,\,L\right)}\left(\left(\frac{\psi}{L^2}R\left(k\psi\right)\right)^{p-1}-\left(\frac{\psi}{L^2}R\left(\left(k+1\right)\psi\right)\right)^{p-1}\right)\\
&= \eta\frac{\psi^{p-1}}{L^{2p-4}}\sum_{k=\lfloor l_T/\psi\rfloor}^{\infty}\frac{D\left(k-\left\lfloor\frac{l_T}{\psi}\right\rfloor,\,L\right)}{Q\left(k,\,L\right)}\left(\left(R\left(k\psi\right)\right)^{p-1}-\left(R\left(\left(k+1\right)\psi\right)\right)^{p-1}\right)\\
&= \eta\frac{\psi^{p-1}}{L^{2p-4}}\sum_{m=0}^{2L}\frac{D\left(m,\,L\right)}{Q\left(m+\left\lfloor\frac{l_T}{\psi}\right\rfloor,\,L\right)}\left(\left(R\left(m\psi+l_T\right)\right)^{p-1}-\left(R\left(m\psi+l_T+\psi\right)\right)^{p-1}\right)
\end{aligned}$$

Beyond the critical volume V_{crit} the behaviour is one-dimensional and we have for $l_T \gg 2\psi L$:

$$
\begin{aligned}
E_T\left(l_T\right) &\approx \eta L^2 \sum_{k=\lfloor l_T/\psi \rfloor}^{\infty} \frac{D\left(k - \left\lfloor \frac{l_T}{\psi} \right\rfloor ,\, L\right)}{L^4} \left(\left(V_{crit} + \left(k-1-2L\right)\psi L^2\right)^{p-1}\right. \\
&\qquad \left. - \left(V_{crit} + \left(k-2L\right)\psi L^2\right)^{p-1}\right) \\
&\approx \eta L^2 L^{-4} \sum_{m=0}^{2L} D\left(m,\, L\right) ((V_{crit} + (m - 1 - 2L + \frac{l_T}{\psi})\psi L^2)^{p-1} \\
&\qquad -(V_{crit} + (m - 2L + \frac{l_T}{\psi})\psi L^2)^{p-1}) \\
&\approx \eta L^2 L^{-4}\left(L^4\right)\left(\left(V_{crit} + \left(l_T - \psi\right)L^2\right)^{p-1} - \left(V_{crit} + l_T L^2\right)^{p-1}\right) \\
&\approx \eta L^2 \left(\left(\left(l_T - \psi\right)L^2\right)^{p-1} - \left(l_T L^2\right)^{p-1}\right) \\
&\approx \eta \psi^{p-1} L^{2p} \left(\left(\frac{l_T}{\psi} - 1\right)^{p-1} - \left(\frac{l_T}{\psi}\right)^{p-1}\right) \\
&\approx \eta \psi^{p-1} L^{2p} \left(1-p\right)\left(\frac{l_T}{\psi}\right)^{p-2} \\
&= \eta \psi L^{2p} \left(1-p\right)\left(l_T\right)^{p-2}
\end{aligned}
$$

So the temporal distance distribution varies asymptotically by $O\left(\left(l_T\right)^{p-2}\right)$, which is exactly what we'd expect for a simple one-dimensional embedding of a Rentian system. This is an asymptotic form, and the temporal-distance distribution is more complicated at smaller time-scales, due to the availability of spatial locality. Interestingly, for a given temporal distance l_T the total number of links varies by $O\left(P^p\right)$ for $P = L^2$ cores, instead of by $O\left(P\right)$, demonstrating the effect of Rentian sharing between cores – as there is more spatial communication to reduce temporal communication.

Note that many of these derivations utilise sums up to configuration volume V_{max} – a finite number of embedded instructions with a particular Rentian exponent. We would expect the configuration volume to be constrained by the number of instructions needed to tackle a particular problem-size / data-set, in a single phase of program execution. However, for large problem-size, the deviation in statistics from an 'infinite' volume are small (of order V_{max}^{p-1} for $V_{max} \gg 1$), thus we may as well use the limiting case $V_{max} \to \infty$ for asymptotic derivations.

7.3 Rentian memory and external I/O bandwidth

Here, we derive results that relate the amount of memory per core to the external I/O bandwidth. We note that transportation of data in time necessarily takes up memory. The amount of memory required is proportional to its temporal distance. For example, if all N tiles produced an item of data that was merely consumed the next cycle, and this behaviour occurred every cycle, then it would merely need memory N. If instead, it was

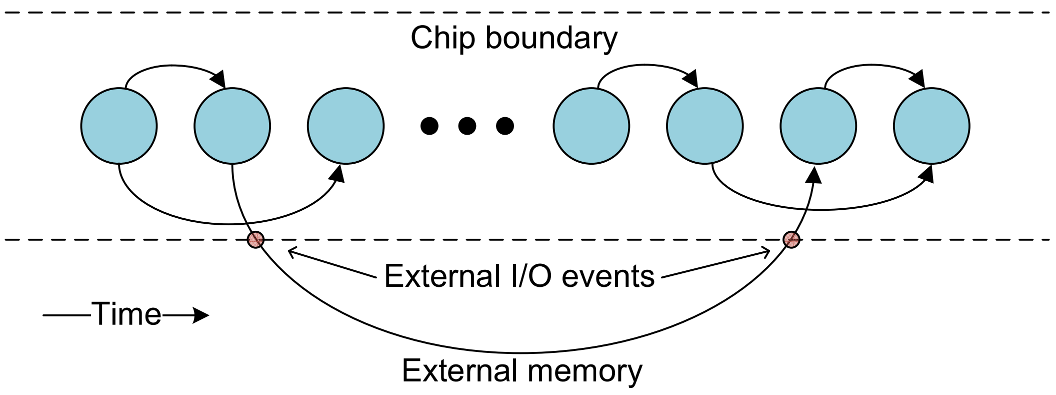


Figure 7.2: Illustrating how long temporal-interconnects go off-chip, thereby inducing external I/O. Here the nodes are instructions, and the arrows between them represent data-flow, with a large temporal gap between nodes represented by ellipses. This example is with a single core, but there can be many cores in parallel, with data flowing to other instructions in the same core, or to other cores, and via external memory.

consumed every T cycles, then it would need memory NT. Figure 7.2 helps illustrate the general strategy of subsequent derivations. If we exclude the initial fetching of data from external memory, then we can think of external I/O events as occurring when a temporal interconnect crosses into external memory. Each temporal interconnect that goes off-chip takes up a single word of output I/O going off-chip to external memory and a single word of input I/O coming back, however we will just count them as one I/O word. There is limited capacity for on-chip temporal interconnect (on-chip memory), and the surplus is effectively offloaded to external working memory. So, unless the working-set is small enough to fit in on-chip memory, there is necessarily a partition by external I/O into temporal interconnect that resides entirely on-chip, and that which goes off-chip. However, some partitionings can be better that others. A good partitioning, for a given total on-chip memory, should best allocate the temporal interconnect on-chip so as to minimise the number of crossings into external memory.

Let us call this idealised perfect on-chip memory a *Rentian Memory*. The name, as we shall see later, is because such a memory follows Rentian scaling, much like VLSI wires do. Such a memory is certainly not representative of caches, which will be handled later on, but it may be possible for a well-managed scratchpad memory to be close in performance to a Rentian Memory.

When considering this memory, we must narrow our focus to the temporal component of the spatio-temporal interconnect. Let λ_k be the expected number of links that are of exactly temporal length k (irrespective of the spatial-length component), and let l_k be the proportion of links of temporal length k that stay on-chip. i.e.

$$\forall k,\ 0 \leq l_k \leq 1$$

For convenience let us define: $u_k = l_k \lambda_k$

We can characterise the amount of per-core external I/O bandwidth B as being the sum of the *number* of wires that don't fit in memory, that is:

$$B = \sum_k (1 - l_k)\,\lambda_k = 1 - \sum_k l_k \lambda_k = 1 - \sum_k u_k$$

Note that this is bandwidth in one direction, we should multiply by two to account for both outgoing and incoming communication. The amount of memory M needed to accommodate the on-chip temporal interconnects depends on their lengths:

$$M = \sum_k u_k k$$

For a given value of $M > 0$, to minimise B one needs to maximise $\sum u_k$ under the constraint: $0 \leq u_k \leq \lambda_k$ and of M above.

Lemma 7.1. *The solution that minimises external I/O bandwidth B for on-chip memory $M = \sum_k u_k k$ occurs when $\{u_k\} = \{\lambda_1, ..., \lambda_{s-1}, \beta, 0, ..., 0\}$ where $u_s = \beta$ for some s, $0 < \beta \leq \lambda_s$. Thus a greedy allocation from shortest temporal interconnect to longest is optimal.*

Proof. Assume the solution is not so, then without loss of generality, we can choose the last non-zero entry to be s with value $l_s > 0$. There must exist an entry $0 \leq u_r < \lambda_r$, s.t. $r < s$. Let $\delta_r = \lambda_r - u_r$, then we can increase u_r by $\delta = \max\left\{\delta_r, \frac{r}{s} l_s\right\}$ and decrease u_s by $\frac{s}{r}\delta$. But this decreases B by $\frac{s-r}{r}\delta > 0$, which implies the original solution was not minimal, and is thus a contradiction ☐

All this means is that with limited memory storage on chip, to minimise overflow to external memory I/O, only the shortest temporal interconnects should be stored on chip (i.e. the ones with the greatest temporal locality).

7.3.1 Many-core scaling

Lemma 7.2. *A spatio-temporal embedding with Rent's exponent p leads to a spatial Rent's exponent of $p' = \max\left\{2 - \frac{1}{p}, 1 - \frac{1}{d}\right\}$ for spatial embedding dimension d.*

Proof. For a fractal-dimensional computation graph of dimension f_D embedded in some space we expect:

$$p = 1 - \frac{1}{f_D}$$

However, when viewing just the spatial-embedding, we are contracting along the temporal dimension. Thus we are reducing $f_D > 2$ by one, yielding:

$$\begin{aligned} p' &= 1 - \frac{1}{f_D - 1} \\ &= \frac{f_D - 2}{f_D - 1} \\ &= 2 - \frac{f_D}{f_D - 1} \\ &= 2 - \frac{1}{p} \end{aligned}$$

However, we know that p' is constrained to be at a minimum equal to $1 - \frac{1}{d}$ for spatial embedding dimension d [109]. Thus yielding $p' = \max\left\{2 - \frac{1}{p},\ 1 - \frac{1}{d}\right\}$. □

Now we can properly tackle the problem of how external I/O bandwidth scales given memory size, number of cores, and the Rent's exponent of the algorithm.

Theorem 7.3. *Let P be the number of cores, and M the memory* per core. *Then, at large problem-sizes, a Rentian many-core processor exhibits the following scaling for total external I/O bandwidth:*

$$B_{tot} \propto M^{p'-1} P^{p'} \tag{7.2}$$

for Rentian spatial exponent p' as per Lemma 7.2.

Proof. We will first start with any topology of cores such that each core can reach any other core within S spatio-temporal steps, and has longest spatial length L. To minimise cost, a generalised Rentian embedding is given by the configuration-space with:

$$\lambda_k = \eta \int_{V_{k-1}}^{V_k} v^{p-2} dv$$

where λ_k is the proportion of wires with *spatio-temporal* cost distance k, p is the Rent's exponent, V_k is the number of nodes within cost distance c_k, where c_k is the monotonically increasing set of unique cost distances in the topology, and η is a constant normalisation factor for the distribution.

Regardless of topology (that is at least 2D), we can assume that after some small multiple of L spatio-temporal steps, each spatial node reaches every other spatial node, and so the volume V increases by P per step thereafter. We want to examine behaviour for $V \gg V_{crit}$, so let us choose $S \gg L$ spatio-temporal steps as an appropriate distance with respective configuration volume $V_S \gg V_{crit}$ and memory M_S.

Let us define $k(V)$ as the spatio-temporal cost distance for volume $V \geq V_S$:

$$k(V) = \left\lfloor \frac{V - V_S}{P} \right\rfloor + S$$

Then the temporal distances for a spatio-temporal cost distance k has a range $[k - \psi L, k]$ which can be simply approximated by k as $\psi L \ll S \leq k$.

We are interested in the behaviour of M for large V_{cutoff}, which we set as the configuration volume cut-off for the partition between on-chip temporal interconnect and off-chip temporal interconnect. We employ a greedy-strategy of on-chip memory allocation, which is I/O bandwidth optimal as per Lemma 7.1. Let l_k be the proportion of links of temporal length k that stay on-chip, and let us define: $u_k = l_k \lambda_k$. Then:

$$\begin{aligned}
M &= \sum_k u_k k = M_S + \sum_{k=S}^{\infty} u_k k \\
&\approx M_S + \eta \int_{V_S}^{V_{cutoff}} v^{p-2} k(v)\, dv \\
&\approx M_S + \eta \int_{V_S}^{V_{cutoff}} v^{p-2} \left(\frac{v - V_S}{P} + S \right) dv \\
&= M_S + \eta \frac{V_{cutoff}^p - V_S^p}{pP} + \eta (SP - V_S) \frac{V_{cutoff}^{p-1} - V_S^{p-1}}{(p-1)P}
\end{aligned}$$

For $V^p \gg V_S^p$:

$$M \approx M_S + \frac{\eta}{pP} V_{cutoff}^p$$

With asymptotic behaviour:

$$\log M \approx p \log V_{cutoff} + \log \frac{\eta}{pP}$$

Let us then characterise the external I/O bandwidth B per core at large V_{cutoff}:

$$\begin{aligned}
B &= 1 - \sum_k u_k \\
&\approx \eta \int_{V_{cutoff}}^{\infty} v^{p-2}.dv \\
&= \eta \frac{V_{cutoff}^{p-1}}{1-p}
\end{aligned}$$

Then for large V_{cutoff}:

$$\log B \approx (p-1) \log V_{cutoff} + \log \frac{\eta}{(1-p)}$$

We then have:

$$\begin{aligned}\log B &\approx \left(1-\frac{1}{p}\right)\left(\log M - \log\frac{\eta}{pP}\right) + \log\frac{\eta}{(1-p)} \\ &= \left(1-\frac{1}{p}\right)\left(\log M + \log P + \log\frac{p}{\eta}\right) + \log\frac{\eta}{(1-p)} \qquad (7.3)\end{aligned}$$

Then at large M we necessarily have large V and hence:

$$\frac{\partial \log B}{\partial \log M} = 1 - \frac{1}{p}$$

And we also have at large M:

$$\frac{\partial \log B}{\partial \log P} = 1 - \frac{1}{p}$$

As B is the I/O bandwidth per core, to arrive at the total bandwidth $B_{tot} = C.B$ then:

$$\frac{\partial \log B_{tot}}{\partial \log P} = 2 - \frac{1}{p}$$

This means that the total I/O bandwidth scales according to:

$$B_{tot} \propto P^{2-\frac{1}{p}}$$

At first appearances this might seem to be different from an expected Rentian behaviour of $B_{tot} \propto P^p$. However, we need to remember that this is the purely spatial, rather than spatio-temporal scaling behaviour. From Lemma 7.2 we can then write:

$$B \propto (MC)^{p'-1} \qquad (7.4)$$

with constant of proportionality from eqn 7.3 of order ~$\left(\frac{p}{1-p}\right)\left(\frac{\eta}{p}\right)^{\frac{1}{p}}$, i.e.

$$B_{tot} \approx \left(\frac{p}{1-p}\right)\left(\frac{\eta}{p}\right)^{\frac{1}{p}} M^{p'-1}P^{p'}$$

Or more directly, from eqn (7.4):

$$B_{tot} \propto M^{p'-1}P^{p'}$$

□

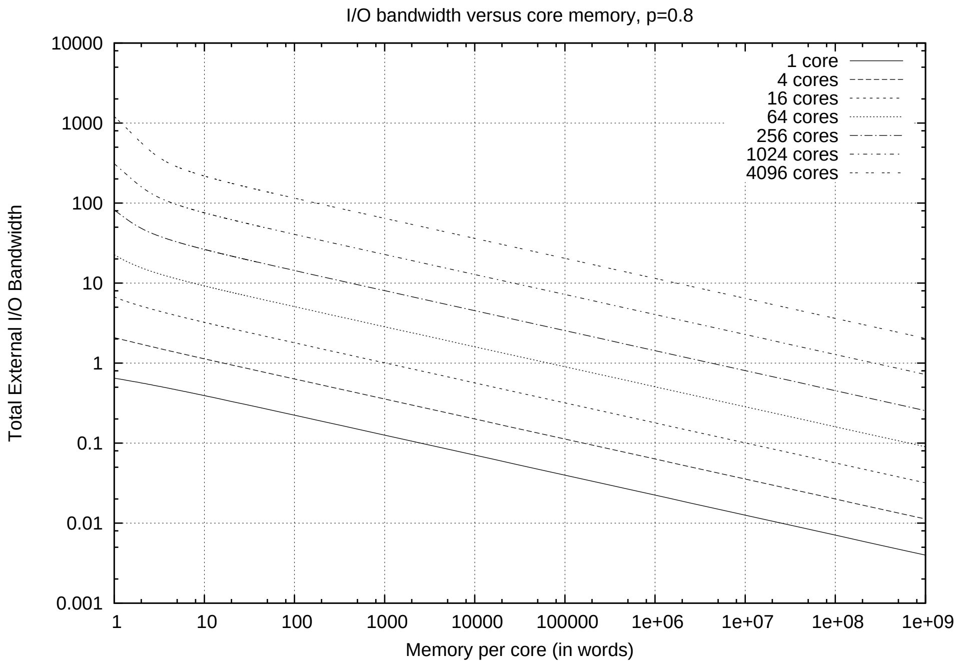


Figure 7.3: Numerical evaluation of model illustrating how external I/O bandwidth scales with the number of cores and the memory per core for a 2-D mesh CMP. Note that this is asymptotic behaviour with an unbounded 'working-set' size, and excluding 'compulsory misses' where new data is fetched into on-chip memory for the very first time. An I/O bandwidth of one, here, equates to one read/write access per instruction.

This characterises the scaling of external I/O bandwidth with respect to on-chip memory per core, and the number of cores. The spatio-temporal Rentian model can be directly numerically evaluated from the configuration-space description, and source code is shown in Appendix F for this. An example from a numerical evaluation of the generalised Rentian model and 2-D mesh topology is given in figure 7.3 with $p = 0.8$ and thus $p' = 0.75$. The slopes in the tail region when fit with a least squares linear regression yield $-0.25 \pm 0.02\%$ as expected. The slopes across cores is $0.75 \pm 0.03\%$, also as expected. One can see how in the head region, the availability of other cores allows fine-grain spatial locality to be exploited, which once used up, leads to the limiting-case behaviour of temporal-only locality. Note that this derivation didn't make any assumptions about the spatial topology of the CMP. Depending on the spatial topology, the head-region may be extended outwards, and thus transition to asymptotic tail behaviour at larger memory sizes. Note that figure 7.3 assumes that $\psi = 1$, that is very fine-grain communication between cores. We will cover the dependence on the cost of communication in the next section.

7.3.2 Accounting for costly spatial communication

The previous derivations assumed extremely fine-grain spatial communication, implicitly assuming that it is as cheap as on-chip temporal communication. This is a rather unrealistic assumption to make given how very little energy it takes to move data in time (by merely storing it) versus in space. We define a cost ratio $\psi \in \mathbb{N}^*$ to be the equivalent temporal cost it takes to move data one unit in space plus the minimum number of units in time (to an adjacent core). So one way to think of parameter ψ is that it describes how coarse grain the spatial communication is. To minimise cost, a generalised Rentian embedding is given by:

$$\tilde{\lambda}_k = \alpha \sum_{V=V_{k-1}+1}^{V_k} V^{p-2}$$

and

$$\lambda_k = \sum_{s=0}^{2N} \kappa_{k,s} \tilde{\lambda}_{k+\theta s}$$

where $\tilde{\lambda}_k$ is the proportion of wires with *spatio-temporal* cost distance k, λ_k is the proportion of wires with *temporal* distance k, and $\kappa_{k,s}$ is the proportion of spatio-temporal cost distance $(k + \psi s)$ that has temporal distance k.

In a 2-D mesh, for the first $\psi - 1$ steps, there are only temporal steps which do not spread spatially to other cores, but after the ψ^{th} step there are nodes that are one spatial step away. To calculate λ_k we must sum up all the bits with just temporal distance k from $\tilde{\lambda}$. For the special case of $\psi = 1$ we have $\lambda_k = \tilde{\lambda}_k$ as earlier. At $2\psi N$ spatio-temporal steps, each spatial node reaches every other spatial node, and so the volume V increases by P per step thereafter. This also means that $\kappa_{k,s}$ is in a region where there is no more spatial growth and thus the proportions affecting λ_k do not change with k. We denote this with $\kappa_{k,s} = \kappa_s, \forall k \geq 2\psi N$. Let us characterise this 'tail' region of the distribution.

Our derivation follows similarly to section 7.3.1, with:

$$\begin{aligned}
B &= 1 - \sum_k u_k \\
&= -u_s + \sum_{k=s+1}^{\infty} \lambda_k \\
&\approx \alpha \int_V^{\infty} \sum_{s=0}^{2N} \kappa_{k,s} \left(v + \psi s\right)^{p-2} .dv \\
&= \alpha \sum_{s=0}^{2N} \kappa_s \int_V^{\infty} \left(v + \psi s\right)^{p-2} .dv \\
&= \alpha \sum_{s=0}^{2N} \kappa_s \frac{\left(V + \psi s\right)^{p-1}}{1-p}
\end{aligned}$$

For $V \gg 2\psi N$ we have:

$$\begin{aligned} B &\approx \alpha \sum_{s=0}^{2N} \kappa_s \frac{V^{p-1}}{1-p} \\ &= \alpha \frac{V^{p-1}}{1-p} \sum_{s=0}^{2N} \kappa_s \\ &= \alpha \frac{V^{p-1}}{1-p} \end{aligned} \tag{7.5}$$

So that the behaviour for large V is unaffected as:

$$\log B = (p-1)\log V + \log \frac{\alpha}{(1-p)}$$

For M there is a larger M_{crit}, but as we are only interested in the asymptotic behaviour at large M, we can again approximate by:

$$\log M \approx p \log V + \log \frac{\alpha}{pP}$$

Leading us to the same solution as in section 7.3.1 with:

$$B_{tot} \propto M^{p'-1} P^{p'}$$

Given the solution at $P = 1$ is independent of ψ, the same constant of proportionality applies, and is unaffected by ψ.

We are also interested in the behaviour of M_{crit}, the point where the system transitions as it runs out of spatial locality to exploit. We know that both B_{crit} and M_{crit} are related by V_{crit}. Let us label M_{crit0}, B_{crit0} and V_{crit0} as the base case of $\psi = 1$. We then have, as a result of the ψ expansion:

$$V_{crit} = \psi V_{crit0}$$

Thus substituting into eqn 7.5:

$$B_{crit} = \psi^{p-1} B_{crit0}$$

and as from eqn (7.4):

$$\frac{B_{crit}}{B_{crit0}} \approx \left(\frac{M_{crit}}{M_{crit0}}\right)^{p'-1}$$

this yields:

$$\begin{aligned} M_{crit} &\approx \psi^{\frac{p-1}{p'-1}} M_{crit0} \\ &= \psi^{p} M_{crit0} \end{aligned}$$

The analytic expression for M_{crit0} as a function of p and P is rather lengthy but it is of order $O\left(\frac{\eta P^{\frac{3}{2}p}}{p}\right)$. We leave it as an exercise to work out by calculating V_{crit0}, then B_{crit0} and then approximating M_{crit0}.

Although we have introduced the *relative-cost* parameter ψ, this does not imply that the rate of spatial communication is inversely proportional to ψ. A value of $\psi = 100$ does not mean there is a hundred times less spatial communication than at $\psi = 1$. Indeed, we can show that the quantity of spatial communication approximately scales by ψ^{1-p}.

We can think about the Rentian summation in section 7.3.1 as like an inverted square pyramid that grows from its tip and then extends to a maximum area. We can relate the area of the cross section to the volume according to a function $A(V)$. The proportion of external communication for a given volume is then $f(V) = \frac{A(V)-1}{A(V)}$. The cost factor ψ effectively stretches the pyramidal shape by that factor, so that we instead use: $f\left(\frac{V}{\psi}\right)$ Then the bandwidth emanating from the core (excluding memory-effects) is given by:

$$\begin{aligned} B &= \eta \sum_{V=1}^{\infty} f\left(\frac{V}{\psi}\right) V^{p-2} \\ &\approx \eta \int_{1}^{\infty} f\left(\frac{V}{\psi}\right) V^{p-2} dV \\ &= \eta \int_{1/\psi}^{\infty} f(W)(W\psi)^{p-2} \psi dW \\ &= \eta \psi^{p-1} \int_{1/\psi}^{\infty} f(W) W^{p-2} dW \end{aligned}$$

But we know that $f(V) = 0$ for $V \leq 1$ as $A(V) = 1$, so:

$$\begin{aligned} B &\approx \psi^{p-1} \int_{1}^{\infty} \alpha f(W) W^{p-2} dW \\ &= \psi^{p-1} \kappa \end{aligned}$$

The constant κ is necessarily less than 1, since $f(W) < 1$, but as the number of cores becomes large, $\kappa \sim 1$, so that $B \sim \psi^{p-1}$. If on average a core communicates one word with its neighbours every τ cycles then $\psi \sim \tau^{\frac{1}{1-p}}$. Thus the maximum τ that can be supported whilst still in the asymptotic region scales approximately as:

$$\tau_{max} \sim \left(\frac{M}{M_{crit0}}\right)^{\frac{1-p}{p}}$$

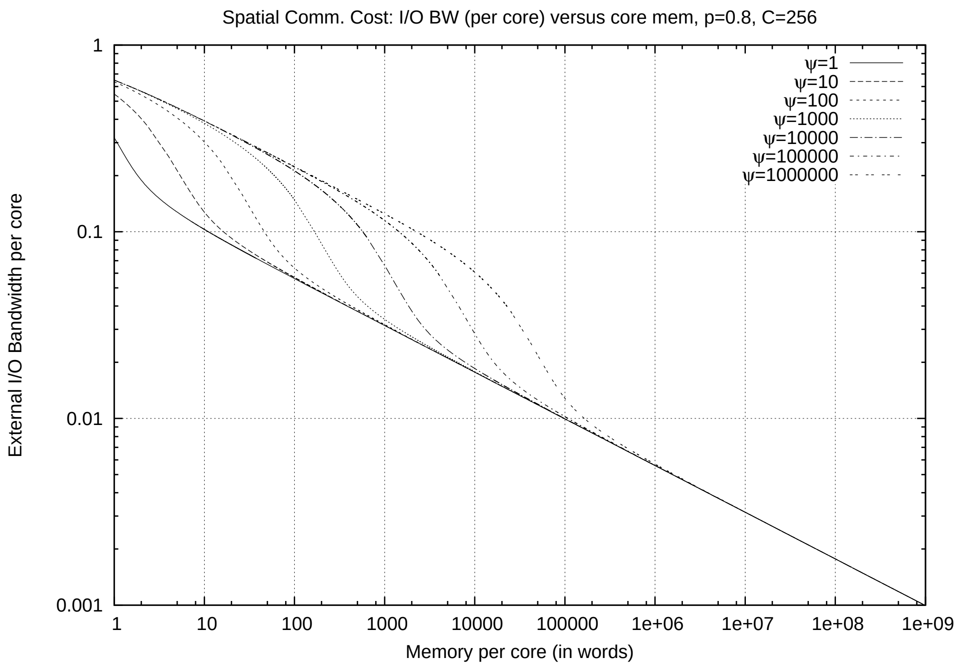


Figure 7.4: Numerical evaluation of impact of coarser-grain spatial-communication parameter ψ. An I/O bandwidth per-core of one, here, equates to one read/write access per instruction.

So for 10^6 words of Rentian storage per core (say 4MB), and $M_{crit0} \sim 10$ (reasonable for a 256-core system as seen in figures 7.3 and 7.4 with $\psi = 1$), and $p = 0.8$, then $\psi \sim 2\times10^6$ and $\tau_{max} \sim 20$, i.e. one would want each core to communicate a word with another core at least every ~20 cycles, otherwise the I/O bandwidth would grow dramatically as seen in figure 7.4 with the higher ψ values. In this figure we can see how the parameter ψ affects the ability to exploit spatial locality at low memory sizes – with a rapid drop in I/O bandwidth seen as locality finally gets exploited at that scale. This reduction in I/O bandwidth from the head of a distribution compared to the extrapolated asymptotic tail is due to spatial locality and becomes considerably larger with the number of cores as seen in figure 7.3 for 1024 and 4096 cores. We also note that in figure 7.4, while the asymptotic result is independent of ψ (the effective cost of spatial communication), the width of the head region is affected by it. A larger ψ effectively pushes out the region where spatial locality can be exploited, and hence where the asymptotic tail behaviour is valid. For larger numbers of cores, this means that an efficient spatial communication topology, with fine-grain communication is desired to maintain this asymptotic scaling with memory.

In figure 7.5 we see how the cost parameter ψ of one million, and a corresponding $\tau \sim 20$, pushes out the region where spatial-locality is exploited, compared to figure 7.3 with a ψ of one. Here, the asymptotic tail doesn't start until the order of $10^5 - 10^6$ words, depending on the number of cores.

An important observation to make is that although by having sufficient memory one can hide the performance effect of coarse-grain spatial communication, this strategy runs out of steam as the number of cores increases, thus also requiring finer grain commu-

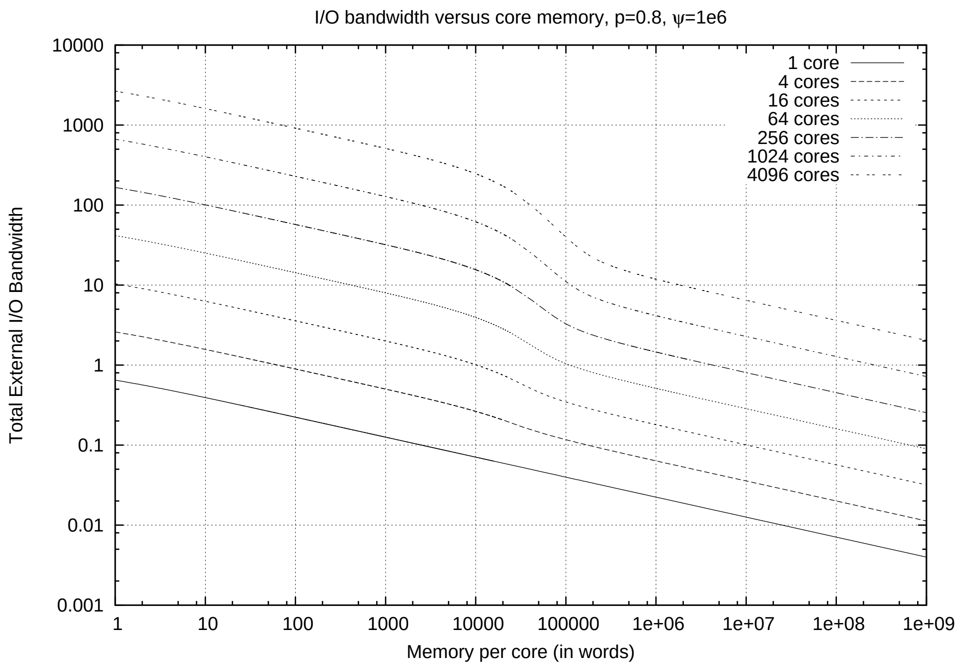


Figure 7.5: Numerical evaluation of model illustrating how external I/O bandwidth scales with the number of cores and the memory per core for a 2-D mesh CMP. Note that this is asymptotic behaviour with an unbounded 'working-set' size, and excluding 'compulsory misses' where new data is fetched into on-chip memory for the very first time. An I/O bandwidth of one, here, equates to one read/write access per instruction.

nication in order to place performance back into the asymptotic tail region. This is a surprising result. The ψ scaling required to stay in the asymptotic region approximately follows $\Omega\left(M^{\frac{1}{p}}P^{-\frac{3}{2}}\right)$. As one moves to very large numbers of cores, software must become increasingly fine grained in exploiting inter-core dependencies, as τ scales by $\Omega\left(M^{\frac{1-p}{p}}P^{-\frac{3}{2}(1-p)}\right)$. For $p = 0.8$ this is $\Omega\left(M^{0.25}P^{-0.3}\right)$, so increasing by 16x cores without increasing the amount of memory *per core* requires ~0.43x finer-grain τ (average cycles per word of spatial communication). One can imagine that ultimately this could keep happening until $\tau \sim 1$ with very large penalties for moving out of the asymptotic tail. At this point, the only way to constrain I/O bandwidth would be to increase memory.

7.3.3 Alternate Topologies

Interestingly, in the derivation in section 7.3.1, in the asymptotic region there is no dependence on the topology of the many-core system. The internal details of the topology are effectively hidden from external I/O bandwidth provided the memory exceeds the critical threshold. However, this critical threshold depends on the topology, and is related to the temporal length required for each node to be reachable from every other node ($2\psi N$ in the case of the 2D mesh). Whilst, as discussed earlier in section 7.1.1, a more highly connected topology is unlikely to reduce this critical threshold if communication is dominated by 2-D distance traversal costs, a more loosely connected topology can certainly result in an increase of the critical threshold (such as having a 1-D communication topology). Also, a stacked-die solution which is governed by 3-D distance traversal costs would result in a lower critical threshold, and better spatial distance distributions than the 2-D physical case.

7.3.4 Real caches versus Rentian memory

The memory model we used was very ideal in that it optimally minimises the external I/O bandwidth by knowing precisely which items to store on-chip and which to offload for long-term storage. Although scratch pad memories *may* be able to function close to an ideal Rentian memory, most current architectures use caches which operate on a statistical basis. Hartstein et al. [49] observed that the time distribution between cache-line access for large complex workloads behaved as a power-law, and Hartstein et al. derived analytical expressions for the performance of caches [49]. They showed that this explains the well known '$\sqrt{2}$ law' [28] whereby every doubling of cache size reduces the miss-rate only by about $\sqrt{2}$. They derived models for direct, set-associative and fully associative cache models, and validated them on a number of benchmarks with very large datasets to show that it accurately modeled cache scaling behaviour. Among other things, they showed that the asymptotic exponent was largely independent of cache architecture, meaning that the benefits of a direct-mapped versus fully-associative version were a one-off constant in terms of its asymptotic scaling (much like a well implemented algorithm may have a smaller constant factor for its asymptotic scaling than a poorer implementation).

There is an important difference to note between Hartstein's power-law distribution and the Rentian one – while theirs governs cache-lines, the Rentian one governs individual words. Going from one to the other requires an understanding of Address-Space locality. However, we can take two extremes of Address-Space locality, and show that they produce the same asymptotic tail exponent. For one extreme, we can assume no address-locality, so that each word that is used effectively resides on its own cache line. This means that the inter-access distribution of cache-lines would be identical to the inter-access distribution of words. At the other extreme, we can assume that there is full-locality, so that accesses are done as a bundle – where each access of any word in the cache-line, is immediately followed by accesses to the rest of the words in that cache-line. Recall that each word must still be accessed with the Rentian power-law distribution, but what is happening is that the accesses for multiple words are all bundled together in a correlated fashion. This results in a higher frequency of accesses to the same cache-line at small temporal distances, but still results in exactly the same asymptotic power-law exponent for the larger temporal distances. Given both of these extremes have the same power-law exponent for cache-line temporal distances as the Rentian one does, it strongly suggests that this holds in general (although it is not a formal proof).

Another way of understanding Address-Space locality is to pose it as another embedding problem, much in the same way as instructions are embedded, so is data. The ordinary address-space model allows the entire address-space to be accessed randomly with no difference in penalty. That is, there is nothing *special* about the particular address-location – any other location could serve just as well for storing the data, etc... This makes the address-space effectively indistinguishable (here, fully-connected at equal cost) from an access-locality standpoint (even if for convenience we use a linear address space). The cache-line however changes this by effectively adding a fully-connected cluster – the length of the cache-line, where accesses are cheaper. Multiple levels of caching with growing cache-line sizes creates a hierarchy of cost scales. However, because the length of the cache-lines are still finite, the configuration volume available for cheaper-accesses is likewise small, and the asymptotic behaviour remains the same. This is similar to how in the Rentian CMP model above, spatial locality is at first exploited leading to a steeper head distribution, but quickly runs out and leads to exactly the same asymptotic tail exponent.

This leads one to conclude that we would expect the exponent for the cache-line inter-access time distribution to be the same as the word inter-access time distribution, namely $\mu = 2-p$. Using Hartstein's results we would then expect cache-misses to scale according to M^{p-1} which is slightly different from the perfect Rentian model of $M^{p'-1}$ for memory size M. In general, the performance gap of a cache compared to a perfect memory is expected to scale as being $\Theta\left(M^{p-p'}\right)$ times less efficient. Alternatively, the equivalent Rentian memory for a cache memory scales according to $\Theta\left(M^{\frac{p-1}{p'-1}}\right) = \Theta\left(M^{p}\right)$. Another way of stating this is that the efficiency of a cache in minimising cache-misses, compared to an ideal scratchpad memory, scales according to $\Theta\left(M^{p}\right)$, that is a Rentian Memory of only size $\sim M^p$ would do just as well. The analysis here helps give an upper-bound to the performance gap with a scratchpad memory, since no memory can do as well as the ideal Rentian one.

In figure 7.6 and figure 7.7 we can see the effect of both exploiting spatial locality and of Rentian memories versus caches. The top two curves show the effect of not exploiting spatial-locality between cores for both a Rentian memory and cache-based memory, whilst the bottom two curves show how exploiting spatial locality amongst the 1024 cores leads to a significant benefit, but with asymptotically the same slopes as the top two curves. We can see here that for Rentian exponent of $p = 0.8$, leveraging the available spatial-locality is equivalent in benefit to over a thousand-fold increase in on-chip cache memory, in terms of external I/O bandwidth reduction. The cache behaviour shown here is a best-case model (for external I/O bandwidth) of cache-line size equal to one word, so that each word is its own cache-line and is fully-associative. This means that it doesn't fetch any words it doesn't use, and matches the Rentian model at $M = 1$. Depending on address-locality, for larger cache-line sizes of S words, the bandwidth may be offset by a constant multiplicative factor increase of up to S, making it worse still. However, from a cache-line miss perspective rather than a raw word-based I/O bandwidth measure, the address-locality reduces the number of cache-line misses of both the cache and

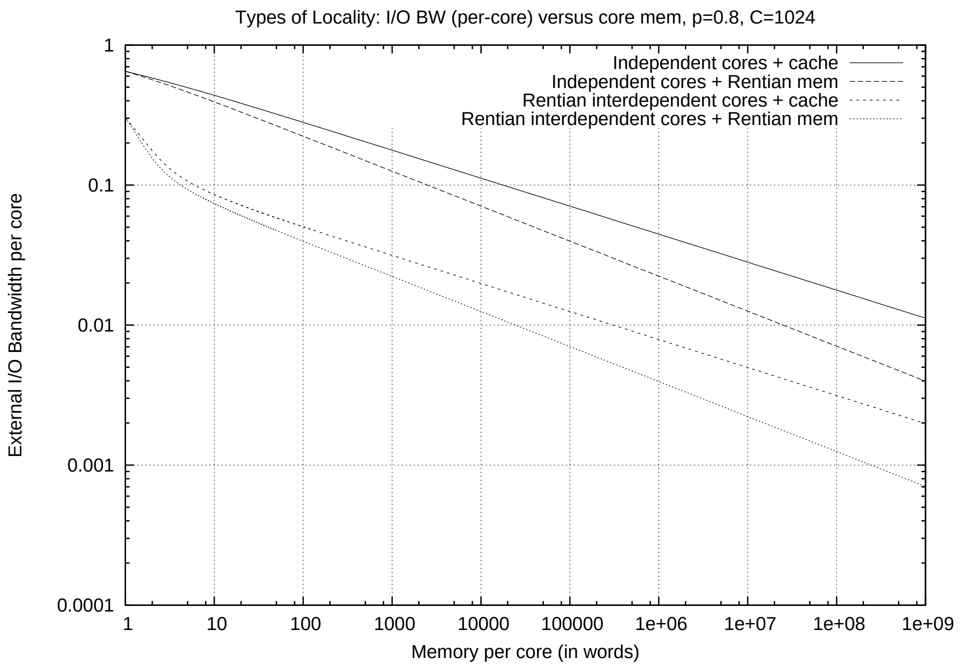


Figure 7.6: Numerical evaluation of model at $\psi = 1$, demonstrating the effect of using Rentian Memory vs. Cache model, and of interdependent vs. independent cores. An I/O bandwidth per-core of one, here, equates to one read/write access per instruction.

the equivalent Rentian model by the same multiplicative factor. Here, the equivalent Rentian model would optimally embed entire *cache-lines* of data instead of *words* of data. What this means is that regardless of cache-line size, the cache model has absolutely poorer performance and asymptotically poorer scaling than the Rentian model in both external I/O bandwidth and in cache-line misses. We also note in figure 7.7 how in the head region (at lower memory sizes), the I/O bandwidth for Rentian interdependent cores matches that of independent cores. This is because they are not exploiting spatial locality in this region, as it is too infrequent/costly, and hence behave like independent cores effectively communicating via off-chip memory. Here, caches are modelled as being equivalent in performance to correspondingly smaller Rentian memories of size M^p. So, in the asymptotic tail region, we see that the transition region for caches, where spatial locality is exploited, occurs later than for the ideal Rentian memories due to this relative inefficiency of the caches.

7.3.5 Area & bandwidth constrained many-core scaling

Supposing CMPs are in the external I/O bandwidth-limited domain, one can ask how best to utilise the doubling of transistors every process generation for performance. For a fixed B_{tot}, one would need to trade off according to:

$$M \propto P^{\frac{p'}{1-p'}},$$

for an idealised Rentian memory of M per core.

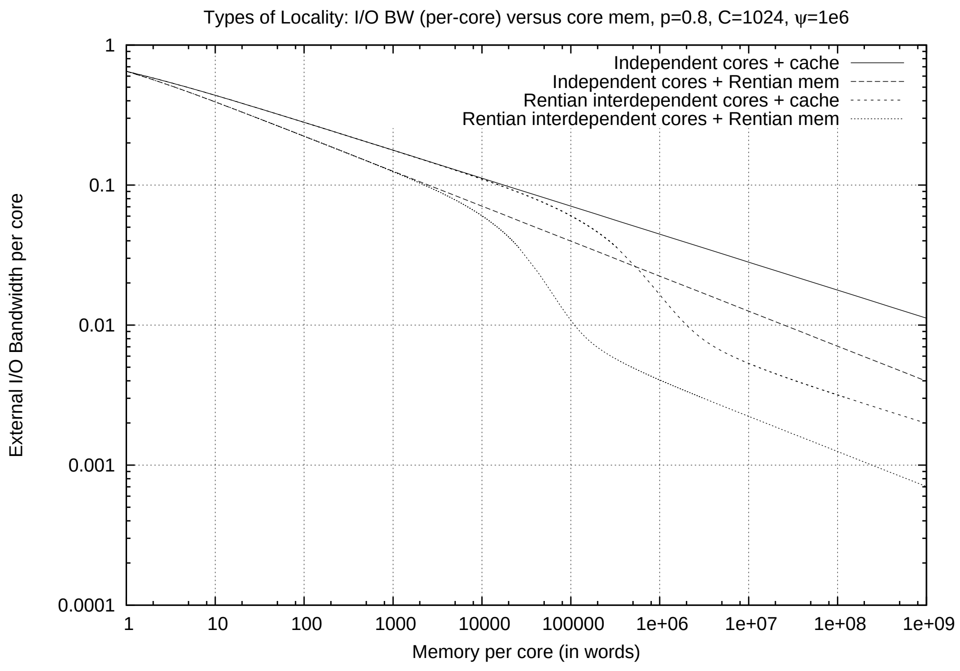


Figure 7.7: Numerical evaluation of model at $\psi = 10^6$, demonstrating the effect of using Rentian Memory vs. Cache model, and of interdependent vs. independent cores. An I/O bandwidth per-core of one, here, equates to one read/write access per instruction.

This means for an example of $p' = 0.75$, upon doubling the number of cores, the memory *per core* would need to increase eight-fold in order to maintain the same total external I/O bandwidth, resulting in a sixteen-fold total increase in on-chip memory. Thankfully, with technology scaling, the bandwidth available is also growing (see Chapter 2), albeit more slowly. Nonetheless, the power consumption to support this growth in bandwidth could be crippling. Here we explore the cores/memory tradeoff for a limited-growth in bandwidth with scaling.

Let ρ be the amortised area for each memory word, and let σ be the logic area for each core (including amortised routing networks). Let M be the memory-per-core, P the number of cores, and A the total area. Then we model total area A as:

$$A = \rho M P + \sigma P$$

Let us start with values A_0, M_0 and P_0 where this is a base case that is known to hold. Let us then define multipliers:

$$\begin{aligned} a &\equiv A/A_0 \\ m &\equiv M/M_0 \\ c &\equiv P/P_0 \end{aligned}$$

This gives us:

$$\begin{aligned}
aA_0 &= mc\rho M_0 P_0 + c\sigma P_0 \\
a\rho M_0 P_0 + a\sigma P_0 &= mc\rho M_0 P_0 + c\sigma P_0 \\
(a - mc)\,\rho M_0 &= \sigma\,(c - a) \\
\frac{c-a}{a-mc} &= \frac{\rho M_0}{\sigma}
\end{aligned}$$

Let $0 < \gamma < 1$ be the current proportion of area that is core logic, then:

$$\gamma = \frac{\sigma}{\rho M_0 + \sigma}$$

and

$$\begin{aligned}
\frac{c-a}{a-mc} &= \frac{1}{\gamma} - 1 \\
\frac{c-a+a-mc}{a-mc} &= \frac{1}{\gamma} \\
\frac{a-mc}{c-mc} &= \gamma
\end{aligned}$$

Suppose the total I/O bandwidth scales by a^{β}, then we have:

$$a^{\beta} = m^{p'-1}c^{p'}$$

Given a and γ we can then numerically solve for m and c. For the asymptotic behaviour we have:

$$\beta \log a = (p' - 1)\log m + p' \log c$$

and:

$$a = c\,((1-\gamma)\,m + \gamma)$$

thus, assuming m grows with a, for large a we have:

$$\log a \approx \log c + \log m + \log(1-\gamma) \tag{7.6}$$

Substituting for m yields:

$$\log c \approx (1 - p' + \beta)\log a - (1 - p')\log(1-\gamma)$$

i.e.

$$c \propto\!\sim a^{1-p'+\beta}$$

and

$$m \propto\!\sim a^{p'-\beta}$$

To satisfy the assumption that m is growing with a we require that $\beta < p'$. This then relates the growth in the number of cores to the area given the spatial Rentian scaling and external I/O bandwidth scaling factors. We notice that when bandwidth growth can keep up with the spatial Rent's exponent, then the number of cores scales linearly with area, as expected. A numerical evaluation shows reasonably close agreement - for example with $p' = 0.75$ and $\beta = 0.6$, the scaling exponent over a range of 10^6 cores is 0.861 versus an expected 0.85.

Using the high-end ITRS bandwidth growth[1], we have $\beta = 0.67$ and choosing $p' = 0.75$ means that for every doubling of transistors, we expect cores to grow by 1.9x and memory to grow by 1.06x, indicating that locality allows good scaling. For higher values of p', where it approaches unity (little to no locality), the number of cores grows by 1.6x and Rentian memory by 1.26x. This is more in line with the ITRS predicted doubling of cores every two processor generations.

If we replace the ideal Rentian memory with caches instead we substitute the memory scaling exponent from p' to p:

$$\beta \log a = (p-1)\log m + p' \log c$$

Substituting for m into eqn 7.6 yields:

$$\log c \approx \frac{(1-p+\beta)}{p'-p+1}\log a - \frac{(1-p')}{p'-p+1}\log(1-\gamma)$$

$$c \propto\!\sim a^{(1-p+\beta)/(1-p+p')}$$

and:

$$m \propto a^{(p'-\beta)/(1-p+p')}$$

For $p' = 0.75$, then $p = 0.8$ yielding cores growing by 1.8x (instead of 1.9x) and cache per core (amortised) grows by approximately the same 1.06x. As p' approaches unity,

[1]See Chapter 2, figure (2.2)

so does p, therefore the scaling for poor-locality algorithms is the same for caches as it is for Rentian memory, namely:

$$c \propto a^{\beta}$$

and:

$$m \propto a^{1-\beta}$$

In this case, the number of cores can only grow if there is an increase in external I/O bandwidth. This means that applications that have large working sets, but are written to exploit parallelism through data independence rather than data interdependence can, ironically enough, only run on fewer cores as they are constrained by I/O bandwidth. Algorithms that employ large data interdependence amongst cores, on the other hand, may be more complicated but allow for better scaling of the number of cores, for the given I/O bandwidth constraints.

GPUs have small working sets and high data parallelism, but at the same time, the I/O bandwidth grows approximately linearly with raw computational power. High-end desktop single-GPU cards already consume almost 400W at peak, it is not clear how much further they can keep pushing the thermal envelope. Given the growing disparity between communication energy and computational energy, if computational performance requires an almost linear increase in communication, it implies that future GPUs will proportionally spend growing amounts of their energy budget on communication, with computational performance scaling poorly to maintain their thermal profile.

7.3.6 Shortcomings of the model

An important shortcoming of the spatio-temporal Rentian model is that it only covers data locality, and does not model instruction locality and the bandwidth needed to feed it instructions. Indeed to maximally exploit data locality, there is an implicit assumption that instructions can be embedded anywhere. One can then ask – if instructions can be embedded anywhere, does that mean the movement of instructions themselves is uniform or non-local? Thankfully, this is not necessarily the case. A way to tackle this question, however, is to consider coarse-grain multi-scale embedding, whereby the embedding problem is hierarchical – occurring at multiple scales, as it does in VLSI. This means that the instruction information would only need to move at the level of its hierarchy, which can preserve asymptotic Rentian scaling of I/O bandwidth for instructions as well. Indeed, this type of embedding approach is discussed next, in Chapter 8.

7.4 Conclusion

This chapter looked at the implications of Rentian software as applied to a CMP model. It uses a spatio-temporal Rentian model for the analysis, that allows an examination of how external I/O bandwidth is affected by the number of cores, their exploitation of spatial locality, the temporal coarseness of inter-core communication, the quantity of on-chip idealised-memory per core as well as expected behaviour for caches. For a doubling of transistor area as per Moore's law, it also explored how to balance the tradeoff between on-chip memory and cores in an I/O-limited scaling model. An idealised on-chip memory was introduced, called Rentian Memory, that is 'perfect' in the sense that it minimises the external I/O bandwidth of misses, was still shown to exhibit the power-law scaling 'miss' behaviour of caches, albeit with a better scaling exponent.

Based on this spatio-temporal Rentian model, it is possible to project the scalability requirements for future CMP applications. They should:

- exploit parallelism through data interdependence rather than data independence
- accomplish their goal with algorithms that have lower complexity of communication – i.e. lower Rent's exponents (e.g. using wavelets if possible, instead of large FFTs)
- ultimately, need to become fine-grained in their communication between cores

From a hardware standpoint, this means that architectures which support such exploitation of locality, such as Tilera and Picochip will better support Rentian scaling as the number of cores grows, than those of the current x86 many-core architectures from Intel or AMD.

Chapter Eight

8

Effect of Locality on Asymptotic Complexity

This chapter covers the analysis of asymptotic communication costs in algorithms. Indeed, it is shown how these communication costs can dominate over traditional computational complexity. The direct calculation of Rentian parameters is demonstrated on some example algorithms using first-principles.

The embedding problem is discussed, with an example showing how optimal embeddings do not compose to form another optimal embedding. However, it is then shown that optimal embeddings don't need to compose in order to achieve asymptotically Rentian scaling. Hierarchical composition is introduced, instead, demonstrating that random or worst-case links at each level of hierarchy still preserves Rentian scaling, but with a constant factor overhead. This makes Rentian embeddings practically composable for a CMP-system, at a constant-factor cost of optimality.

Finally, we consider the costs of asymptotic costs of accessing memory. As data has a physical position, in order to write to or retrieve data, there is always some implicit spatial communication involved. Lower bounds are shown for this cost, as it relates to the Rentian exponent – and under certain conditions, the memory access cost can be $O(1)$. As an example, it is shown how this affects the cost-analysis of very large dense matrix multiplications.

8.1 Direct Rentian analysis of algorithms

Although Rent's rule was observed in the 1950s, it has since then predominantly been applied to the analysis of large existing VLSI graphs, and experimentally determined by partitioning and placement. Little work has been done in analysing Rentian behaviour directly on structures or algorithms. This is perhaps a little surprising, however it is unlikely to have been of practical value in the VLSI domain, as final VLSI graphs are easily analysed by partitioning and placement. In moving our analysis to software, however, the first-principles analysis of algorithms becomes important in determining their communication scaling properties, as the actual computation and communication graph is generated as the software runs. We give some sample *ab-initio* characterisations below.

8.1.1 Scaling behaviour of trees

For any k-ary tree structure, the minimal terminal scaling law is given by:

$$T = \Theta\left(G^{1}\right)$$

i.e. the Rent's exponent $p = 1$ with unbounded dimensionality. Where G is the number of nodes in a region, and T is the number of edges connecting nodes from inside the region to nodes outside the region.

Proof. We show that for any connected set of G interior nodes:

$$T = (k-1)\,G + 2$$

By induction: we start with one node that has $(k+1)$ edges consisting of one parent edge and k child edges, satisfies the case of $G = 1$. For any connected set S of $G = n$ nodes, connecting another node X results in the connecting edge being subtracted from both S and X, and leaves k dangling edges from X, resulting in an increase of edges by $(k-1)$. This satisfies the case of $G = n + 1$. Since any connected set of nodes can be constructed starting from the base case, by incrementally adding nodes, this completes the proof. □

8.1.2 Iterative Fibonacci sequence

The iterative Fibonacci algorithm has the dependence:

$$f_k = f_{k-1} + f_{k-2}$$

The interior computation has a recurrence pattern that consists of two input edges $(f_{k-2},\, f_{k-1})$ and two output edges $(f_{k-1},\, f_k)$. Any contiguous chain of this recurrence pattern results in:

$$T = 4$$

i.e.

$$p = 0$$

This yields a dimensionality of one, which is also trivially observable from the simple dependence chain.

We note that a recursive form of the Fibonacci calculation (without re-use of calculated results) would result in a binary tree, and thus have Rent's exponent $p = 1$.

8.1.3 Matrix multiplication

The $N \times N$ matrix multiplication algorithm, denoted $MM\left(N \times N\right)$, can be decomposed into eight $MM\left(\frac{N}{2} \times \frac{N}{2}\right)$ multiplications, which then involve point-additions at that level of hierarchy.

The matrix multiplication has $\sim 2N^3$ operations (multiplication and addition) and $3N^2$ terminals (inputs and output). This leads to:

$$G = 2N^3$$

$$T = 3N^2$$

implying that:

$$T = \frac{3}{2^{2/3}} G^{2/3}$$

So that:

$$p = \frac{2}{3}$$

and

$$dim = 3$$

This implies that communication for matrix multiplication is $O\left(1\right)$ provided it is embedded in a growing three-dimensional substrate. Indeed, we know that systolic arrays with a 2D topology, plus one dimension of time, works for this [22, 67]. Cannon utilised a 2-D torus topology, but noted that a 2-D mesh is also easily capable of simulating a 2-D toroid version with constant penalty. Leiserson considered a 2-D hexagonal [67] topology for streaming the matrix multiplication from the edges of the systolic array. In general, however, the matrices may already reside in on-chip memory as part of a larger computation.

Similarly, LU decomposition and other common matrix operations are known for systolic arrays with $O\left(1\right)$ link length and $O\left(N\right)$ steps [67], and also yield a simple exponent of $p = \frac{2}{3}$.

8.1.4 Handling memory accesses

We should note that the count of terminals is based on the number of *unique* edges coming out of a boundary, connected to instructions within the boundary. For example, two blocks of code may access the same memory location. Both accesses result in spatio-temporal interconnect – however, when considering the combined two blocks (at larger instruction counts), these contribute only one terminal at that boundary. For random accesses, then, the distribution of accesses can also determine how many terminals are at the boundary of combined random accesses. Thus the distribution and locality of memory accesses directly affects the count of terminals, and their Rentian scaling – which is precisely what we want. For data-driven memory access patterns, this means that the Rent's exponent can vary not only based on the underlying algorithm, but also on how the data drives the algorithm. For example, an arithmetic or Huffman compression algorithm generates and accesses entries in a data-structure based on the input data stream, and hence the scaling behaviour of Terminals to Instructions depends on the characteristics of the data-stream being compressed, as well as the algorithm.

8.1.5 Fourier transforms

The interior scaling behaviour of a Fourier transform involves a hierarchy of butterflies. If we compose based on a sub-tree of butterflies, we have for a butterfly of width N, a depth of $\log_2 N$. Henceforth we denote $\log N$ to mean $\log_2 N$. Thus:

$$G = N \log N$$

$$T = 2N$$

if:

$$\log T = p \log G + \kappa$$

for $N = 2$, we have $G = 2$ and $T = 4$, so $\kappa = 2 - p_2$, where p_2 is the value of p for $N = 2$.

$$\log N + 1 = p\,(\log N + \log\log N) + 2 - p_2$$

$$p = \frac{\log N - 1 + p_2}{\log N + \log\log N}$$

As

$$\begin{aligned} dim &= \frac{1}{1-p} \\ &= \frac{\log N - 1 + p_2}{\log\log N - 1 + p_2} \\ &= \Theta\left(\frac{\log N}{\log\log N}\right) \end{aligned}$$

We note that this compares well with the trivial lower bound of the dilation λ, given by the ratio of graph diameters. We note that the FFT comprises $N \log N$ nodes with a graph diameter of $2 \log N$, thus:

$$\lambda = \frac{(N \log N)^{1/dim}}{2 \log N}$$

Keeping the desired dilation λ constant and taking logarithms results in:

$$\begin{aligned} \log \lambda &= \frac{1}{dim}(\log N + \log\log N) - \log\log N - 1 \\ dim &= \frac{\log N + \log\log N}{\log \lambda + \log\log N + 1} \\ &= \Theta\left(\frac{\log N}{\log\log N}\right) \end{aligned}$$

8.1.6 Related work

The closest work was by Fox [43] regarding analytical derivations of the communication overhead in high-performance parallel computing domain. He showed that the cost of matrix multiplication (on a mesh) is $N^{3/2}$. There are two crucial differences between his work and this. First of all, he considers the 2-D domain of the data, rather than the spatio-temporal domain of the computational data-dependency graph. Incidentally, he also uses the term ‘space-time’, however in his definition space refers to the *data* itself rather than actual physical space. Secondly, his cost metric for communication is based on the number of transactions crossing the boundary, rather than the physical distance it must travel. This certainly makes sense for a parallel computing environment, although it does not for CMP. Also, he does not consider a hierarchical decomposition, although he notes that in VLSI, that Rent’s rule implies that the system is self-similar across scales or ‘grain size’ as he terms it, and discusses the behaviour upon varying the ‘grain size’. For these reasons, his dimensionality for matrix multiplication is two, rather than three as in here.

Fox also considers embeddings of the FFT onto the hypercube (rather than the 2-D mesh). From his analysis of communication versus computation in the hypercube topology, and as the hypercube has dimensionality of $\log_2 P$ for P processors, he suggests

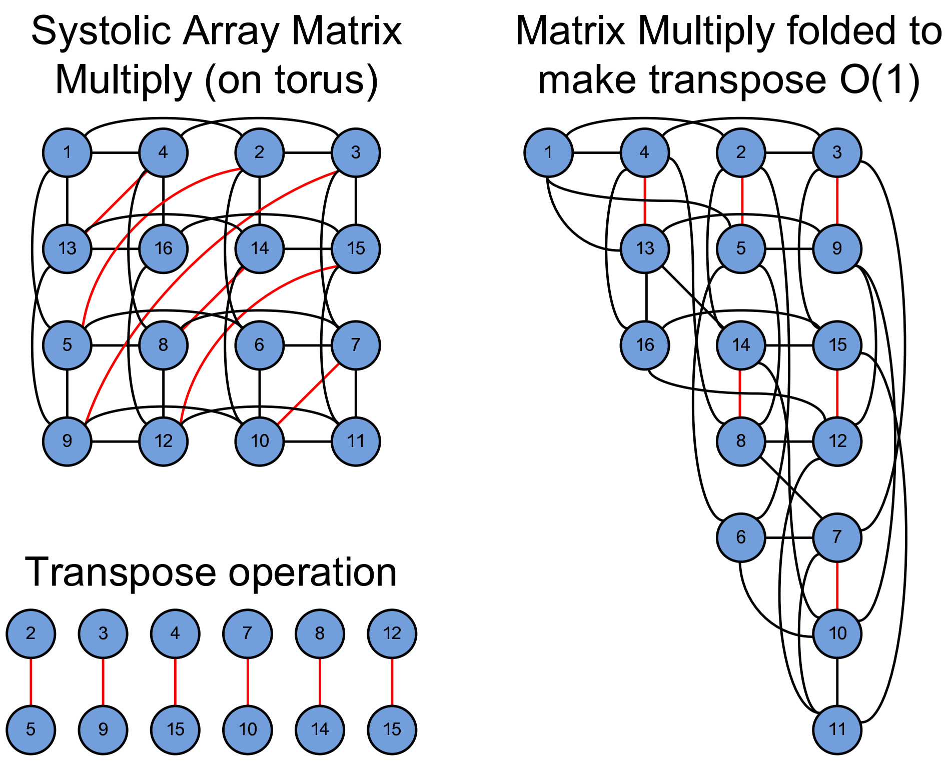

Systolic Array Matrix Multiply (on torus)
Matrix Multiply folded to make transpose O(1)
Transpose operation

Figure 8.1: Optimal embeddings don’t compose. An optimal *spatial* embedding of $N \times N$ square matrix multiplication leads to a systolic array with torus topology, however optimal embedding for the transpose operation has different needs. Transpose *within* the torus results in non-local links of Manhattan-metric length $2(N-1)$, thus growing as $O(N)$. The matrix multiply can be folded along the diagonal to produce an $O(1)$ cost transpose operation, with further $O(1)$ dilation of matrix multiply links. (It can even be folded again to reform a square placement.) However as there are N^3 communication operations using these links, this can actually make total cost worse.

that the N-point FFT has an information dimension of $O(\log_2 N)$. The Rentian analysis shows that it is actually a little less than this.

8.2 The ‘embedding problem’

Real VLSI designs have the benefit of a near-complete network description before tools try to optimally embed it into silicon. Although designers partition their designs into modules and blocks, for the most part, the placement tools have abundant freedom to shuffle gates around within a region. This means there is a large amount of implicit information in the placement. For example, for N gates with N positions, there are $N!$ possible configurations, of which the VLSI embedding is a carefully chosen one. Instructions in software also have implicit position (their instruction address), however, in terms of optimising their spatio-temporal embedding, there is less freedom to move these instructions around. The VLSI equivalent would be a design consisting of many so called *hard-IP* blocks, which have fixed placement already within the block.

Algorithms and functions are designed in a modular fashion, and they should compose easily, without specific placement optimisations needed that 'bleed' across from one algorithm to the next one. For example, let us take matrix multiplication, followed by either another matrix multiply or by a matrix transposition. The square matrix multiplication $Z = XY$ has a very simple, optimal embedding solution across multiple cores given by Cannon [22] and seen in figure 8.1. This consists of only neighbour communication between nodes in a 2-D toroid (shown as black links) where the row terms of X are passed along rows, and the column terms of Y passed down columns. We see here how a 2-D mesh simulates a 2-D toroid with only a constant factor penalty, by folding and interleaving the torus into the 2D mesh. The transpose operation is essentially a communication operation (here shown as red links). If each node contains a sub-matrix, it merely needs to perform the transpose operation on that sub-matrix, and then swap the result with its corresponding transpose node's. There is a trivial embedding for transposition (seen at the bottom of figure 8.1), which simply places each node next to its transpose partner, resulting in only neighbour communication as they swap them. The *'embedding problem'* is that these two parts of the algorithm have very different embeddings, and optimal communication in both does not result in optimal communication across them. That is, their embeddings do not compose optimally. We can see that in the $N \times N$ matrix multiply, that the transposition operation (seen in red) results in non-local communication. In general, the worst-case link distance grows by $O(S)$ for sidelength $S \leq N$ of the computation. In contrast, if we were to follow the matrix multiplication with another matrix multiplication, for example in exponentiation, then their embeddings can match, and there is $O(1)$ local communication from one matrix multiplication to the next.

One may instead wish to construct a combined 'matrix-multiply and transpose' operation, by optimally embedding the combined networks. One approach for this is to spatially fold the matrix multiplication so that transposition is also a local operation. This is seen on the right in figure 8.1. We note that now the black links for matrix multiplication are longer, however it can be shown that these grow by an average constant factor $\kappa = \frac{3}{2}$, and so are still $O(1)$ regardless of the sidelength S of the computation. Noting the potential inconvenience of a triangular embedding, it is also possible to fold the corners again (almost like origami) so that it reforms a square profile. Again, this maintains $O(1)$ asymptotic local communication. However, there are very different costs involved in the two algorithms. The matrix multiply consists of N^3 communication operations, and so the total cost grows to $\sim 2\kappa N^3$ (where the factor of 2 is due to the torus topology). The transposition operation has communication cost $\sim N$ compared to the operation on the regular torus topology of $\sim SN^2$. So the total communication cost has changed from $(2N^3 + SN^2)$ to $(3N^3 + N)$. However, we know that $S \leq N$, and typically S can be much smaller than N, thus folding the topology to make transposition local, actually increases total communication costs. Indeed, the non-locality of the matrix transpose doesn't actually go away simply because another $O(1)$ embedding is produced, but rather, it causes the matrix multiplication to have a less efficient embedding that makes total costs increase.

We should note that the diagonal symmetry property of transposition on matrices is a particularly special property that was employed in this example to combine the embeddings. In general, when composing embeddings from two algorithms, there is no such property that can be leveraged. What the above example shows, however, is that even when such a special property *can* be employed to make a combined embedding, it can actually make matters worse. Moreover, in practise, we would like the embeddings themselves to compose easily – that is we would like to describe an embedding for an algorithm once, without needing special cases for combinations with other algorithms such as matrix-multiply with transpose.

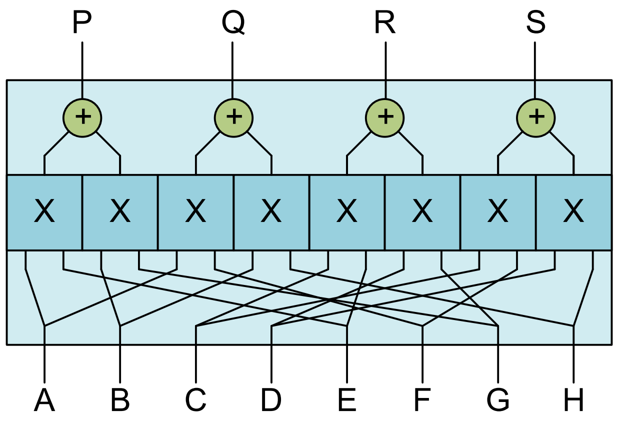
As the embeddings of two algorithms will generally compose to produce non-local communication between them, it suggests that such non-local communication is practically unavoidable and that perhaps Rentian scaling is unlikely to emerge. As we shall see next, this is not actually the case.

8.2.1 Solution: hierarchical composition/decomposition

It is important to remember that Rentian locality does not mean that there are *no* long interconnects. Instead, Rentian locality only *constrains* the amount of non-local interconnect. Moreover, Rentian scaling is governed by the Rent’s exponent as an asymptotic behaviour – small constant factors in cost are less important than the power-law behaviour itself.

What this means is that optimal embedding is not a prerequisite to Rentian scaling. Indeed, as we have seen with composition of matrix-multiplication and transposition, combining embeddings to minimise communication costs may not actually confer real or significant advantages. Instead, we need to manage the amount of non-locality. A clue to solving the *embedding problem* lies with Donath’s original derivation of wire-length distributions and average length. Donath’s original predictions for wirelengths turned out to be correct in terms of its power-law exponent, but was off by a constant-term factor of about two. The reason is that Donath considered Rent’s rule in a hierarchically partitioned manner, and assumed *non-local random* connectivity, to connect sub-blocks at each scale of the hierarchy. What is important to note is that this is still actually consistent with Rentian scaling and only resulted in a constant factor increase in the wire-length distribution, preserving the distribution’s power-law exponent, and preserving Rentian scaling properties. This means that Rentian scaling can accommodate random non-local connections, with only a constant factor penalty, provided the non-locality is constrained to that physical scale of the hierarchy.

This allows us to approach the embedding problem with a divide-and-conquer strategy loosely based on Donath’s original derivation, adapted for characteristics of the software domain versus VLSI. We do this by hierarchically decomposing the spatio-temporal graph of an algorithm, with constraints on the amount of merging code allowed at each



$$\begin{bmatrix} P & Q \\ R & S \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

Figure 8.2: Decomposition of Matrix Multiply function into smaller Matrix Multiplies (denoted by X). Merge logic (additions), and sub-partitions can be placed randomly within each level of the hierarchy. Spatio-temporal interconnect may be non-local, but the non-locality is constrained to the physical domain of that partition.

level. Donath did not have to deal with the presence of merging code, but only with spatial partitionings.

Let us then quantify the asymptotic cost of communication for a hierarchically decomposable algorithm. We partition the parent problem into *s* child sub-problems with problem-size reduction *r*. For example, an $N \times N$ square matrix multiply can be decomposed into eight matrix multiplies of size $\frac{N}{2} \times \frac{N}{2}$, so $s = 8$ and $r = 2$. This can be seen in figure 8.2, where the matrix multiplication is split into four quadrants which are calculated separately (in eight blue boxes) and then merged together (in green). We can also see the physical consequence of this partitioning on a CMP in figure 8.3. Here, we can see again the eight sub-matrix multiplications arranged in both time and space, along with the thin merge logic. Another level of decomposition is also shown in the figure consisting of smaller matrix multiplies and corresponding merge logic. We should note that although *decomposition* is discussed here and below, the same argument can also be made for self-similar *composition* as well.

Let $G(N)$ be the number of instructions (gates) required to accomplish the computation. This is the traditional 'computational complexity' of the problem. For many algorithms we have asymptotically that this is $\Theta(N^\gamma)$, and for these we can define:

$$G(N) = gN^\gamma$$

for some constant g. In the case of matrix multiplication $\gamma = 3$. Note that $\gamma = \frac{\log s}{\log r}$ for Rentian partitioning, as the bulk of the instructions should be in the s children of problem size $\frac{N}{r}$.

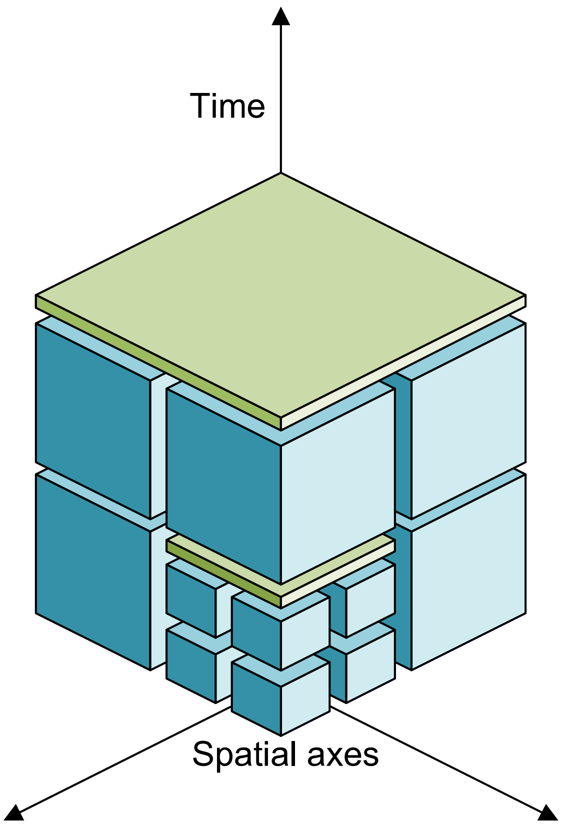


Figure 8.3: Hierarchical decomposition of an algorithm (matrix multiply) into space and time. Green indicates merge logic, blue indicates a lower-level of the hierarchy. An additional level of decomposition is shown for one blue box.

The child sub-solutions may need further merging computation to form the parent solution. With Rentian scaling, it necessarily also involves interconnect at that level as otherwise there would be too many terminals at the parent boundary. The number of merge instructions is: $G(N) - sG\left(\frac{N}{r}\right)$. Let us assume the number of I/Os for problem size N follows Rent's rule:

$$T(N) \equiv k\left(G(N)\right)^{p}$$

For matrix multiplication we have two inputs of size N^2 and one output of size N^2, so $p = \frac{2}{3}$ (given $\gamma = 3$) and $k = 3g^{-p}$.

There are $sT\left(\frac{N}{r}\right)$ terminals due to sub-problems that need to be wired up, and we will assume that asymptotically there are at most $mT(N)$ internal and external terminals (inputs and outputs) of merge instructions, for some constant $m \geq 0$. This also ensures an asymptotically small contribution of merge instructions compared to the childrens' instruction count. In the case of matrix-multiply $m = 1$.

These terminals need to be wired up to each other and to the parent terminals. Following Donath, given an average fanout for the region, then the average wiring per terminal is $\alpha = \frac{Fanout}{1+Fanout}$ which is a minimum of $1/2$ for simple edges (one source and one destination), and is at most one (for high-fanout), regardless of the connectivity. So this factor is tightly constrained as $\frac{1}{2} \leq \alpha < 1$, and so does not contribute to the asymptotic complexity terms. Then the total number of interconnects is given by:

$$I(N) \equiv \alpha\left((1+m)\,T(N) + sT\left(\frac{N}{r}\right)\right)$$

The average cost of interconnect at this level of hierarchy is something that can be directly estimated given a cost function and a distribution of distances. In the simple case of cost being linearly proportional to distance, and for uniform-random links or worst-case links, the cost is simply proportional to the side-length of the volume of instructions. So for a d-dimensional spatio-temporal embedding, we would expect the average cost of interconnect to be proportional to:

$$c(N) \propto (G(N))^{1/d}$$

However, this is only up to the spatio-temporal size of the system with P cores with volume $V_{crit} \propto P$. Thereafter, the embedding volume growth is one-dimensional, so that for a Manhattan metric let us define:

$$c(N) = \begin{cases} (G(N))^{1/d} & , G(N) \leq \zeta P \\ (\zeta P)^{1/d} + \frac{G(N)}{\zeta P} - 1 & , G(N) > \zeta P \end{cases}$$

In general, however, we can calculate the total communication cost $C(N)$ of the algorithm as being the sum of the costs of the sub-problems, plus the cost of interconnects at that level of the hierarchy so that:

$$\begin{aligned} C(N) &= sC\left(\frac{N}{r}\right) + I(N)\,c(N) \\ &= sC\left(\frac{N}{r}\right) + \alpha\left((1+m)\,T(N) + sT\left(\frac{N}{r}\right)\right)c(N) \end{aligned}$$

In the region $G(N) \leq \zeta P$ we have:

$$C(N) = sC\left(\frac{N}{r}\right) + \alpha\left((1+m)\,T(N) + sT\left(\frac{N}{r}\right)\right)(G(N))^{1/d} \tag{8.1}$$

We can hierarchically decompose until we reach a base case with H levels of hierarchy so that $r^H = N$, and $s^H = N^{\log s/\log r}$:

$$\begin{aligned}
C\left(N\right) &= \sum_{h=0}^{H-1} s^h \alpha k \left((1+m)\left(G\left(\frac{N}{r^h}\right)\right)^p + s\left(G\left(\frac{N}{r^{h+1}}\right)\right)^p \right)\left(G\left(\frac{N}{r^h}\right)\right)^{1/d} \\
&= \sum_{h'=1}^{H} s^{H-h'} \alpha k \left((1+m)\left(G\left(r^{h'}\right)\right)^p + s\left(G\left(r^{h'-1}\right)\right)^p \right)\left(G\left(r^{h'}\right)\right)^{1/d} \\
&= \alpha k s^H \sum_{h'=1}^{H} s^{-h'} \left((1+m)\left(g r^{\gamma h'}\right)^p + s\left(g r^{\gamma(h'-1)}\right)^p \right)\left(g r^{\gamma h'}\right)^{1/d} \\
&= \alpha k s^H g^{p+1/d} \sum_{h'=1}^{H} s^{-h'} r^{\gamma(p+1/d)h'} \left((1+m) + s r^{-\gamma p}\right) \\
&= \alpha k s^H g^{p+1/d} \left((1+m) + s r^{-\gamma p}\right) \sum_{h'=1}^{H} \left(s^{-1} r^{\gamma(p+1/d)}\right)^{h'} \\
&= \alpha k s^H g^{p+1/d} \left((1+m) + s r^{-\gamma p}\right) \left(\frac{\left(s^{-1} r^{\gamma(p+1/d)}\right)^{H+1} - \left(s^{-1} r^{\gamma(p+1/d)}\right)}{\left(s^{-1} r^{\gamma(p+1/d)}\right) - 1} \right) \\
&= \alpha k g^{p+1/d} \left((1+m) + s r^{-\gamma p}\right) \left(\left(N^{\gamma(p+1/d)} - N^{\log s/\log r}\right) \frac{r^{\gamma(p+1/d)}}{r^{\gamma(p+1/d)} - s} \right)
\end{aligned}$$

For $p > 1 - 1/d$ and $G\left(N\right) \leq \zeta P$, we then have:

$$C\left(N\right) = \Theta\left(\left(G\left(N\right)\right)^{p+1/d}\right)$$

and this has average spatio-temporal cost of:

$$\Theta\left(\left(G\left(N\right)\right)^{p+1/d-1}\right)$$

This asymptotic scaling behaviour is precisely what we would expect with optimally-embedded Rentian scaling. We should also point out that the asymptotic total communication cost here can do no better than the computational complexity, and is typically *worse*. This also makes intuitive sense if we consider that the computational complexity essentially counts instructions, whereas the communication cost counts the distanced-weighted edges between instructions (which at a minimum grows proportionally to the number of instructions).

For cost at $G\left(N_{crit}\right) = \zeta P$ we have:

$$C\left(N_{crit}\right) = \alpha k g^{p+1/d} \left((1+m) + s r^{-\gamma p}\right) \left(\left(\left(\frac{\zeta P}{g}\right)^{(p+1/d)} - \left(\frac{\zeta P}{g}\right)^{\frac{\log s}{\gamma \log r}} \right) \frac{r^{\gamma(p+1/d)}}{r^{\gamma(p+1/d)} - s} \right)$$

For the region $G\left(N\right) > \zeta P$ we have:

$$\begin{aligned} C(N) &= C(N_{crit}) + \frac{\alpha k}{\zeta P} s^H \sum_{h'=H_{crit}+1}^{H} s^{-h'} \left((1+m) \left(G\left(r^{h'}\right)\right)^p + s \left(G\left(r^{h'-1}\right)\right)^p \right) \times \\ & \left(G\left(r^{h'}\right) + (\zeta P)^{1+1/d} - \zeta P \right) \end{aligned}$$

With asymptotic total spatio-temporal interconnect cost as $G(N) \gg \zeta P$ of:

$$\Theta\left(\frac{(G(N))^{p+1}}{P}\right)$$

and an average spatio-temporal cost of:

$$\Theta\left(\frac{(G(N))^{p}}{P}\right)$$

In the case of matrix multiply, we see that as it grows, it can no longer fit into the fixed number of cores and $d = 1$ (temporal dimension only). We then have for the matrix multiplication problem, an average interconnection cost at large scales of $\Theta\left(N^{3\left(\frac{2}{3}+1-1\right)}/P\right) = \Theta\left(\frac{N^2}{P}\right)$ and total spatio-temporal cost of $\Theta\left(\frac{N^5}{P}\right)$. This total spatio-temporal cost can be interpreted as the total communication energy consumption for the algorithm (in both spatial communication and memory storage costs).

8.2.2 Interdependent composition/decomposition

Although in the preceding example of matrix multiplication, we decomposed the problem into independent sub-problems, we can also choose to decompose into *interdependent* sub-problems with communication occurring between them throughout their execution. There is no assumption made in the derivation that requires independence, but it merely makes the analysis simpler.

For example, we can hierarchically decompose a systolic-array implementation of the matrix-multiplication problem. This time we decompose into eight sub-problems with spatio-temporal volume $G(N) = 2N^3$, with dimensions $2N \times N \times N$ where the factor of two accounts for a single add and a single multiply per systolic array node. There are in general $T(N) = 8N^2$ terminals for the sub-problems: $3N^2$ data inputs including the previous partial solution, $3N^2$ data outputs including the current partial solution, and $2N^2$ items being shuffled to and from neighbours. There are no merge instructions, as merging additions take place within the systolic construct, so $m = 0$. Plugging these into eqn 8.1 now results in exactly the same asymptotic scaling behaviour, but with interdependent partitions, instead of independent partitions into sub-problems.

We should note that in some ways, this decomposition analysis of computation and communication has some rudimentary similarity to the D-BSP and Multi-BSP models discussed in Chapter 3, as there is a recursive decomposition of the algorithm into localised regions in those as well. However, in actuality there are many fundamental differences. Firstly, the decomposition occurs in a hierarchy of space-*and*-time, rather than with the supersteps in the space of processors. The decomposition here, is also not necessarily a binary one, and crucially, unlike the D-BSP and Multi-BSP models, this decomposition fully permits communication at all times across the decomposition boundary – as long as they are counted as 'terminals'. So whereas the systolic-array implementation of Matrix Multiply cannot be analysed in the D-BSP and Multi-BSP models, it can be analysed here.

8.2.3 Analysing non-Rentian algorithms: FFT

We can also apply this to non-Rentian systems, such as the FFT. By partitioning the algorithm into independent sub-FFTs and plugging into eqn 8.1, we get altogether different behaviour, with an asymptotic cost of $\Theta\left(N^{1+1/d}\left(\log N\right)^{1/d}\right)$. Although this is not as bad as a completely random embedding with cost $\Theta\left(\left(N\log N\right)^{1+1/d}\right)$, as some locality can be leveraged, it is still only better off by a factor of $\left(\log N\right)^{1/d}$, which makes it behave close to a random embedding. We note that as the size of the problem increases, the number of cores runs out and d becomes 1, yielding an asymptotic spatio-temporal cost of $\Theta\left(\frac{N^2\log N}{P}\right)$. For the case of a single processor $P = 1$ this is a cost of $\Theta\left(N^2\log N\right)$. This has a simple interpretation – as the computational time is $\Theta\left(N\log N\right)$ and the storage requirements are $\Theta\left(N\right)$, so that the total temporal communication cost is merely the product of these.

8.2.4 Sorting

We note that tree-based sorting algorithms that take time $O\left(n\log n\right)$ in a RAM model, would actually take time $O\left(n^{1+1/d_S}\right)$ upon accounting for communication costs in a d_S-dimensional substrate. Lang et al. developed a 2-D sort algorithm that takes $O\left(\sqrt{n}\right)$ time-steps if each element n has its own processor [65]. This approach maps to the CMP domain as well.

8.3 Spatial communication cost of memory accesses

Although in Chapter 7 we discussed the cost of temporal interconnect as being proportional to the temporal distance traversed due to, say, the power consumption of retaining data, another measure concerns the purely spatial communication cost of accessing the data. As was shown in Chapter 2, because data necessarily takes up physical space, accessing an entry requires traversing the physical space, thus incurring a spatial communication cost. As per Chapter 7, the minimum aggregate cost is achieved by greedily allocating temporal interconnect according to access frequency, as per the Rentian Memory model. So short temporal interconnects should be located close to the accessor, and long temporal interconnects should be located more remotely. We have for a temporal distance distribution $f(t)$ the storage space taken up by temporal interconnect of up to distance τ as:

$$M(\tau) \equiv \int_0^\tau t f(t)\, dt$$

If the density of storage is uniform in d_S physical dimensions of *space*, then the spatial cost of access $C_T(T)$ is given by:

$$C_T(\tau) \equiv \Theta\left((M(\tau))^{1/d_S}\right)$$

The average spatial memory access cost $\bar{C_T}$ is then given by:

$$\bar{C_T} \equiv \int_0^{\tau_{max}} f(t)\,(M(t))^{1/d_S}\, dt$$

For example, in the single-core case, if we have a temporal Rentian behaviour for $t \geq t_{min}$ of:

$$f(t) = \lambda t^{p-2}$$

then:

$$M(\tau) = \frac{\lambda}{p}\left(\tau^p - t_{min}^p\right)$$

we can also relate the temporal distance τ to the amount of memory needed for all temporal interconnect up to τ as:

$$\tau(M) = \left(\frac{p}{\lambda}M + t_{min}^p\right)^{1/p}$$

Now, we have:

$$C_T(\tau) \propto \left(\tau^p - t_{min}^p\right)^{1/d_S}$$

then:

$$\bar{C_T} = \begin{cases} \Omega\left(t_{max}^{\left(1+\frac{1}{d_S}\right)p-1}\right) & , p > \frac{d_S}{d_S+1} \\ \Omega\left(\log t_{max}\right) & , p = \frac{d_S}{d_S+1} \\ \Omega(1) & , p < \frac{d_S}{d_S+1} \end{cases}$$

or, alternatively let M_{max} be the amount of memory needed for the algorithm, then:

$$\bar{C_T} = \begin{cases} \Omega\left(M_{max}^{1+\frac{1}{d_S}-\frac{1}{p}}\right) & , p > \frac{d_S}{d_S+1} \\ \Omega\left(\log M_{max}\right) & , p = \frac{d_S}{d_S+1} \\ \Omega\left(1\right) & , p < \frac{d_S}{d_S+1} \end{cases}$$

These are lower-bounds as this is derived for an optimal cost memory model – the Rentian memory of Chapter 7. Real memories and caches are likely to do worse than this. We note that even though memories are spatially distributed, if the distribution of temporal accesses has exponent below a critical threshold, the average access cost can potentially be reduced to $O\left(1\right)$. This occurs when the Rentian dimensionality of the algorithm being executed is less than the spatio-temporal embedding dimension $d = (d_S + 1)$. However, if the Rent's exponent is higher, the average costs grow according to the longest temporal distances or memory footprint.

For uniform memory accesses, we have another special-case of $p = 1$, and thus $t_{max} \propto M_{max}$. This yields an average spatial access cost $\Omega\left(M_{max}^{1/d_S}\right)$. We note that this is the same as shown in Chapter 2 for the binary tree traversal for $d_S = 2$. This also means that if the access pattern has sufficient locality, we can leverage locality by moving frequent accesses closer to the accessor, and potentially recover $O\left(1\right)$ average access costs, making it appear as though it were a uniform memory. In the case of the binary tree traversal, moving frequently accessed items closer to the root would potentially lead to such a reduction in average access cost.

If we recall the dense matrix multiplication problem $MM\left(N \times N\right)$ we know that $p = \frac{2}{3}$. Now, for a fixed number of cores P, surrounded by a pool of memory in $d_S = 2$ spatial dimensions, as the size of the matrix multiply N grows, the spatial communication costs of memory accesses would be expected to grow on average by $\Omega\left(\log N\right)$. This leads to total access costs of $\Omega\left(N^3 \log N\right)$. However, if the operation was surrounded by a pool of memory in $d_S = 3$ spatial dimensions, then we have average access cost $\Omega\left(1\right)$ and a total access cost of $\Omega\left(N^3\right)$, which is what a PRAM or BSP model with uniform memory access, would also yield. Indeed, work by Aggarwal et al. on the special-case of uniprocessor matrix multiplication with a polynomial memory access cost also noted these three regions of behaviour [4]. The result presented here, however, is more general in that it applies to any algorithm with Rentian locality.

Bilardi and Preparata also considered the access-costs of physical distance traversal for a uniform mesh of three or less dimensions [14, 15, 87]. However, they were primarily concerned with the time complexity of simulating PRAMs and other meshes with a higher number of processors. In their model, however, no memory-access parallelism (per processor) was allowed. They noted that memory access parallelism would increase the storage requirements of handling multiple requests, and thus potentially physically dilate the actual memory – making it even more remote. However, this is a little misleading, as the storage requirements of memory access parallelism do not necessarily have to grow in direct proportion to the amount of memory, but can grow at a considerably slower pace. For example, a tree-like memory access structure (for example, with a depth of $\log \log N$) embedded in two or three dimensions, can at a constant-factor slow-down, provide a logarithmically growing amount of storage for memory-access parallelism. Also, without a theory of access-pattern as the Rentian ones presented here, Bilardi and Preparata consider only a particular synthetic access pattern that lacks the tighter locality properties of the Rentian model.

8.4 Conclusion

This chapter demonstrated how one can extract Rent's exponents by direct algorithm analysis, and how these can then be utilised to determine the scaling costs of communication.

In previous chapters, the Rentian scaling was found in experiments where embeddings were assumed to be optimal or near-optimal due to simulated annealing. The problem is that optimal or near-optimal embeddings compose sub-optimally. The key insight here was that one can still maintain Rentian scaling with only a constant factor penalty, even if one relaxes the optimality constraint. Indeed, by merely composing self-similar algorithm embeddings in a hierarchical fashion, Rentian scaling properties are retained, and consequently the associated analytical results from Chapter 7. It was further shown how to derive the asymptotic communication costs of algorithms from such a hierarchical analysis.

Finally, lower-bounds were derived for the spatial-communication component of memory accesses. It was shown that this lower-bound undergoes transitions at critical Rent's exponents, based on their temporal locality and the physical dimensionality of the memory.

Chapter Nine

9

Rentian scaling in neuronal networks

As we integrate even larger quantities of logic into new computational systems with increasing parallelism, we must tackle the growing challenges of rigid power-consumption, thermal and noise constraints. Thus it is inspiring that evolution has already tackled these problems and given us a highly-parallel system as a proof of existence – the brain. As Sarpeshkar has pointed out [92], the human brain is remarkably energy efficient, consuming only 12W of energy for the order of 3.6×10^{15} synaptic operations per second, which in terms of operations per second per Joule, is up to seven orders of magnitude more efficient than existing processors[1].

In brains, as in VLSI, there is a significant cost to communication. Basser derived cable equations for myelinated axons [10], and showed how nodal and myelin capacitance per unit length affect the signal propagation. Moreover this capacitance needs to be charged and discharged with a minimum potential voltage required at the destination, similarly to VLSI technology, with signalling costs growing according to length. Thus it is informative for the continued technological evolution of electronic computational systems, to understand what natural evolution has already done to tackle communication. Indeed, Sarpheshkar's pioneering work noted that for the sake of energy efficiency, information should be coded as a hybrid of analog and digital signalling, and distributed among multiple paths to combat noise [92]. As the human brain consumes about 20% of the total energy whilst accounting for only 2% of the mass in the human body, with a large proportion of that cost being to build and maintain connections between neurons [5, 80], there is abundant incentive for natural evolution to minimise communication. One may ask, therefore, whether or not Rentian scaling has been employed by brains to achieve its desired functionality whilst keeping communication costs down, or if evolution has perhaps come up with an alternative solution that allows it to diverge from Rentian behaviour.

[1]Note that Sarpeshkar's comparison is dated to a high performance processor from 1998

9.1 Clarifying the collaboration

As the following work is the result of a very close collaborative effort, it is important to be clear about the manner in which it arose, and the division of efforts.

The initial inspiration for applying Rent's rule to the human brain came from attending a talk by Professor Steve Furber who discussed their project of a real-time neuronal network simulator and its interconnection network called *SpiNNaker* [44]. Afterwards, in a meeting in March 2007 with Professor Furber and my own supervisor, Dr Moore, we speculated further as to whether or not the human brain might obey Rent's rule. After the meeting, I derived allometric scaling results (see section 9.3) showing that *if* brains actually did obey Rentian behaviour then there should be an observable power-law scaling relationship between the volume of white and grey matter, and that this exponent should lie within a certain expected range. I decided to compile statistics for a range of brain sizes to investigate if this was indeed the case, and instead found publications [115, 19] already showing that there was a very clear power-law scaling relationship across mammalian brains, and that there didn't seem to be particularly good explanations as to why this was the case for both the neocortex and the cerebellum. I concluded that Rentian scaling was a distinct possibility, but I lacked expertise in the area or datasets to test it directly. After attending a fascinating Networks and Neuroscience Symposium in March 2008, I contacted Professor Ed Bullmore. He was interested in this idea and connected it to Herbert Simon's notion [94] that all information processing systems (Brains and Computers) will have common emergent properties - such as hierarchy and modularity. His exceptionally talented PhD student Danielle Bassett was very keen to take a lead and collaborate together on this. I wanted to test for physical Rentian scaling in particular, and to see if the exponent matched the allometric scaling of Bush and Allman [19], based on my derivations. I also wanted to try box-counting to measure fractal dimensionality. They suggested a number of datasets, and I detailed methodology on how to examine these for Rentian behaviour within a brain, and we taught each other many things about our respective fields. Soon after commencing, we found some work by Beiu and Ibrahim [12], examining the possibility of Rent's rule applying to brains. They proposed an 'alternative interpretation' of Rent's rule that they claimed it does follow. However, their analysis had fundamental flaws, that my own allometric derivations had already addressed.

My own physical Rentian-analysis computing environment was not suitable for these datasets, but Bassett took up the lead by quickly adapting, and improving on the methodology I had outlined, into her own Matlab environment. On my end, I adapted my fractal box-counting and topological Rentian environments for these datasets. We interpreted the results of each of these together, and concluded that we were indeed seeing evidence of Rentian behaviour. Dr Bassett performed further analyses on modular hierarchical decomposition, building upon Newman's spectral optimisation algorithm [78], which I do not claim any credit for, and is not presented here. Bassett also came up with the methodology of fitting to other two-parameter-fits to test the relative goodness-of-fit of the power-law. I also refined the allometric scaling derivations to account for varying neuronal complexity, thanks to her input about their properties. We were a little surprised that the theory actually worked, and certainly pleased, when we found that the Rent's exponent for the human brain datasets gave a good prediction for the allometric scaling exponent of white versus grey matter across mammalian brains.

Dr Bassett took the lead and wrote the bulk of the final paper, with sections written by me as joint first-author, and with a lot of back-and-forth in refining it together. Excited about our findings, we initially submitted to *Nature* in July 2009, but did not receive a positive response from them. In cross-disciplinary work such as this, that brings in knowledge from entirely separate fields, we had to also resolve differences in what is accepted knowledge in their respective fields – such as the assertion in the Network Science community that 'small world' networks [110] do not exhibit fractal scaling [91], whereas VLSI deals with datasets that are known to be both small world and have fractal scaling [21, 97]. After a lengthy period of submissions, reviews and revisions, we were accepted by *PLoS Computational Biology* in March, and published in April 2010. In the interim period, two papers [90, 86] in particular came out that took a *little* bit of our thunder, examining topological Rentian scaling (in *c.elegans*), versus our focus on physical Rentian scaling, that we added as 'prior work' and discussed in our final version, however we feel that our work here, is a significant contribution to the field.

This chapter focuses on my own contributions to this work, dealing more specifically with the Rentian properties observed and predicted allometric scaling. For a more detailed discussion of the neuroscience behind it, and on the complementary work on characterisation of hierarchical modularity, please refer to our paper [33].

9.2 Datasets and their limitations

The *C.elegans* dataset consisted of previously published data from Kaiser et al. and Chloe et al. [61, 27, 112]. This combines the complete spatial position information of each neuron, along with their network connectivity. Here, wherever there is a synapse connection between any two neurons, whether electrical or chemical, this is represented by an edge in the network. Altogether there are 277 nodes and 2105 edges. We should note that because the neurons innervate an entire organism, they are not homogeneously distributed, with approximately 131 nodes making up the 'head' and the remaining dispersed over the rest of the organism.

Complete neuronal wiring maps of the human brain are unlikely to become available anytime in the immediate future, thus we necessarily needed to use coarse-grain maps that detailed the wiring at the most global scales. The datasets used were the most fine-grained maps of connectivity available. For these human neocortical datasets, two forms of network measurement were used based on *Diffusion Spectrum Imaging* (DSI), and

Magnetic Resonance Imaging (MRI). For the MRI dataset [11], 259 healthy subjects were scanned by MRI, with grey-matter cortical thickness measurements taken in 104 regions according to a fixed parcellation of the neocortex, namely the *Brodmann* areas. Lerch et al. [68] had found that correlations in grey-matter neocortical thickness across subjects resulted in similar results to previously obtained connectivity maps (starting from a single seed area) using an explicit connectivity mapping technique – Diffusion Tensor Imaging (DTI). The implication is that this technique may be a faster, simpler method of building reasonably reliable *global* connectivity maps of neocortical regions. This dataset consists of 104 nodes and 1606 edges.

The DSI approach actually utilises MRI technology but in a different manner. It measures the diffusion of water molecules in a 3D volume. By analysing and following this diffusion vector of water along axons (with surrounding myelin sheaths acting as diffusion barriers), it is possible to map their connectivity. However correctly 'following' voxels of diffusion can be problematic due to resolution constraints, as there may be multiple candidate neighbouring voxels to follow. This means that longer axons in a complex tract may be harder to follow, with a resultant lower probability. The heuristic utilised by the dataset authors [47] progressively builds the connectivity matrix by adding connections to a backbone minimum spanning tree, starting with the highest probabilities, until reaching average degree four. We should note that this may ordinarily lead to some bias of shorter connections over longer connections. Although Haggmann et al. tried to apply a distance bias correction in their preprocessing, it is still a matter of debate as to what the correct method is for this [47], and such a bias may still be present. In this dataset, the entire neocortex was parcellated into 998 cortical regions, and five subjects were scanned to build the dataset, with one subject scanned twice.

For comparison, a VLSI circuit was also analysed. This is the s953 benchmark circuit [16], with 440 gates and 772 edges. Note that to maintain consistency and due to limitations with the DSI and MRI datasets, edges were used to denote connectivity, instead of hyper-edges.

9.3 Allometric scaling

The self-similarity of Rent's rule governs not only the internal characteristics of VLSI circuits, but also the scaling characteristics as these circuits grow in size [99]. Thus, if Rent's rule did hold for the brain, we might expect a similar relationship across a range of different brain sizes, as *allometric scaling*. For our derivation, we first need to define the number of connections in a cross-sectional area as a function of the white matter volume, and second, to determine the number of processing elements in the whole system as a function of the grey-matter volume.

In VLSIs, the wiring of the circuit is located in multiple layers above the logic of the circuit. Thus, the area of the logic-limited circuit is equal to the area of the logic, $V = G$.

In the human brain, however, the white matter tracts are embedded in the same space as the 'logic' or grey matter. The 'wiring' of the white matter tracts can cause the 'logic' or grey matter segments to be farther apart from each other. Thus, the volume of the system is given as $V = W + G$, which for a homogeneous distribution of axons within grey-matter, is an increase to just G of $(W+G)/G$ or $(1+W/G)$. This stretching further causes an increase in the white matter volume, W, by increasing axon length. Since axon length is a one dimensional linear term, the dilated white matter is larger than the undilated white matter by a stretching constant μ given by the cube root of the total increase:

$$\mu = \sqrt[3]{1+W/G}. \tag{9.1}$$

Over the scale of the mammalian white matter, this dilation effect is very small. Even if W varies as widely as $W \sim (0, G]$, μ will vary as $\mu \sim (1, 1.26]$. In $\log_{10}$ space, this variation in μ becomes $\log_{10}(\mu) \sim (0, 0.1]$ which is a small perturbation of $\log(W)$, over the range of mammalian white matter values. Thus, it is possible to approximate the dilated white matter volume $W_{dilated}$ by the undilated white matter volume $W_{undilated}$.

$$W_{dilated} \sim W_{undilated} \tag{9.2}$$

In the remainder of this derivation, the simple symbol W will be used to refer to $W_{dilated} \sim W_{undilated}$. Furthermore, this is a worst-case behaviour for dilation with interconnect uniformly distributed in grey-matter. If we could better separate the white matter from the grey matter, then distances between grey-matter could be relatively undilated at lower-levels of the hierarchy. To understand this, recall from Chapter 8, that to obey Rent's rule, it is merely sufficient to obey it with hierarchical decomposition (rather than uniformly). Indeed it can even support random or worst-case interconnect at each hierarchical level with only a constant-factor penalty. This means that, if connected in a more tree-like fashion, dilation need only occur for interconnect at that particular level of hierarchy, with little or no dilation at the lower levels, but with relatively more dilation at higher levels. This is somewhat analogous to the multi-layered approach of VLSI interconnect, where there is a relatively undilated bottom logic layer with very little interconnect to dilate logic, and multiple interconnect layers above it that tunnel into the logic layer.

To determine the number of connections in the cross-sectional area, S, as a function of the white matter volume, W, we first approximate the brain as a sphere and therefore the cross-sectional area, A, of the brain as equivalent to the area of a circle:

$$A = \pi r^2, \tag{9.3}$$

where r is the radius of the sphere. We also know that the volume, W, of the sphere is given by

$$W = \frac{4}{3}\pi r^3, \tag{9.4}$$

which can be rewritten as

$$r^3 = \frac{3}{4\pi}W. \tag{9.5}$$

Substituting, we find that

$$A = \pi(\frac{3}{4\pi}W)^{2/3} = \pi^{1/3}\left(\frac{3}{4}W\right)^{2/3} = C_1 W^{2/3}, \tag{9.6}$$

where $C_1 = \pi^{1/3}(\frac{3}{4})^{2/3}$ contains all constants independent of W.

We should point out that although a sphere was used in this derivation, any fixed three-dimensional shape would also result in cross-sectional area growing by the 2/3 power of volume. Thus other shapes may alter the constant C_1, but not the exponent 2/3.

By definition, the number of connections within the cross-sectional area can be written as

$$S = \theta A, \tag{9.7}$$

where S is the number of synapses, θ is the number of synapses per unit area, and A is the cross-sectional area. Bassett pointed out, that from prior anatomical data [3], the number of synapses per unit volume is independent of white matter volume, W, and therefore the number of synapses per unit area, θ, is also independent of W. So we can now define θ to be

$$\theta = C_2. \tag{9.8}$$

Substituting eqns 9.6 and 9.8 into eqn 9.7, we can write the number of synaptic connections S as a function of white matter volume, W:

$$S = \theta A = C_1 C_2 W^{2/3} = C_3 W^{2/3}. \tag{9.9}$$

where $C_3 = C_1 C_2 = \pi^{1/3}(\frac{3}{4})^{2/3}\theta$.

To define a relationship between grey matter volume G and the number of processing elements N, we first recall that the number of neurons in the brain scales disproportionately slowly with the grey matter volume as $G^{2/3}$ while the number of synapses scales directly with G [3]. Therefore, the number of synapses per neuron, or synaptic complexity of the neurons, is increasing with brain volume. In computer circuits, we also have gates that vary in complexity, and circuits can even be built up of macrocells that have very high complexity. However, to perform the gate count needed for Rentian analysis in VLSIs, we must count computing elements in comparable terms, typically of the simple 2-input logic gate (NAND2). Thus, larger and more complex computing elements are counted as multiple NAND2 gates. In order to work with the same counting statistics in the human brain, we define a constant-complexity processing unit as using a fixed number of synapses. Since the number of synapses scales with G, then so does the number of constant-complexity computing units in the grey matter:

$$N = \phi G, \tag{9.10}$$

where ϕ is the number of constant-complexity computing elements per unit volume.

Now we can rewrite the Rentian scaling relationship

$$S = kN^p, \tag{9.11}$$

where k is the Rent coefficient and p is the Rent exponent, in terms of white matter W and grey matter G volumes:

$$C_3 W^{2/3} = k(\phi G)^p, \tag{9.12}$$

or

$$W = C_4 G^{3p/2}, \tag{9.13}$$

where C_4 is $\left(\frac{k\phi^p}{C_3}\right)^{3/2} = \frac{4}{3\sqrt{\pi}}\left(\frac{k\phi^p}{\theta}\right)^{3/2}$.

Neglecting this constant, we can write the allometric scaling relationship more simply as $W \sim G^{\frac{3p}{2}}$. Thus, the allometric scaling exponent $a = \frac{3p}{2}$ should be multiplied by 2/3 to find an estimate of the Rent exponent, p. Taking the Rent's exponents from the MRI and DSI datasets, we may therefore predict the expected allometric scaling between white matter and grey matter.

The cross-sectional count of interconnect here is only for white matter, rather than grey matter – which also contains some local interconnect. Because of the self-similarity of Rent's rule, the presence of local grey-matter interconnect does not necessarily affect the scaling characteristics. An analogy can be made with VLSI. If one were to divide a logic block and count the total number of wires cut, they would scale according to the Rent's exponent. However, the same behaviour would occur if one dropped all short wires up to a certain length (say by removing the bottom two layers of metal in a VLSI). Now counting only the wires bisected in the top layers of VLSI metal, the same asymptotic power-law exponent would be seen. Another analogy can be found by observing that connections between gates aren't even the smallest form of interconnect. Gates, themselves are composed of many interconnected transistors. Here, the short local grey-matter interconnect is treated analogously to the short, local poly-silicon interconnect between transistors within gates, and the white-matter is treated analogously to the remaining metal-based interconnect. If one really were to properly partition a VLSI logic block, then gates should also be chopped, and their internal interconnect (traditionally poly-silicon) should be exposed. Nonetheless, we are content with operating at a gate-granularity, rather than at a transistor-granularity. Moreover, we are far more interested in the asymptotic scaling behaviour at large lengths, rather than at short lengths – as it is these that dominate the *total* interconnect cost.

9.4 Methodology

For these datasets, analyses were performed to determine the presence of physical Rentian scaling, topological Rentian scaling, and to determine a box-counting dimension of the network. Most of these methodologies have already been explained in previous chapters. Topological Rentian scaling was explained in Chapter 3. The fractal-dimensional box-counting of Song et al. was covered in Chapter 5.

The most interesting analyses here are of the physical Rentian scaling. Unlike the gates in VLSI, or software in a CMP, the physical position of network nodes and their connectivity are constrained by nature and evolution, rather than as an immediate solution to an optimisation problem. Thus, unlike the traditional domain of Rentian analysis, it is more important to establish that physical rather than the topological network obeys Rent's rule. Furthermore, it is the actual physical Rent's exponent, rather than the topological Rent's exponent, that is needed for characterising the connectivity properties of these neuronal networks, and their resultant allometric scaling. Also, unlike in VLSI, where gates are more or less homogeneously distributed, for neurological networks the physical distribution of nodes can be more heterogeneous. In the case of *C.elegans*, in particular, most of the neurons are located in concentrated regions at the far ends of the organism, whilst the remaining neurons are interspersed between them, innervating the rest of the organism. For the physical Rentian scaling analysis, the physical network was covered by 5000 boxes of uniformly random sizes and uniformly random spatial position, and the number of terminals (links connecting to nodes within the box) were plotted against the number of nodes within the box.

9.4.1 Re-wired networks

Due to the physical and metabolic cost of interconnect, one may reasonably ask why neuronal networks aren't *more* minimally connected to reduce these costs. We thought it would be interesting, therefore, to compare the actual Rentian scaling properties against extremal cases of minimally re-wired and randomly re-wired versions of these datasets. For the minimally re-wired set, the spatial information was used by Dr Bassett to first build a minimal spanning tree to ensure connectivity, and the next shortest edge was iteratively added until the total number of edges matched the original dataset. For the randomly rewired networks, edges were randomly added until it matched the total of the original dataset.

9.4.2 Non-power-law null hypotheses

With Rentian scaling, we would expect an approximately power-law distribution in the number of terminals to the number of nodes. In order to assess its significance, it was important to test it against other null hypotheses. To be fair we utilised other two-parameter models, including logarithmic, exponential, and two-parameter polynomial fits.

9.5 Small World vs. Rentian Fractality

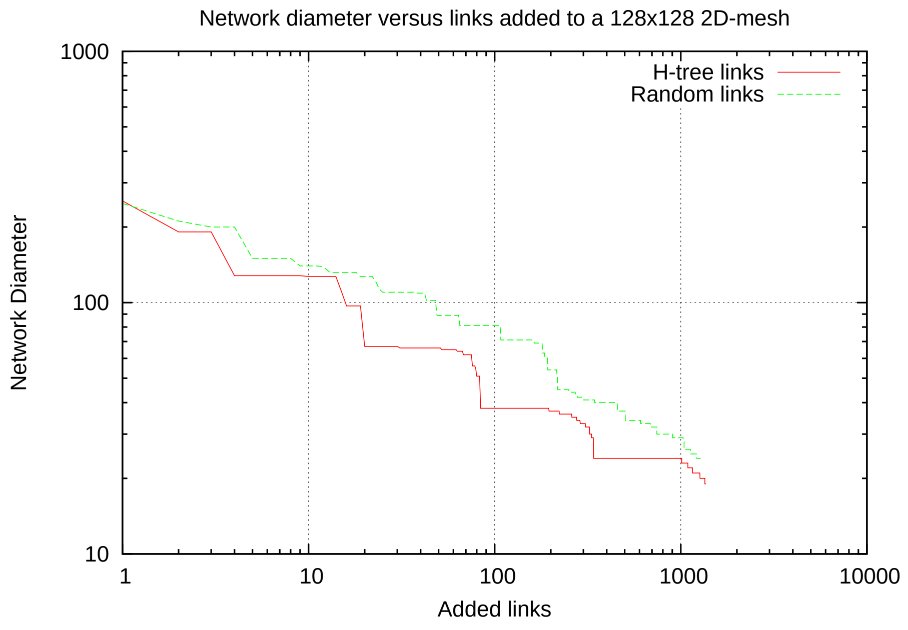
There are important differences in fractal behaviour for Rentian scaling and box-counting measures. Significantly, Rentian scaling is fully compatible with small-world networks, whereas the box-counting method of Song et al. is not [95]. As many networks including neuronal ones are likely to be small-world, it is important to elaborate on their differences. As described earlier, we estimated the topological dimension of the information processing networks using both topological Rent exponents and box-counting. We observed that fractal scaling was clearly visible in the Rentian analysis and less visible in the box-counting plots. However, it is known that small-world properties affect the scaling measured by a box-counting analysis. Work by Rozenfeld [91] discusses this, and demonstrates that there is *some* compatibility between fractal box-counting dimension and small-world topologies. However, even though the fractal topology of a network might be seen at small scales, the small world nature means that there is a single box that covers the entire network with side-length of order $\log N$. This means that the small-world behaviour rapidly dominates in box-counting, leading to a rapid cut-off of the power-law behaviour.

Compared to box-counting, Rentian scaling looks at how the number of edges crossing a boundary scales with the number of nodes inside it. In VLSI, Ozatkas has observed that this can be thought of as relating the volume of logic inside the boundary, to the flow of information (surface area) across that boundary in terms of edges [84, 83]. The Rentian exponent is then given by the ratio of surface area scaling to volume scaling - and can thus be related to the dimensionality of information flow scaling. One can easily show that large d-dimensional meshes have Rent's exponents of $p = (d-1)/d$. Moreover, as already noted in Chapter 3, we can define the Rentian information-flow dimension to be:

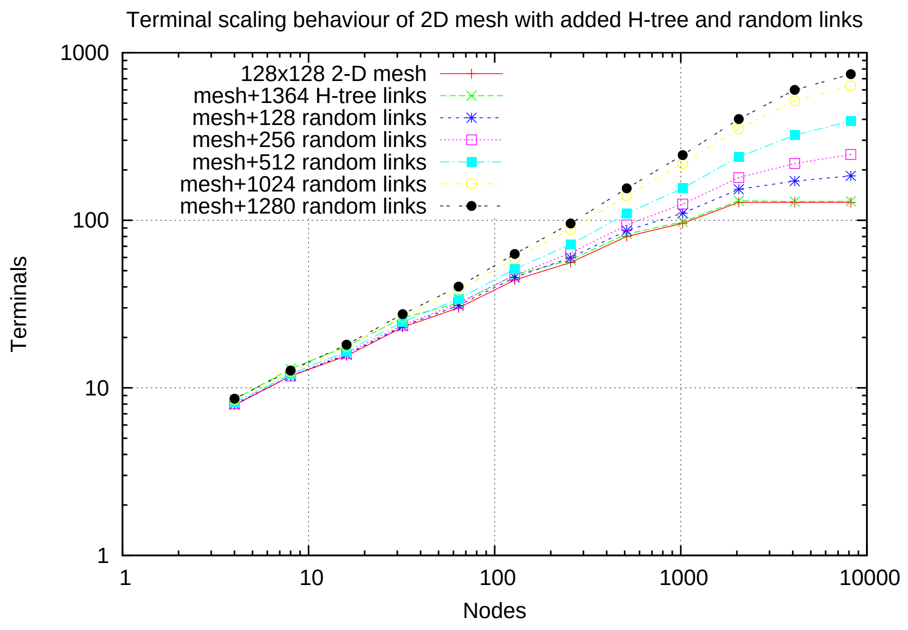
$$d = 1/(1-p)$$

Whereas the topological distance used for assessing small-world behaviour and in box-counting analyses examines the number of hops from one node to another, it isn't concerned with the quantity of information flow between them. A single link may connect two small world sub-networks, and thus allow only a small flow of information between these two sub-networks whilst maintaining small-world behaviour. That is, the single link is shared between all the nodes on one sub-network that want to communicate with a node on the other sub-network. Although such a network may be small-world, one has to wonder whether that single link is sufficient for all the information flows required for solving real computational problems – certainly despite the desire to reduce wiring costs, such connectivity patterns generally aren't practical in VLSI. A Rentian analysis, however, is primarily concerned with the quantity and scaling of communication required, rather than topological distance alone.

VLSI networks are known to obey *both* fractal Rentian behaviour and small-world topologies [21, 97]. We should emphasise that these are mutually compatible attributes. For VLSI, a clock tree, typically an H-tree [18], uniformly distributes clock timing information to disparate parts of the logic network, so that they can synchronise their communication. H-trees are called thus because their basic structure consists of physical links, in the 2D space of a VLSI chip, that look like the letter H. At each of the four ends of the H, another H of half the size is attached at its middle, and so on, thus forming a fractal tree of H's. Clock trees are very expensive from both a manufacturing and power-



(a) Network diameter with addition of links to a 128x128 2D mesh



(b) Effect of added links to Rentian scaling.

Figure 9.1: Effect of adding links to network diameter and Rentian scaling. Note how the H-tree links reduce network diameter leading to a Small-World network, and yet largely preserve the fractal Rentian scaling properties of the underlying 2D mesh. In contrast, the random links disrupt the Rentian scaling.

consumption standpoint. The H-tree emerged as a way to achieve the goal of reducing communication skew at leaf nodes (ensuring that the logic is globally synchronised as a small-world), whilst keeping costs reasonable. We should point out that clock trees are by no means the only cause for small-world topologies, and that the rest of the circuitry may also exhibit small-world behaviour for other reasons.

In figure 9.1a we see the effect on network-diameter of adding multiple levels of an H-

Tree versus adding random links to a 2-D mesh (not torus), resulting in a small-world network for both types. Here, each level of the H-tree consists of four links to a central node. In figure 9.1b we also see that the Rentian behaviour and exponent are only slightly affected by the addition of an H-tree, whereas the random links lead to a large divergence in the Rentian behaviour and exponent. Thus the *nature* of the small-world behaviour is important in assessing its impact on Rentian scaling. In the case of random links, the number of links cut by partitioning approximately grows by the number of nodes in each partition, whereas in the case of the H-tree, the number of links cut is small and approximately constant at each level of partitioning. This compares with the 2-D mesh, where the number of links cut grows approximately by the square-root of the number of nodes in each partition. Importantly, unlike the box-counting metric of Song [95] where small-world properties dominate over any fractal connectivity, the small-world property does not necessarily impact on the physical and topological measures of Rentian fractal behaviour. It is also important to distinguish between the expected truncated power-law distribution of physical link distances versus the potentially non-power-law nature of topological distances, especially in small-world networks.

9.6 Results

Here we examine the results of topological Rentian partitioning, fractal box-counting and physical Rentian scaling analyses. We then use the physical Rentian scaling results to show how the predicted allometric scaling of the mammalian neocortex compares to observations.

9.6.1 Topological properties

We observed good topological Rentian scaling in each of the datasets, with a clear Region I as seen in figure 9.2. We note the general agreement between the DSI subjects in Region I, despite deviations at the larger non-power-law Region II scale. We found that a power law also provided a significantly better fit than other models as can be seen in figure 9.5B. The best-fit exponents p_T are shown in table 9.1, along with the estimated fractal dimensionalities $\hat{D}_T\left(p_T\right) = 1/\left(1 - p_T\right)$.

Given the excellent Rentian scaling seen in topological space, we decided that a box-counting analysis was also appropriate for determining fractal dimensionality. We see in figure 9.3 that unlike the box-counting results of software from Chapter 5, we certainly do not see a clear power-law scaling behaviour. However, we should note that there are far fewer nodes to work with than in Chapter 5, and recall that there are boundary effects at both small box-sizes and small box-counts. Utilising a similar methodology

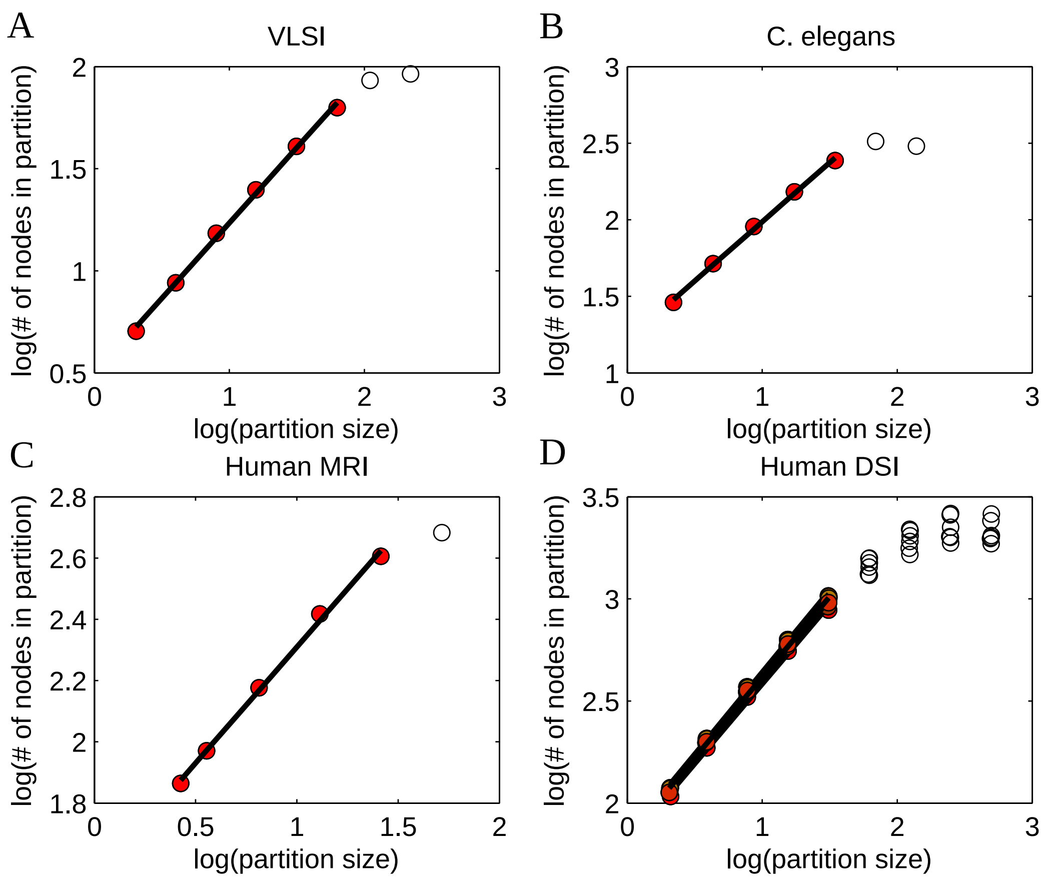


Figure 9.2: Topological Rentian scaling observed in datasets

Network	Partitioning		Box Counting
	p_T	$\hat{D}_T\ (p_T)$	D_T
VLSI	0.73 ± 0.04	3.81 ± 1.04	4.02 ± 0.66
C. elegans	0.77 ± 0.06	4.42 ± 1.06	4.52 ± 0.49
Human MRI	0.75 ± 0.07	4.12 ± 1.07	5.07 ± 1.58
Human DSI 1	0.78 ± 0.07	4.54 ± 1.07	4.68 ± 0.77
Human DSI 2	0.80 ± 0.06	5.06 ± 1.06	4.59 ± 0.76
Human DSI 3	0.77 ± 0.09	4.42 ± 1.08	4.56 ± 0.81
Human DSI 4	0.79 ± 0.06	4.97 ± 1.06	5.24 ± 0.39
Human DSI 5	0.79 ± 0.07	4.84 ± 1.07	4.60 ± 0.82
Human DSI 6	0.78 ± 0.08	4.73 ± 1.09	4.64 ± 0.76

Table 9.1: Topological Rentian scaling and box-counting dimensionality. Errors for p_T and D_T are 95% confidence intervals.

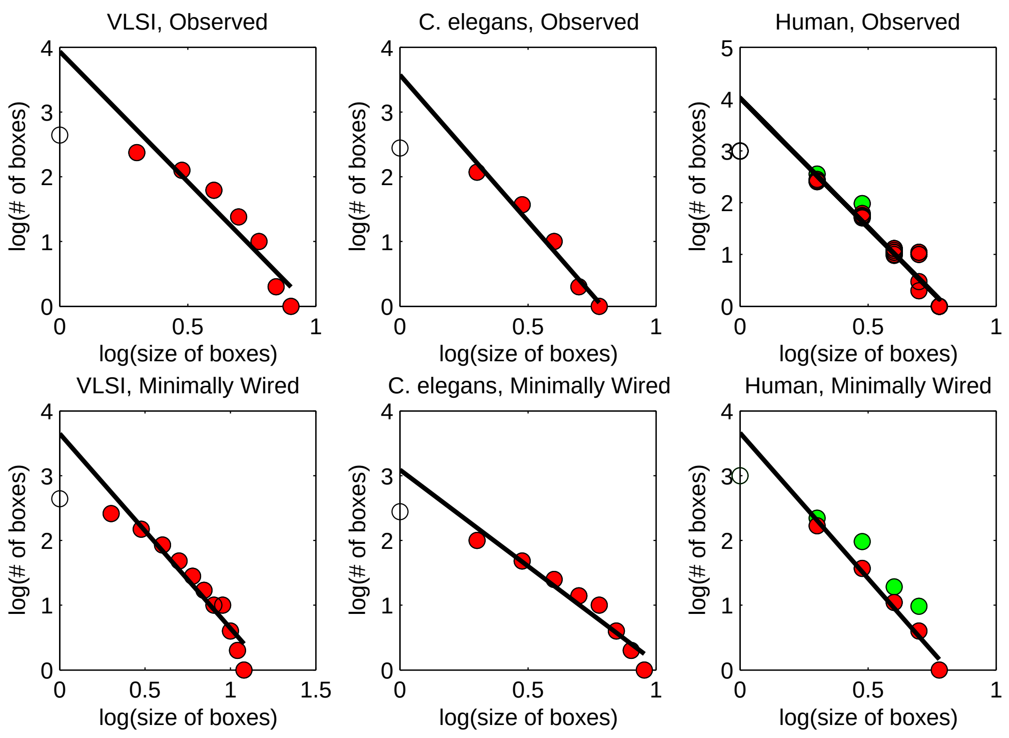
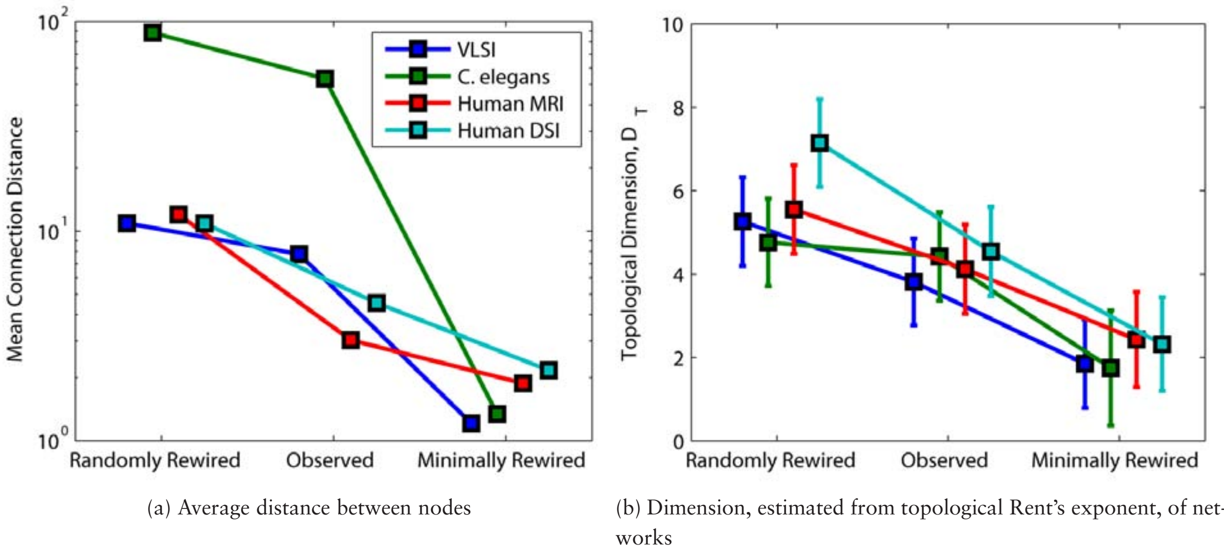


Figure 9.3: Box-counting measures on the topological and minimally-rewired datasets. For Human dataset, the green dots are from MRI and the red are from DSI. The randomly-rewired dataset provided too few points to be meaningful so are not shown here.



(a) Average distance between nodes

(b) Dimension, estimated from topological Rent's exponent, of networks

Figure 9.4: Comparison of observed, minimally rewired and randomly rewired network properties.

to Chapter 5, we ignore the first and last points, and measure the best-fit scaling of the central points, to estimate a dimensionality. Remarkably, despite the lack of points, as seen in table 9.1, there was good agreement in box-counting dimension and the estimated dimensionality determined by topological Rentian scaling.

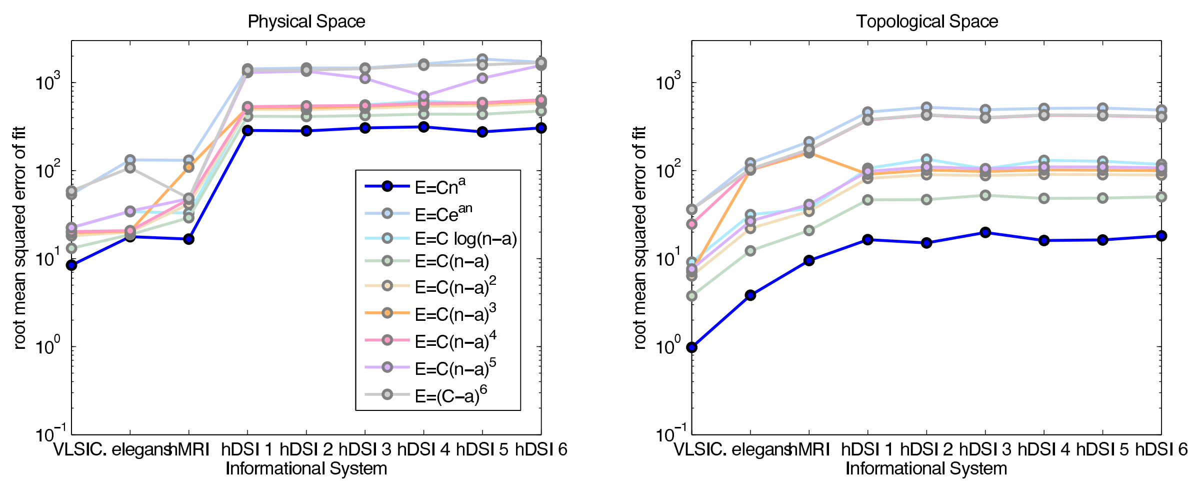


Figure 9.5: Comparison of RMS errors in fit across multiple two-parameter models.

We can also compare the basic network properties of average physical distance and topological dimension between the observed, minimally-rewired and randomly-rewired network, as seen in figure 9.4. Here distance is normalised by setting the average distance of neighbouring nodes found in the dataset to be one. We note that the average distance for the *C.elegans* dataset is much higher for the randomly rewired network due to the larger inhomogeneities in the spatial distribution of nodes. We see in figure 9.4, that the observed networks have properties that lie in-between the minimally wired and randomly rewired networks, in terms of average distance and topological dimensionality. As the size of the network increases, we would expect the random network to tend to higher topological dimensionality, whereas the real network would largely maintain its dimensionality, as would the minimally re-wired network.

9.6.2 Physical Rentian scaling

As seen in figure 9.6 we can also see a strong presence of physical Rentian scaling in the biological datasets. The RMS errors for a variety of fits is shown in figure 9.5A, where the power-law distribution is seen to have the lowest RMS fit out of the candidate two-parameter functions.

For the case of the VLSI network, the gates were embedded in a 2-D region with a simulated-annealing heuristic based on the TimberWolf algorithm. Unfortunately, the results for the embedding of the VLSI network were not as optimal as has been previously reported [100].

Examining the minimally re-wired networks in figure 9.7, we can see that the Rentian behaviour is disrupted for *C.elegans* and dispersed for the other datasets. As was derived in Chapter 6, we would expect a fixed length distribution (such as nearest neighbour) to result in Rentian scaling of $p = 1 - 1/d_E$ for embedding dimension d_E, but only in the case of a homogeneous spatial distribution of nodes. The best-fit slopes are shown

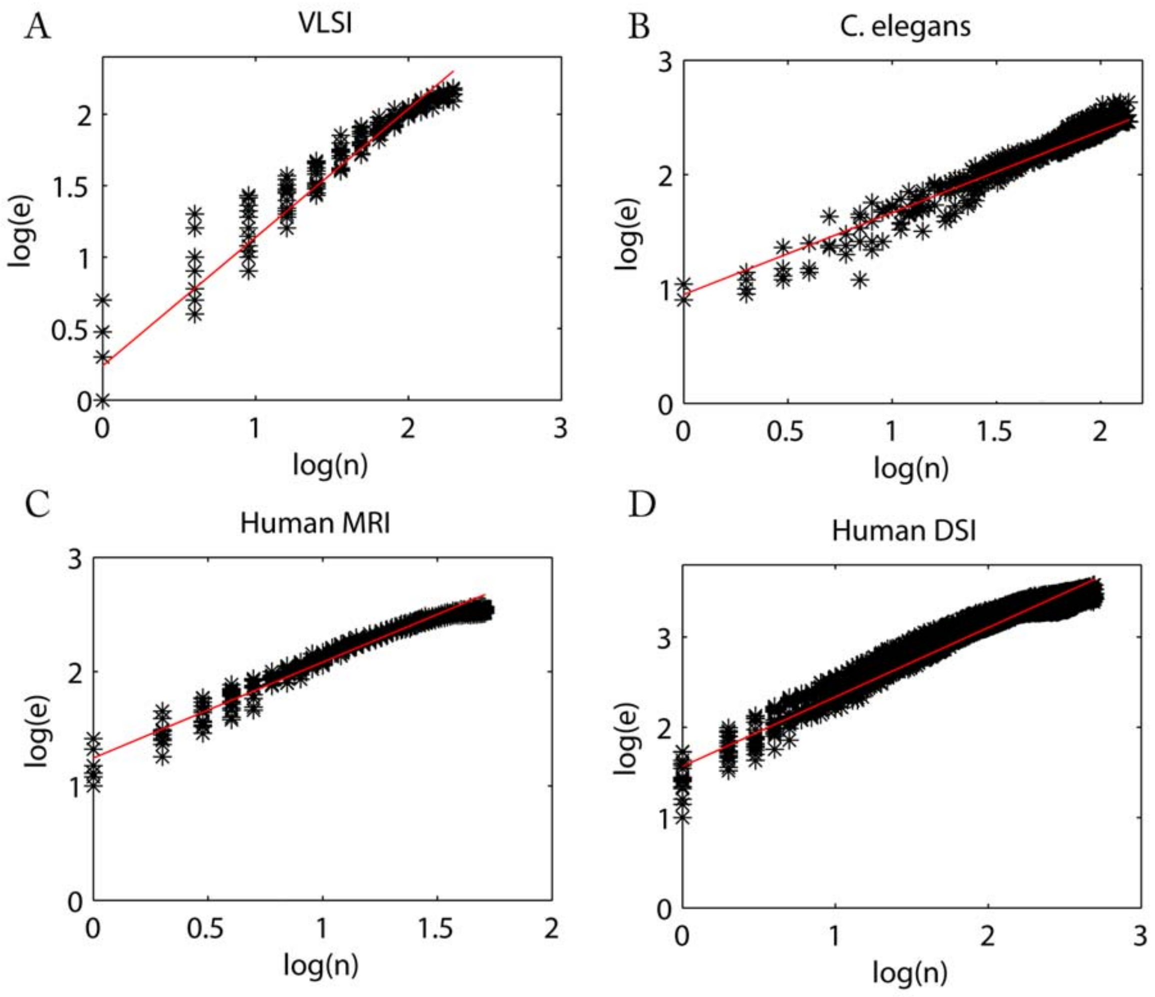


Figure 9.6: Observed physical Rentian scaling for the datasets

in table 9.2, and we observe that for VLSI and DSI datasets, they are close to their expected theoretical values of 1/2 and 2/3 respectively (for a 2-D network embedded in 2-D space, and a 3-D network embedded in 3-D space respectively). Compared to the DSI dataset, the MRI dataset is considerably more dispersed, making slope estimation harder, and yields a high exponent of 0.93. While the DSI and VLSI datasets have a largely homogeneous distribution of nodes in space, the MRI and, in particular, the *C.elegans* dataset have a heterogeneous distribution. It is understandable, therefore, that upon minimally rewiring its network, the *C.elegans* dataset simply doesn't appear to follow any scaling relationship. We observe that in comparison to their minimally rewired versions, the original four datasets in figure 9.6 have considerably tighter scaling, with a non-trivial Rentian exponent, despite any heterogeneity in their spatial distribution. This all points to a strong support of the Rentian-scaling hypothesis for these neurological datasets.

Based on the physical Rentian exponents observed in the human dataset and assuming it is representative, we can infer an expected allometric scaling exponent across all mammalian brains. We observe in figure 9.8, the data from Bush and Allman [19] that shows

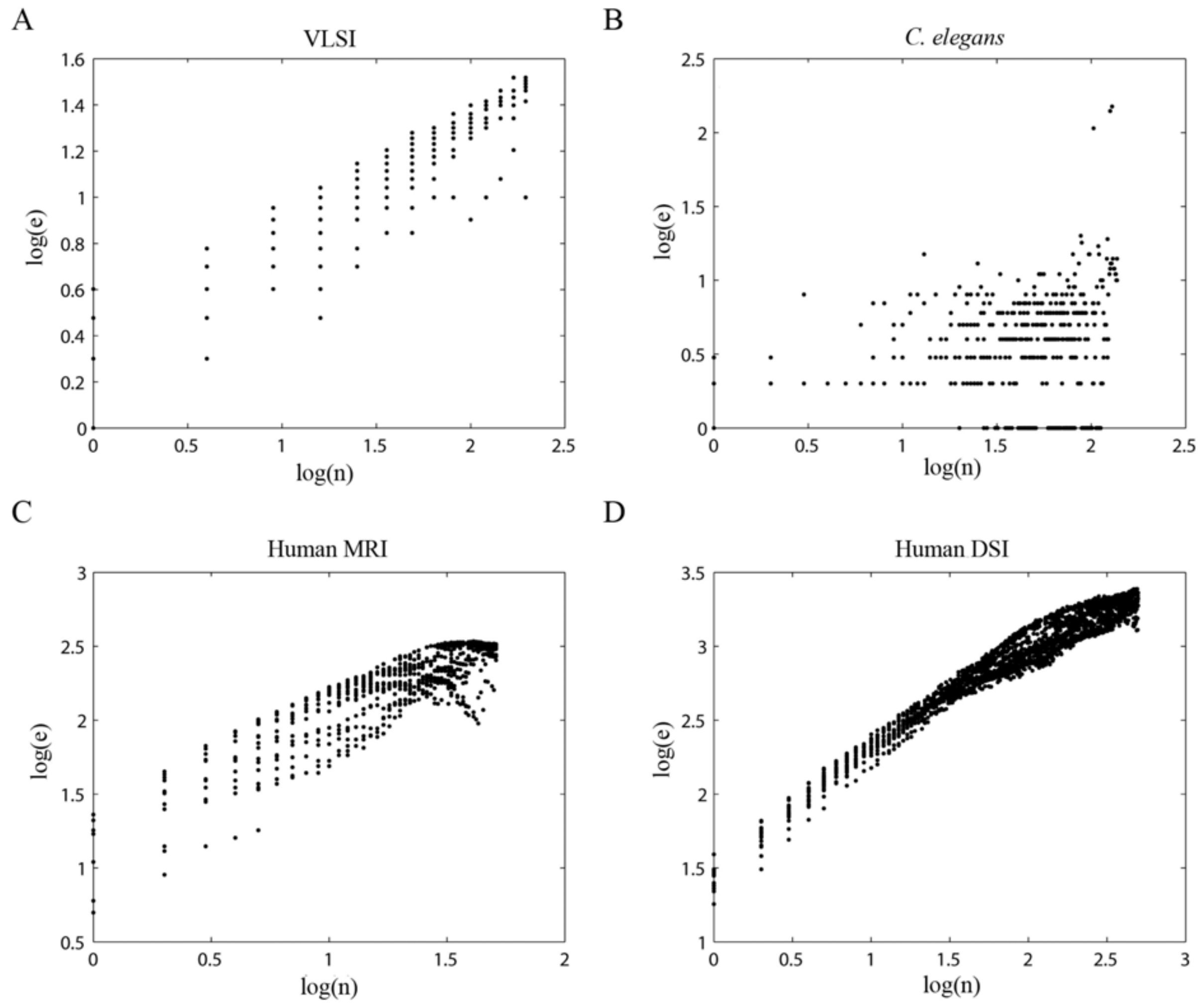


Figure 9.7: Scaling property of minimally re-wired networks. Rentian scaling is disrupted in C.elegans, and dispersed in the VLSI and MRI datasets. The DSI dataset still exhibits relatively tight scaling but with Rent's exponent 0.68 which is close to the expected value of 2/3 for a 3-D network embedded in 3-D space.

a clear power-law scaling behaviour between white matter and grey matter in the neocortex, across many orders of magnitude of mammalian brain sizes (logarithms being in base-10 here). On top of it are plotted the expected scaling behaviour based on the Rentian fits of the MRI and DSI datasets. We can see a very good agreement for the MRI dataset, and a reasonably good agreement with the DSI one.

9.7 Prior work

Bieu et al. [12] also looked at the allometric scaling of white matter to grey matter, and tried to relate this to Rent's rule, however they appear to have misunderstood Rent's rule. They refer to the power-law relationship of white matter to grey matter *volume* as implying the *same* power-law exponent for the *number* of connections vs. the *number* of neurons (with an added linear term). They have effectively equated the volume of white matter to the number of links, and volume of grey matter to the number of neurons.

Network	Nodes	Edges		Physical	Topological	
		ρ	D_E	p	p_T	$\hat{D}_T\ (p_T)$
Observed						
VLSI	440	0.4%	2	0.901 ± 0.006	0.73 ± 0.04	3.81 ± 1.04
C. elegans	277	2.7%	2	0.74 ± 0.07	0.77 ± 0.06	4.42 ± 1.06
Human MRI	104	15%	3	0.828 ± 0.005	0.75 ± 0.07	4.12 ± 1.07
Human DSI	1000	2.7%	3	0.782 ± 0.014	0.78 ± 0.07	4.54 ± 2.12
Randomly Rewired						
VLSI	440	0.4%	2	0.927 ± 0.003	0.81 ± 0.06	5.26 ± 2.42
C. elegans	277	2.7%	2	0.805 ± 0.003	0.79 ± 0.05	4.75 ± 1.48
Human MRI	104	15%	3	0.874 ± 0.003	0.82 ± 0.06	5.55 ± 1.06
Human DSI	1000	2.7%	3	0.925 ± 0.002	0.86 ± 0.05	7.14 ± 2.77
Minimally Rewired						
VLSI	440	0.4%	2	0.509 ± 0.005	0.46 ± 0.06	1.85 ± 0.23
C. elegans	277	2.7%	2	N/A	0.43 ± 0.28	1.75 ± 1.69
Human MRI	104	15%	3	0.93 ± 0.01	0.59 ± 0.13	2.43 ± 1.13
Human DSI	1000	2.7%	3	0.68 ± 0.004	0.57 ± 0.11	2.32 ± 0.79

Table 9.2: Comparison of observed, randomly rewired and minimally rewired network properties. Errors are 95% confidence intervals, except for the DSI dataset which includes both fit-error and variance between subjects.

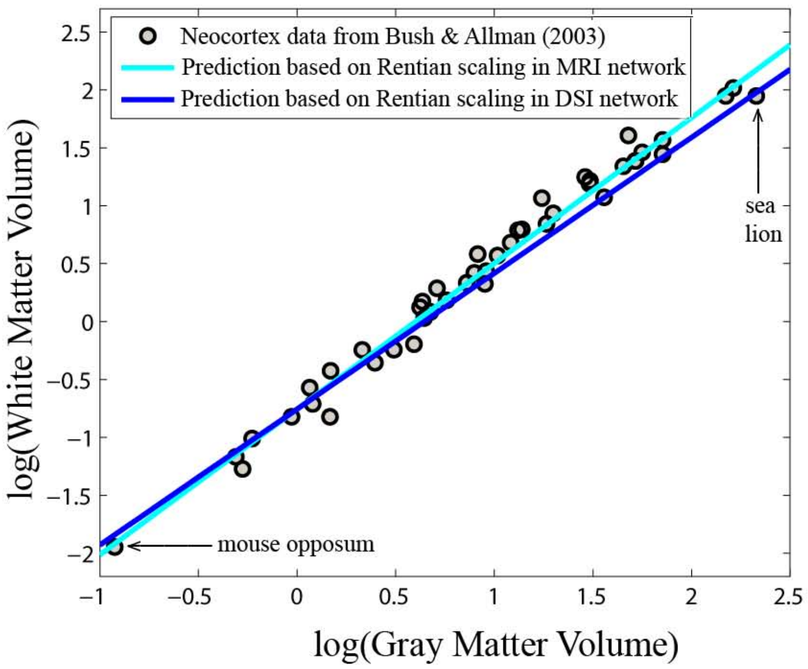


Figure 9.8: Allometric Scaling

What they have potentially neglected is a fundamental, well-known result in the application of Rent's rule, in that it results in a power-law growth in *both* the total number of connections *and* in their average length [39], even though the most-prevalent interconnection is to immediate neighbours [40]. Instead, they have implicitly assumed that the average volume of a 'connection' remains constant, even as the neocortical volume scales over six orders of magnitude. They, instead, propose using an 'alternative interpretation' of Rent's rule that merely considers the scaling of the *total* number of connections (regardless of their length) as following a power-law as the number of nodes increases. However, what they may not have realised is that the true utility of Rent's rule lies in its characterisation of the locality of these connections. This is important because in brains as in VLSI, it is not merely the number of connections that matters, but the *distribution* of lengths that is crucial in determining wiring costs – a property that is independent of their 'alternative interpretation' of total connections alone. Their confusion may perhaps lie with an earlier paper that introduces the alternative interpretation [66], which plugs in the exponents fitted with this 'alternative interpretation' of total connections, rather than of terminals, into length-distribution models that are wholly derived and based on the accepted form of Rent's rule. The original 'alternative interpretation' publication then claims better predictive accuracy, despite violating the underlying assumptions of the models these values are plugged into, to the extent that Donath's model needs to be 'renormalised' to deal with negative numbers of predicted wires, and appears to provide no justification as to why violating these assumptions is acceptable with Davis' model. We should further note that as of present, this 'alternate form' of Rent's rule does not appear to be utilised by other authors other than Bieu et al., and the original authors, although it is cited by some others for altogether different reasons. Bieu et al. then claim that the allometric power-law scaling of neocortical *volumes* (or as interpreted by them, links and nodes – after accounting for a fixed proportion of white matter and grey matter being interconnect) follows this 'alternative interpretation' of Rent's rule as they believe that it appears consistent with other types of computational networks such as Crossbars, Cube-Connected-Cycles and Binary Hypercubes (even though these types of graphs are actually known to *not* follow the accepted form of Rent's rule). Unfortunately, this may indicate misunderstandings by them, both of Rent's rule and what its scaling implications truly are [99].

Recent work by Partzsch et al. [86] also examined *C.elegans* for topological Rentian scaling. They showed that its topological Rentian exponent was higher than shown here (0.82), however they used an old spectral partitioning algorithm that, whilst in 1994 claimed to provide 'the lowest possible p [Rent's exponent]' has since been supplanted by more advanced approaches such as *hMetis* [62]. Indeed, *hMetis* was utilised by Reda [90] in another recent paper examining topological Rentian scaling in biological systems. Reda found that the topological Rent's exponent in *C.elegans* was lower than Partzsch's result, and their result is consistent with both the physical Rent's exponents reported here (as well as the topological ones based on *hMetis*). To the best of our knowledge, ours is the first work to report topological Rentian scaling in *human* neuronal networks derived from neuroimaging. We also believe that ours is the first work to explore *physical* Rentian scaling properties in neuronal networks.

9.7.1 Allometric scaling derivations

A number of models have been developed by Prothero, Changizi, Zhang and Sejnowski to try to explain the observed allometric scaling in white versus grey matter [89, 23, 115]. However, one of the main assumptions governing these models – that there are a constant number of neurons per unit area of cortical surface, has recently been shown to be unrealistic [55]. Moreover, the Changizi, Zhang and Sejnowski models produce a near-perfect $4/3$ scaling exponent between white matter and grey matter (the earlier Prothero model produces an incorrect scaling exponent of close to one), and do not allow for very different scaling exponents, such as is observed in the cerebellum [19] and is likely to be seen in non-mammalian species. Unlike previous models, utilising Rentian self-similarity allows us to cross-check the scaling properties within a brain with the predicted allometric scaling across a range of mammalian species. Furthermore, it allows for differences between the neocortex and cerebellum, and as Dr Bassett has noted – even differences between classes of animals, such as for vertebrates and invertebrates.

9.8 Conclusion

It has been shown in this chapter, that the complete nervous systems of *C.elegans*, as well as the top-level coarse-grained interconnect of human MRI and DSI neocortical datasets exhibit both physical and topological Rentian scaling. Moreover, remarkably, the physical Rentian scaling observed *within* the human datasets yield good predictions for the allometric scaling of white matter to grey matter across mammalian brains. Together, this implies that natural evolution has employed Rentian scaling, both within brains and across evolution, at scales of integration far beyond existing VLSI, as a means of achieving the desired functionality whilst minimising communication costs. For computational systems, this further implies that the self-similarity of Rent's rule is likely to continue to govern both the scaling behaviour and internal communication properties of systems at much larger scales than present today.

Chapter Ten

Conclusions and Future Work

For most of the history of computing, transistors have been expensive, and wires have been cheap. Correspondingly, when analysing costs in software the focus has been on computational complexity, with the act of computing an instruction treated as important, whereas the physical movement of data and instructions was of secondary importance. Thanks to technological scaling, we are entering an altogether different era, one marked by on-chip and off-chip communication energy costs that are many orders of magnitude larger than computational ones, and that will likely continue to get exponentially worse. Crucially, at these technological scales, costs (be they in energy, latency, congestion/wiring, area or performance) grow according to the *physical* distance of data traversal, thus the old High Performance Computing tricks of using high-dimensional topologies no longer works. Instead, the communication locality of algorithms must be exploited and instructions and data carefully positioned in the two (for computer chips) or three dimensions of physical space.

In a communication-centric era of CMPs with thousands of cores, it is thus crucial to understand and exploit the locality of software. It has been shown here, that the theory of locality in the VLSI domain of gates and wires, based on the fractal properties of *Rent's rule*, can be adapted and extended to the CMP domain of software instructions, memory, processor cores and networks-on-chip. This thesis demonstrated experimental evidence of both spatial and temporal Rentian locality, as well as fractal dimensional properties, over a range of benchmarks applications (comprising PARSEC, SPLASH-2 and MiBench suites), with good predictions of hop-length distributions for the CMP domain, and of the expected Rentian relationship between temporal-access times and their Rent's exponent.

In the Network-on-Chip domain, Rentian locality was shown to lead to considerably better technological scaling in traffic – in line with ordinary VLSI growth trends, compared to the unacceptably large growth of commonly used non-local (uniformly random and transpose) traffic models. It was also demonstrated how the hop-length distribution from the Rentian, exponential and uniform-random models could be used to analytically assess alternate designs for properties such as the expected distribution of routing types required, for fault-tolerance characterisation, and even for congestion analysis. Indeed, the type of traffic model was shown to result in large differences for some of these properties, with the Rentian traffic model performing better under most measures, but not all.

Early in the thesis, Rent's rule was generalised from the VLSI domain of wires and gates, to the Network-on-Chip domain of packets, however it was still desirable to generalise beyond this to a domain where instructions, themselves, are placed both spatially across many cores and temporally. Here, communication between such instructions are routed not just spatially, but spatio-temporally, through both a NoC and on-chip/external memory. For this purpose, results were proven allowing Rent's rule to be applied to arbitrary finite-dimensional vector spaces, and then to configuration spaces, including the spatio-temporal domain of software running on CMPs. In fact, it was this Spatio-Temporal Rentian model for CMPs that was used to test for existing experimental evidence of Rentian locality in multicore benchmarks.

The Rentian model of software locality was shown to have many interesting and surprising implications for computer architecture and software. Employing interdependent parallelism between cores results in greatly reduced external I/O bandwidth requirements over the independent parallelism approaches such as simple vector-parallelism, thread-level speculation or current practice of largely independent threaded-parallelism. At the thousand-core era, and for sufficiently large working sets, this I/O reduction was shown to be comparable to that of a thousand fold increase in on-chip memory. Surprisingly, the model also showed an increasing need to move to finer-grain interdependent communication between cores to alleviate external I/O bandwidth demands. Under external-I/O dominated constraints, it also gave results on how best to balance the chip area between the number of cores and on-chip memory, with technological scaling. A model of a 'perfect' on-chip memory was also introduced, that placed an upper-bound on the performance of any scratchpad memory. Surprisingly, even a perfect memory still showed the power-law scaling 'miss' behaviour of caches, albeit with a better scaling exponent.

Moving from computer architecture to the software itself, it was shown that algorithms may be analysed directly for Rentian parameters from first-principles, in addition to the experimental characterisation seen earlier. However, there was a practical problem raised about preserving Rentian scaling for software, in that optimally embedded algorithms that are Rentian, may not themselves compose optimally, resulting in more non-local communication, and thus potentially disrupting Rentian scaling. This strict optimality turned out not to be practically important for Rentian scaling, as it was shown that algorithms can be embedded in a composable/decomposable modular fashion, with random or even worst-case communication at each local level of the hierarchy, whilst preserving the global Rentian scaling properties. This means that global Rentian properties in software can be realised practically, in a modular, composable fashion. This thesis further examined how Rentian properties of an algorithm affect the lower bounds for the spatial communication that is implicit in any memory accesses. Together with the architectural analysis, the analytical Rentian models allow one to characterise the total expected asymptotic cost of both spatial and temporal communication for algorithms, including for memory accesses and external I/O.

Finally, in considering scales of parallel computation beyond existing technology, the brain was examined as a proof-of-existence. It was shown that Rentian scaling is seen in the complete nervous systems of the nematode worm, and for course-grain maps of the human neocortex extracted by neuroimaging. Moreover, the Rentian scaling within the human neocortex was shown to give good predictions for the allometric scaling of white-matter to grey-matter across mammalian neocortices. For the computing domain, this suggests that Rentian scaling is likely to continue to govern communication in computation at far larger scales of integration than seen today.

10.1 Future Work

There are many promising directions that this research opens up. It is likely that many more techniques from the VLSI arena could be adapted for the CMP software domain, in areas of placement, routing and analysis, not just spatially but even temporally - such as spatio-temporal Steiner trees for distributed and high fanout communication. We saw evidence, in Chapter 5, of heterogeneous Rentian behaviour for the complex *lame* benchmark, and although the analysis here assumed homogeneous Rentian behaviour, the theory should be extendable to heterogeneous, time-varying Rent's exponents corresponding to different phases of execution, and even to scale-varying Rent's exponents, where different levels of the hierarchy may exhibit different Rent's exponents.

The derivations for Chapter 7 assumed a simple scalar structure of processing in CMPs, however, more general structures are easily accommodated such as for superscalar processors, which allow extremely fine-grain communication between instructions on execution units within the same core. These could capture more detailed cost functions of communication within core, versus inter-core. Further exploration could also be made in quantitatively characterising cache models versus the ideal Rentian memory model. Alternate cost models could be used to explore the impact of different network-on-chip topologies. Together, these motivate a direct analytical exploration of many architectural design decisions in a CMP.

Noting that the analysis of allometric scaling presented in Chapter 9 was solely for the mammalian neocortex, it would be very interesting to validate the predicted allometric scaling on the cerebellum as well, where it is known to have a different allometric scaling exponent. Excitingly, the generalisation of Rent's rule in Chapter 6 also opens up the possibility of exploring Rentian scaling in many non-computing or biological systems where physical position matters, such as in human interaction networks (related to epidemic networks) as noted in that chapter, but also in genetic regulatory networks, transportation networks, consumption, supply-chain and organisational networks – among many others. In short, it is hoped that this work will inspire others to also apply Rentian locality to perhaps predict interactions in the physical world around them.

Bibliography

[1] *The SPLASH-2 programs: Characterization and methodological considerations.* ACM, 1995.

[2] *The PARSEC Benchmark Suite: Characterization and Architectural Implications*, October 2008.

[3] M. Abeles. *Corticonics: Neural Circuits of the Cerebral Cortex.* Cambridge University Press, 1991.

[4] A. Aggarwal, B. Alpern, A. Chandra, and M. Snir. A model for hierarchical memory. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 305–314, New York, NY, USA, 1987. ACM.

[5] D. Attwell and S. B. Laughlin. An energy budget for signaling in the grey matter of the brain. *J Cereb Blood Flow Metab*, 21(10):1133–1145, Oct 2001.

[6] R. Bagrodia, Mineo Takai, Yu an Chen, Xiang Zeng, and Jay Martin. Parsec: A parallel simulation environment for complex systems. *IEEE Computer*, 31:77–85, 1998.

[7] Arnab Banerjee, Pascal T. Wolkotte, Robert D. Mullins, Simon W. Moore, and Gerard J. M. Smit. An energy and performance exploration of network-on-chip architectures. *IEEE Trans. Very Large Scale Integr. Syst.*, 17(3):319–329, 2009.

[8] Nick Barrow-Williams, Christian Fensch, and Simon Moore. A communication characterization of SPLASH-2 and PARSEC. In *Proceedings of the 2009 International Symposium on Workload Characterization*, October 2009.

[9] Nick Barrow-Williams, Christian Fensch, and Simon Moore. Proximity coherence for chip-multi processors. In *Proceedings of the 2010 International Conference on Parallel Architectures and Compilation Techniques*, 2010.

[10] P. J. Basser. Cable equation for a myelinated axon derived from its microstructure. *Med Biol Eng Comput*, 31 Suppl:S87–S92, Jul 1993.

[11] Danielle S Bassett, Edward Bullmore, Beth A Verchinski, Venkata S Mattay, Daniel R Weinberger, and Andreas Meyer-Lindenberg. Hierarchical organization of human cortical networks in health and schizophrenia. *J Neurosci*, 28(37):9239–9248, Sep 2008.

[12] V. Beiu and W. Ibrahim. Does the brain really outperform Rent's rule? pages 640–643, May 2008.

[13] Gianfranco Bilardi, Carlo Fantozzi, Andrea Pietracaprina, and Geppino Pucci. On the effectiveness of D-BSP as a bridging model of parallel computation. In *In Proceedings of the International Conference on Computer Science, LNCS 2074*, pages 579–588. Springer-Verlag, 2001.

[14] Gianfranco Bilardi and Franco P. Preparata. Horizons of parallel computation. *Journal of Parallel and Distributed Computing*, 27:172–182, 1993.

[15] Gianfranco Bilardi and Franco P. Preparata. Lower bounds to processor-time tradeoffs under bounded-speed message propagation. In *WADS '95: Proceedings of the 4th International Workshop on Algorithms and Data Structures*, pages 1–12, London, UK, 1995. Springer-Verlag.

[16] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. pages 1929 –1934 vol.3, may 1989.

[17] Dirk Brockmann, Lars Hufnagel, and Theo Geisel. The scaling laws of human travel. *Nature*, 439:462–465, 2006.

[18] J. Burkis. Clock tree synthesis for high performance ASICs. Sep 1991.

[19] Eliot C Bush and John M Allman. The scaling of white matter to gray matter in cerebellum and neocortex. *Brain Behav Evol*, 61(1):1–5, 2003.

[20] A.E. Caldwell, Yu Cao, Andrew B. Kahng, F. Koushanfar, Hua Lu, I.L. Markov, M. Oliver, D. Stroobandt, and D. Sylvester. GTX: the MARCO GSRC technology extrapolation system. In *37th Design Automation Conference*, pages 693–698, June 5-9, 2000.

[21] R. F. Cancho, C. Janssen, and R. V. Solé. Topology of technology graphs: small world patterns in electronic circuits. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64(4 Pt 2), October 2001.

[22] Lynn Elliot Cannon. *A cellular computer to implement the kalman filter algorithm*. PhD thesis, Bozeman, MT, USA, 1969.

[23] M. A. Changizi. Principles underlying mammalian neocortical scaling. *Biol Cybern*, 84(3):207–215, Mar 2001.

[24] Bernard Chazelle and Louis Monier. A model of computation for vlsi with related complexity results. *Journal of the ACM*, 32:573–588, 1985.

[25] Chun-Lung Chen and Ge-Ming Chiu. A fault-tolerant routing scheme for meshes with nonconvex faults. *Parallel and Distributed Systems, IEEE Transactions on*, 12(5):467–475, May 2001.

[26] A.A. Chien and Jae H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, pages 268–277, May 19-21, 1992.

[27] Y. Choe, B.H. McCormick, and W. Koh. Network connectivity analysis on the temporally augmented C. elegans web: A pilot study. *Society of Neuroscience Abstracts 30:921.9.*, 2004.

[28] C. K. Chow. Determination of cache's capacity and its matching storage hierarchy. *IEEE Trans. Comput.*, 25(2):157–164, 1976.

[29] P. Christie and D. Stroobandt. The interpretation and application of Rent's rule. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(6):639–648, Dec. 2000.

[30] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[31] G. Concas, M.F. Locci, M. Marchesi, S. Pinna, and I. Turnu. Fractal dimension in software networks. *EPL (Europhysics Letters)*, 76(6):1221–1227, 2006.

[32] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. Logp: Towards a realistic model of parallel computation. pages 1–12, 1993.

[33] A Meyer-Lindenberg D. Weinberger S.W. Moore E Bullmore D. Bassett, D.L. Greenfield. Efficient physical embedding of topologically complex information processing networks in brains and computer circuits. *PLoS Computational Biology*, April 2010.

[34] W. J. Dally. Computer architecture is all about interconnect. In *HPCA Panel*, February 2002.

[35] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, 2001.

[36] J.A. Davis, V.K. De, and J.D. Meindl. A stochastic wire-length distribution for gigascale integration (gsi). i. derivation and validation. *Electron Devices, IEEE Transactions on*, 45(3):580 –589, mar 1998.

[37] R.P. Dick, D.L. Rhodes, and W. Wolf. TGFF: task graphs for free. In *Hardware/Software Codesign, 1998. (CODES/CASHE '98) Proceedings of the Sixth International Workshop on*, pages 97–101, 15-18 March 1998.

[38] Rajeev K. Dokania and Alyssa B. Apsel. Analysis of challenges for on-chip optical interconnects. In *ACM Great Lakes Symposium on VLSI*, pages 275–280, 2009.

[39] W. Donath. Placement and average interconnection lengths of computer logic. *Circuits and Systems, IEEE Transactions on*, 26(4):272 – 277, Apr 1979.

[40] W. E. Donath. Wire length distribution for placements of computer logic. In *IBM Journal of Research and Development, VLSI Circuit Design*, volume 25, page 152, 1981.

[41] D.K. Ferry and W. Porod. Interconnections and architecture for ensembles of microstructures. *Superlattices and Microstructures*, 2(1):41 – 44, 1986.

[42] Steven Fortune and James Wyllie. Parallelism in random access machines. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 114–118, New York, NY, USA, 1978. ACM.

[43] G.C. Fox. *The Use of Physics Concepts in Computation, NPAC Technical Report SCCS-237 in Computation: the Micro and Macro View*. World Scientific, River Edge, NJ, 1992.

[44] Stephen Furber and Andrew Brown. Biologically-inspired massively-parallel architectures - computing beyond a million processors. *International Conference on Application of Concurrency to System Design*, 0:3–12, 2009.

[45] D. Greenfield, A. Banerjee, J.-G. Lee, and S. Moore. Implications of Rent's rule for NoC design and its fault-tolerance. In *Proc. First International Symposium on Networks-on-Chip NOCS 2007*, pages 283–294, 2007.

[46] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Proc. WWC-4 Workload Characterization 2001 IEEE International Workshop on*, pages 3–14, 2 Dec. 2001.

[47] Patric Hagmann, Leila Cammoun, Xavier Gigandet, Reto Meuli, Christopher J Honey, Van J Wedeen, and Olaf Sporns. Mapping the structural core of human cerebral cortex. *PLoS Biol*, 6(7):e159, Jul 2008.

[48] M. Hamdi and S.W. Song. On embedding various networks into the hypercube using matrix transformations. *Parallel Processing Symposium, International*, 0:650, 1996.

[49] A. Hartstein, V. Srinivasan, T. R. Puzak, and P. G. Emma. Cache miss behavior: is it sqrt 2? In *CF '06: Proceedings of the 3rd conference on Computing frontiers*, pages 313–320, New York, NY, USA, 2006. ACM.

[50] Mikhail Haurylau, Associate Member, Guoqing Chen, Hui Chen, Jidong Zhang, Nicholas A. Nelson, Student Member, David H. Albonesi, Senior Member, and Eby G. Friedman. Optical interconnect roadmap: challenges and critical directions. *IEEE J. Sel. Top. Quantum Electron. (2006)*, 40:434–446, 2005.

[51] Erno Salminen Heikki Orsila, Tero Kangas and Timo D. Hamalainen. Parameterizing simulated annealing for distributing task graphs on multiprocessor SoCs. In *International Symposium on System-on-Chip 2006, Tampere, Finland*, pages pp. 73–76, November 2006.

[52] Wim Heirman, Joni Dambre, Dirk Stroobandt, and Jan Van Campenhout. Rent's rule and parallel programs: characterizing network traffic behavior. In *SLIP '08:*

Proceedings of the 2008 international workshop on System level interconnect prediction, pages 87–94, New York, NY, USA, 2008. ACM.

[53] Jim Held, Jerry Bautista, and Sean Koehl. From a few cores to many: A tera-scale computing research overview. Technical report, 2006.

[54] Richard S. Hemmert. Poisson process and integrated circuit yield prediction. *Solid-State Electronics*, 24(6):511 – 515, 1981.

[55] Suzana Herculano-Houzel, Christine E Collins, Peiyan Wong, Jon H Kaas, and Roberto Lent. The basic nonuniformity of the cerebral cortex. *Proc Natl Acad Sci U S A*, 105(34):12593–12598, Aug 2008.

[56] Mark D. Hill and Michael R. Marty. Amdahlś law in the multicore era. *IEEE COMPUTER*, 2008.

[57] Jingcao Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Design Automation Conference. Proceedings of the ASP-DAC 2003. Asia and South Pacific*, pages 233–239, 21-24 Jan. 2003.

[58] H. Ito, J. Seita, T. Ishii, H. Sugita, K. Okada, and K. Masu. A low-latency and high-power-efficient on-chip lvds transmission line interconnect for an rc interconnect alternative. In *International Interconnect Technology Conference, IEEE 2007*, pages 193 –195, june 2007.

[59] ITRS. International technology roadmap for semiconductors - 2007 edition: Assembly and packaging. Technical report, 2007.

[60] James W. Joyner, Payman Zarkesh-Ha, and James D. Meindl. Global interconnect design in a three-dimensional system-on-a-chip. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(4):367–372, 2004.

[61] Marcus Kaiser and Claus C Hilgetag. Nonoptimal component placement, but short processing paths, due to long-distance projections in neural systems. *PLoS Comput Biol*, 2(7):e95, Jul 2006.

[62] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in VLSI domain. *IEEE Trans. VLSI Syst.*, 7(1):69–79, 1999.

[63] Jongman Kim, C. Nicopoulos, and Dongkook Park. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *33rd International Symposium on Computer Architecture*, pages 4–15, 17-21 June 2006.

[64] B.S. Landman and R.L. Russo. On a pin versus block relationship for partitions of logic graphs. *IEEE Trans. Comput.*, 20(12):1469–1479, Dec. 1971.

[65] H.-W. Lang, M. Schimmler, H. Schmeck, and H. Schroder. Systolic sorting on a mesh-connected network. *IEEE Transactions on Computers*, 34:652–658, 1985.

[66] M.Y. Lanzerotti, G. Fiorenza, and R.A. Rand. Interpretation of rent's rule for ultralarge-scale integrated circuit designs, with an application to wirelength distribution models. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(12):1330 – 1347, dec. 2004.

[67] Charles Eric Leiserson. *Area-Efficient VLSI Computation*. MIT Press, Cambridge, MA, USA, 1997.

[68] Jason P Lerch, Keith Worsley, W. Philip Shaw, Deanna K Greenstein, Rhoshel K Lenroot, Jay Giedd, and Alan C Evans. Mapping anatomical correlations across cerebral cortex (MACACC) using cortical thickness from MRI. *Neuroimage*, 31(3):993–1003, Jul 2006.

[69] Feihui Li, M. Kandemir, and I. Kolcu. Exploiting software pipelining for network-on-chip architectures. In *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*, volume 00, 2-3 March 2006.

[70] S.-W. Liao, Z. Du, G. Wu, and G.-Y. Lueh. A code generation algorithm for affine partitioning framework. In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, volume 2, 20-22 July 2005.

[71] Amy W. Lim, Gerald I. Cheong, and Monica S. Lam. An affine partitioning algorithm to maximize parallelism and minimize communication. In *International Conference on Supercomputing*, pages 228–237, 1999.

[72] C-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, and K. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In *PLDI '05: Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, pages 190–200, New York, NY, USA, 2005. ACM.

[73] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, Feb. 2002.

[74] M. Marklund, G. Brodin, L. Stenflo, and C. S. Liu. New quantum limits in plasmonic devices. *EPL (Europhysics Letters)*, 84(1):17006, 2008.

[75] A. Mineyama, H. Ito, T. Ishii, K. Okada, and K. Masu. LVDS-type on-chip transmision line interconnect with passive equalizers in 90 nm CMOS process. In *ASP-DAC '08: Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 97–98, Los Alamitos, CA, USA, 2008. IEEE Computer Society Press.

[76] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*, pages 188–197, 19-23 June 2004.

[77] A. Mycroft. Programming language design and analysis motivated by hardware evolution. In *SAS*, pages 18–33, 2007.

[78] M E Newman. Modularity and community structure in networks. *Proc Natl Acad Sci U S A*, 103(23):8577–8582, June 2006.

[79] L.M. Ni and P.K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, Feb. 1993.

[80] Jeremy E Niven and Simon B Laughlin. Energy limitation as a selective pressure on the evolution of sensory systems. *J Exp Biol*, 211(Pt 11):1792–1804, Jun 2008.

[81] Kunle Olukotun, Basem A. Nayfeh, Lance Hammond, Ken Wilson, and Kun-yung Chang. The case for a single-chip multiprocessor. *SIGOPS Oper. Syst. Rev.*, 30(5):2–11, 1996.

[82] G. Ottoni, R. Rangan, A. Stoler, and D.I. August. Automatic thread extraction with decoupled software pipelining. In *Microarchitecture, 2005. MICRO-38. Proceedings. 38th Annual IEEE/ACM International Symposium on*, page 12pp., 12-16 Nov. 2005.

[83] Haldun M. Ozaktas. Paradigms of connectivity for computer circuits and networks. *Optical Engineering*, 31(7):1563–1567, 1992.

[84] Haldun M. Ozaktas. Information flow and interconnections in computing: extensions and applications of rent's rule. *Journal of Parallel and Distributed Computing*, 64(12):1360–1370, December 2004.

[85] Gajinder Panesar, Daniel Towner, Andrew Duller, Alan Gray, and Will Robbins. Deterministic parallel processing. *Int. J. Parallel Program.*, 34(4):323–341, 2006.

[86] J. Partzsch and R. Schuffny. On the routing complexity of neural network models – Rent's Rule revisited. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 595–600, 2009.

[87] Franco Preparata and Gianfranco Bilardi. Processor-time tradeoffs under bounded-speed message propagation: Part i, upper bounds. *Theory of Computing Systems*, 32:531–559, 1995.

[88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes: the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1986.

[89] J. Prothero. Cortical scaling in mammals: a repeating units model. *J Hirnforsch*, 38(2):195–207, 1997.

[90] Sherief Reda. Using circuit structural analysis techniques for networks in systems biology. In *SLIP '09: Proceedings of the 11th international workshop on System level interconnect prediction*, pages 37–44, New York, NY, USA, 2009. ACM.

[91] Hernán D. Rozenfeld, Chaoming Song, and Hernán A. Makse. Small-world to fractal transition in complex networks: A renormalization group approach. *Physical Review Letters*, 104(2):025701+, Jan 2010.

[92] R. Sarpeshkar. Analog versus digital: extrapolating from electronics to neurobiology. *Neural Comput*, 10(7):1601–1638, Oct 1998.

[93] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: A many-core x86 architecture for visual computing. *ACM Trans. Graph.*, 27(3):1–15, 2008.

[94] Herbert A. Simon. The architecture of complexity. In *Proceedings of the American Philosophical Society*, pages 467–482, 1962.

[95] Chaoming Song, Shlomo Havlin, and Hernan A. Makse. Self-similarity of complex networks. *Nature*, 433(7024):392–395, January 2005.

[96] J.G. Steffan, J.G. Steffan, and T.C. Mowry. The potential for using thread-level data speculation to facilitate automatic parallelization. In T.C. Mowry, editor, *Proc. Fourth International Symposium on High-Performance Computer Architecture*, pages 2–13, 1998.

[97] D. Stroobandt. On an efficient method for estimating the interconnection complexity of designs and on the existence of region III in Rent's rule. In *VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on*, pages 330–331, 4-6 March 1999.

[98] D. Stroobandt. Recent advances in system-level interconnect prediction. *IEEE Circuits and Systems Society Newsletter*, 11(4):1; 4–20; 48, 2000.

[99] D. Stroobandt. *A priori Wire Length Estimates for Digital Design*. Kluwer Academic Publishers, 2001.

[100] Dirk Stroobandt. A priori wire length distribution models with multiterminal nets. *IEEE Trans. VLSI Syst.*, 11(1):35–43, 2003.

[101] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The Raw microprocessor: a computational fabric for software circuits and general-purpose programs. *Micro, IEEE*, 22(2):25–35, March-April 2002.

[102] M.B. Taylor, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald,

H. Hoffmann, P. Johnson, and J. Kim. Evaluation of the Raw microprocessor: an exposed-wire-delay architecture for ILP and streams. In *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pages 2–13, 19-23 June 2004.

[103] M.D. Taylor, W. Lee, S.P. Amarasinghe, and A. Agarwal. Scalar operand networks. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):145–162, Feb 2005.

[104] C. D. Thompson. Area-time complexity for VLSI. In *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 81–88, New York, NY, USA, 1979. ACM.

[105] Pilar de la Torre and Clyde P. Kruskal. Submachine locality in the bulk synchronous setting (extended abstract). In *Euro-Par '96: Proceedings of the Second International Euro-Par Conference on Parallel Processing-Volume II*, pages 352–358, London, UK, 1996. Springer-Verlag.

[106] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.

[107] Leslie G. Valiant. A bridging model for multi-core computing. In *ESA*, pages 13–28, 2008.

[108] G.V. Varatkar and R. Marculescu. On-chip traffic modeling and synthesis for MPEG-2 video applications. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(1):108–119, Jan. 2004.

[109] P. Verplaetse, J. Dambre, D. Stroobandt, and J. Van Campenhout. On Partitioning vs. Placement Rent Properties. In *In Proc. of Intl. Workshop on System-Level Interconnect Prediction*, pages 33–40, 2001.

[110] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, Jun 1998.

[111] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C-C. Miao, J.F. Brown III, and A. Agarwal. On-chip interconnection architecture of the Tile Processor. *IEEE Micro*, 27(5):15–31, 2007.

[112] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The Structure of the Nervous System of the Nematode Caenorhabditis elegans. *Royal Society of London Philosophical Transactions Series B*, 314:1–340, November 1986.

[113] A.C. Yao. Some complexity questions related to distributed computing. In *Symposium on the Theory of Computing*, 1979.

[114] Payman Zarkesh-ha, Jeffrey A. Davis, William Loh, and James D. Meindl. Prediction of interconnect fan-out distribution using Rent's rule. In *In International*

Workshop on System-Level Interconnect Prediction, pages 107–112. ACM Press, 2000.

[115] K. Zhang and T. J. Sejnowski. A universal scaling law between gray matter and white matter of cerebral cortex. *Proc Natl Acad Sci U S A*, 97(10):5621–5626, May 2000.

[116] Rongtian Zhang, Kaushik Roy, K Koh, and David B. Janes. Stochastic wire-length and delay distributions of 3-dimensional circuits. In *In International Conference on Computer-Aided Design*, pages 208–213. IEEE, 2000.

Appendix A

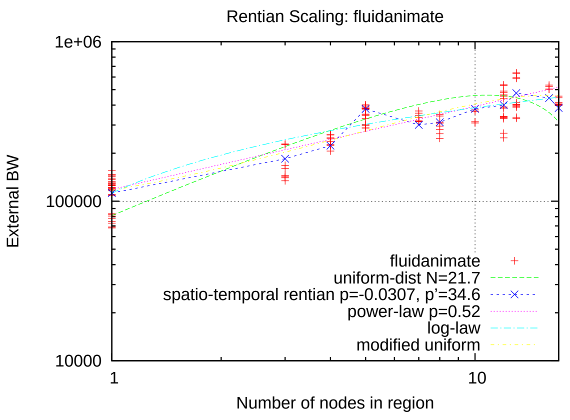
Spatial Scaling: CMP Software

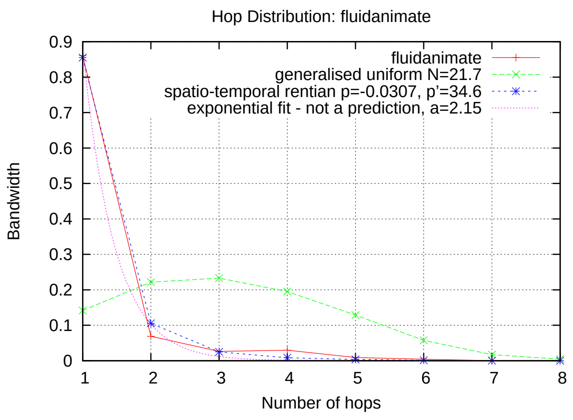
The following plots for SPLASH-2 and PARSEC benchmark applications show the bandwidth versus node distribution on the left, followed by the hop-distributions on the right. These are based on shared-memory traces by Barrow-Williams et al [8] running on a 32-core CMP Simics-based simulator [73].

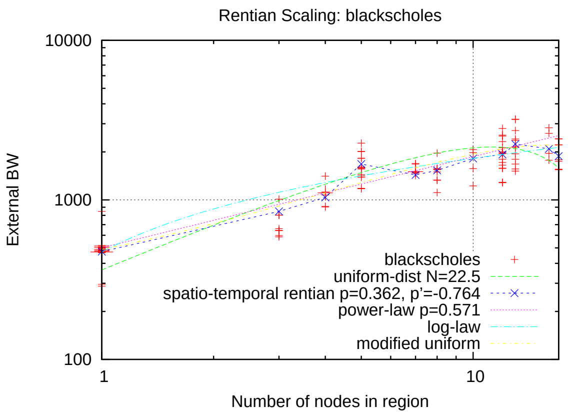
The spatial placement is done by a simulated annealing algorithm, implemented by the author. Its only goal is to minimise aggregate bandwidth (i.e. sum of all communication bandwidths weighted by the distance of the communication).

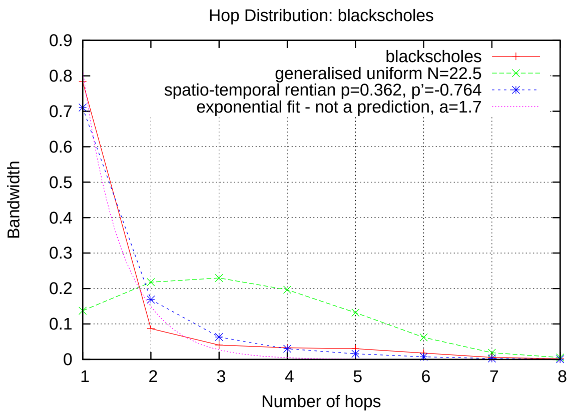
The left-hand side plots show multiple two-free-parameter models fitted to the data. The right-hand side shows the predicted distance distributions for the generalised uniform and spatio-temporal Rentian models based on the fit on the left-hand-side plots. These are the only two models that have associated distance-distribution predictions.

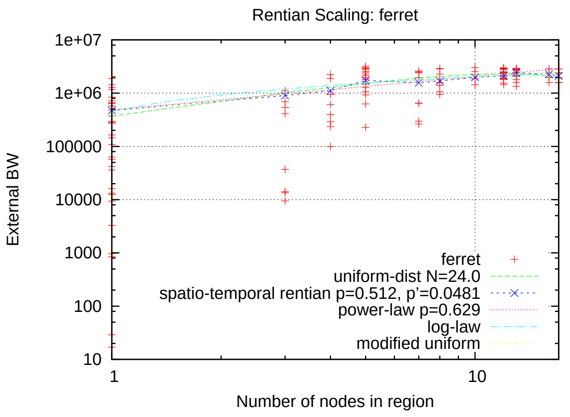
The benchmarks are sorted according to the fitted spatio-temporal Rent's exponent. This means they should be sorted from greatest (predicted) locality to least (predicted) locality.

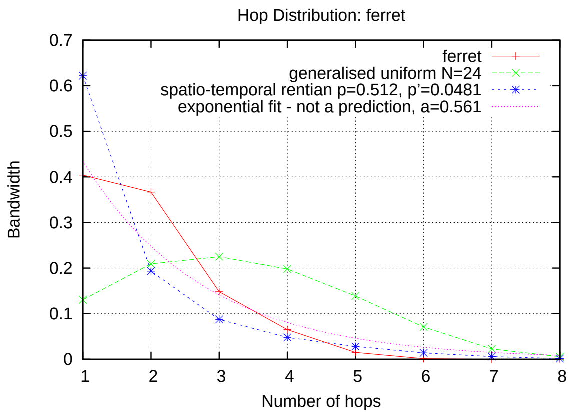

Rentian Scaling: fluidanimate
1e+06
100000
10000
1
10
External BW
Number of nodes in region
fluidanimate
uniform-dist N=21.7
spatio-temporal rentian p=-0.0307, p'=34.6
power-law p=0.52
log-law
modified uniform

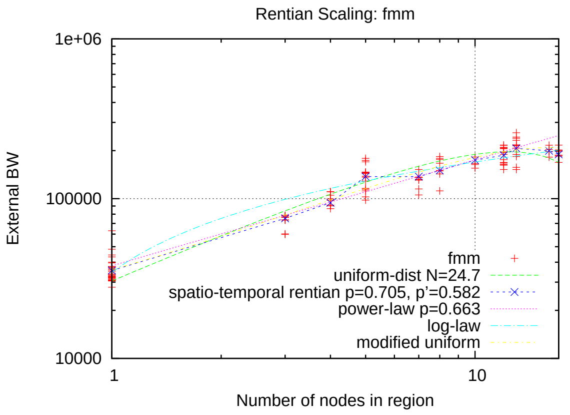

Hop Distribution: fluidanimate
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
Bandwidth
Number of hops
fluidanimate
generalised uniform N=21.7
spatio-temporal rentian p=-0.0307, p'=34.6
exponential fit - not a prediction, a=2.15

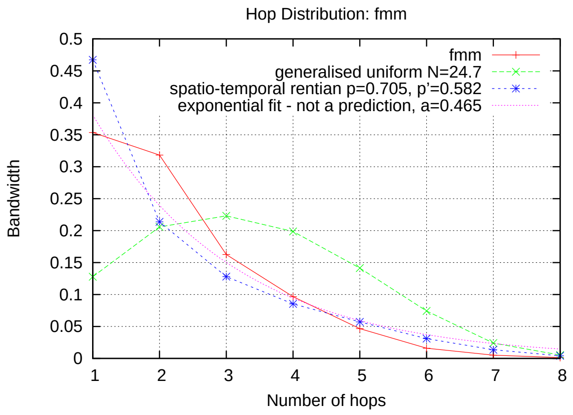

Rentian Scaling: blackscholes
10000
1000
100
1
10
External BW
Number of nodes in region
blackscholes
uniform-dist N=22.5
spatio-temporal rentian p=0.362, p'=-0.764
power-law p=0.571
log-law
modified uniform

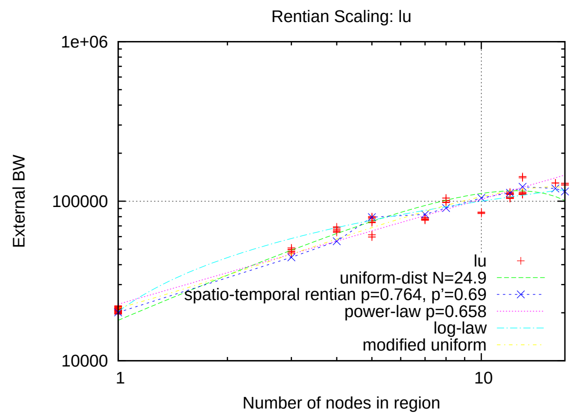

Hop Distribution: blackscholes
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
Bandwidth
Number of hops
blackscholes
generalised uniform N=22.5
spatio-temporal rentian p=0.362, p'=-0.764
exponential fit - not a prediction, a=1.7

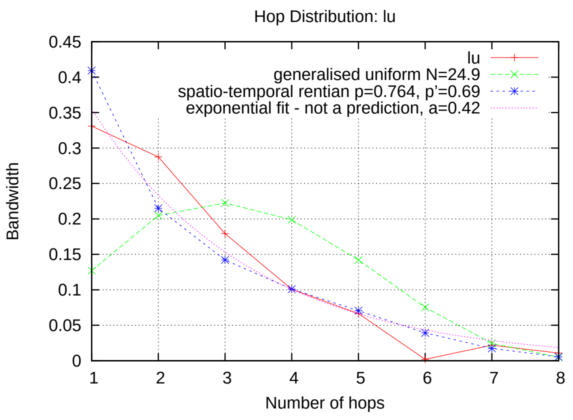

Rentian Scaling: ferret
1e+07
1e+06
100000
10000
1000
100
10
1
10
External BW
Number of nodes in region
ferret
uniform-dist N=24.0
spatio-temporal rentian p=0.512, p'=0.0481
power-law p=0.629
log-law
modified uniform

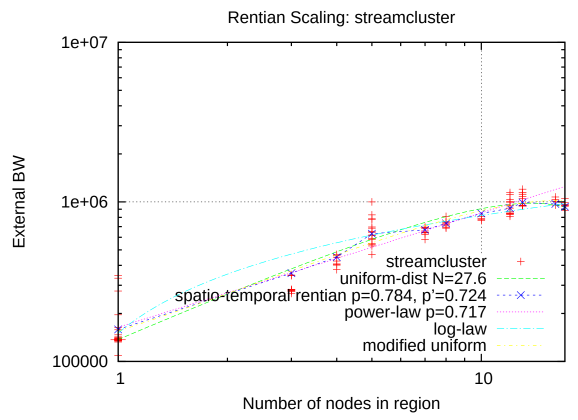

Hop Distribution: ferret
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
Bandwidth
Number of hops
ferret
generalised uniform N=24
spatio-temporal rentian p=0.512, p'=0.0481
exponential fit - not a prediction, a=0.561

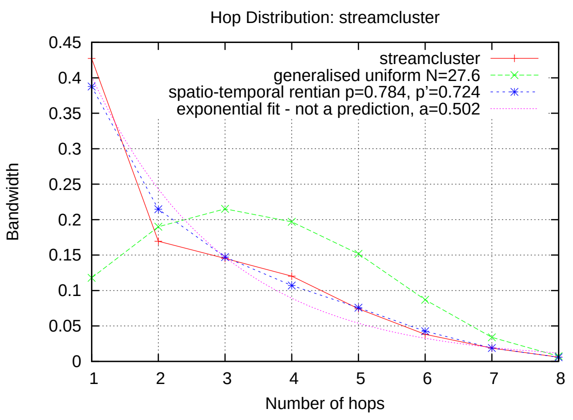

Rentian Scaling: fmm
1e+06
100000
10000
1
10
External BW
Number of nodes in region
fmm
uniform-dist N=24.7
spatio-temporal rentian p=0.705, p'=0.582
power-law p=0.663
log-law
modified uniform

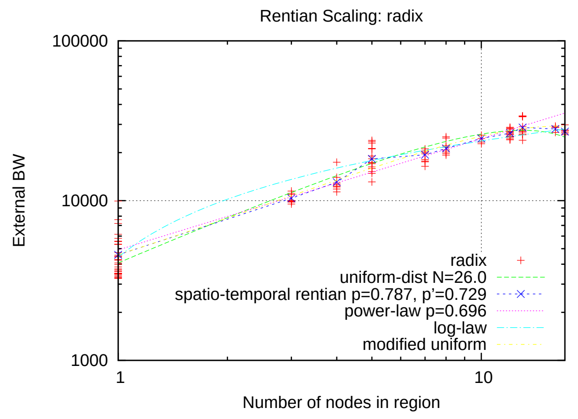

Hop Distribution: fmm
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
Bandwidth
Number of hops
fmm
generalised uniform N=24.7
spatio-temporal rentian p=0.705, p'=0.582
exponential fit - not a prediction, a=0.465

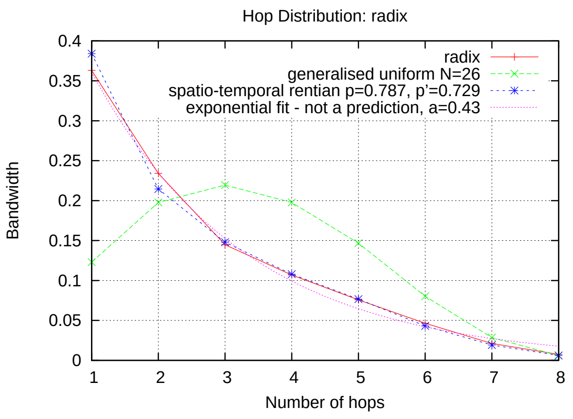

Rentian Scaling: lu
External BW
Number of nodes in region
1e+06
100000
10000
1
10
lu
uniform-dist N=24.9
spatio-temporal rentian p=0.764, p'=0.69
power-law p=0.658
log-law
modified uniform

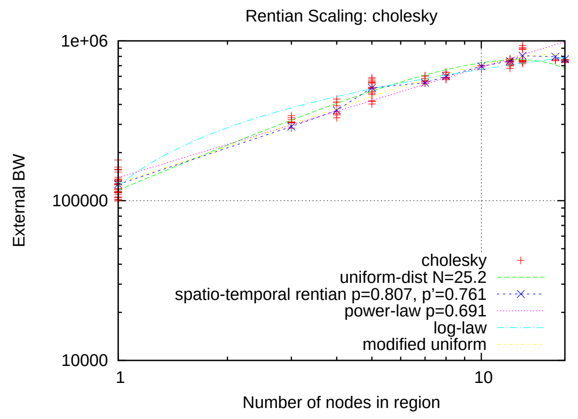

Hop Distribution: lu
Bandwidth
Number of hops
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
lu
generalised uniform N=24.9
spatio-temporal rentian p=0.764, p'=0.69
exponential fit - not a prediction, a=0.42

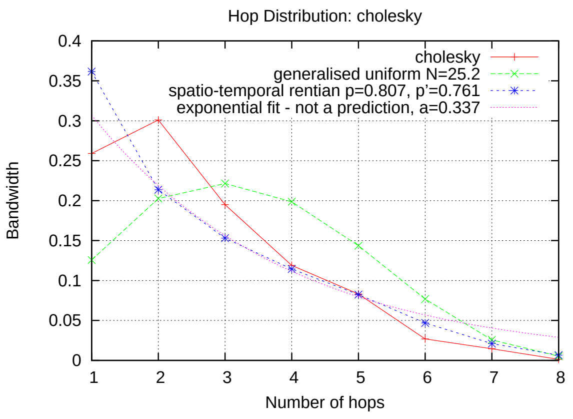

Rentian Scaling: streamcluster
External BW
Number of nodes in region
1e+07
1e+06
100000
1
10
streamcluster
uniform-dist N=27.6
spatio-temporal rentian p=0.784, p'=0.724
power-law p=0.717
log-law
modified uniform

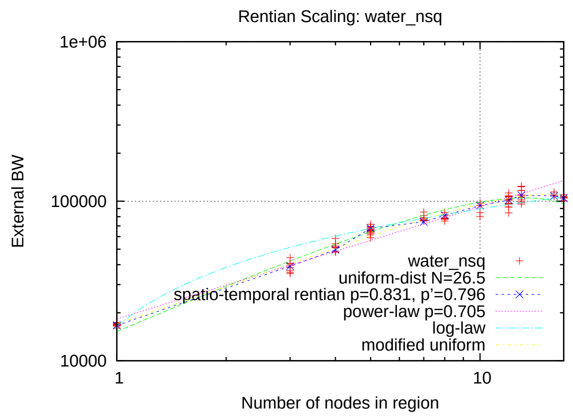

Hop Distribution: streamcluster
Bandwidth
Number of hops
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
streamcluster
generalised uniform N=27.6
spatio-temporal rentian p=0.784, p'=0.724
exponential fit - not a prediction, a=0.502

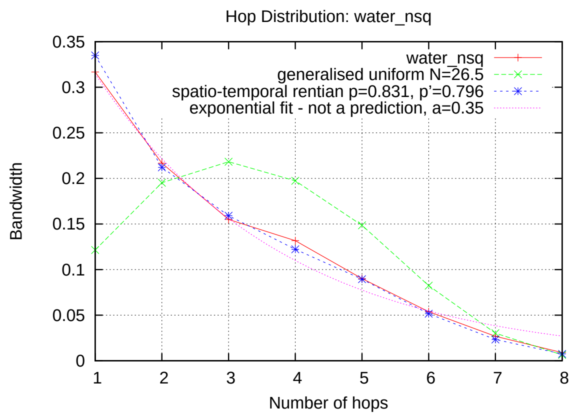

Rentian Scaling: radix
External BW
Number of nodes in region
100000
10000
1000
1
10
radix
uniform-dist N=26.0
spatio-temporal rentian p=0.787, p'=0.729
power-law p=0.696
log-law
modified uniform

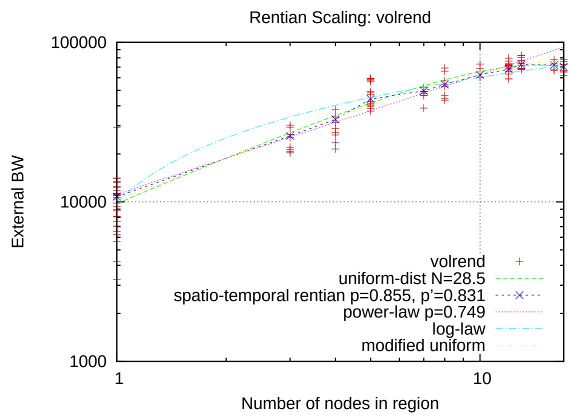

Hop Distribution: radix
Bandwidth
Number of hops
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
radix
generalised uniform N=26
spatio-temporal rentian p=0.787, p'=0.729
exponential fit - not a prediction, a=0.43

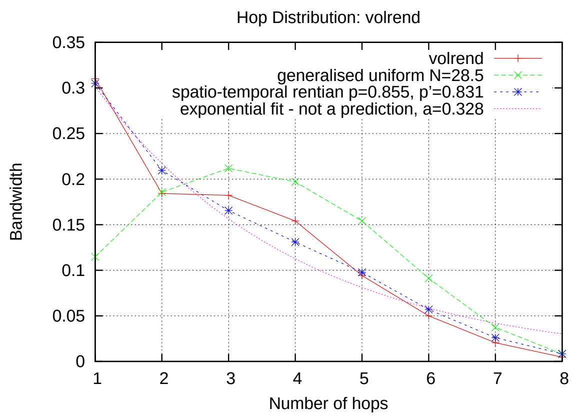

Rentian Scaling: cholesky
External BW
Number of nodes in region
1e+06
100000
10000
1
10
cholesky
uniform-dist N=25.2
spatio-temporal rentian p=0.807, p'=0.761
power-law p=0.691
log-law
modified uniform

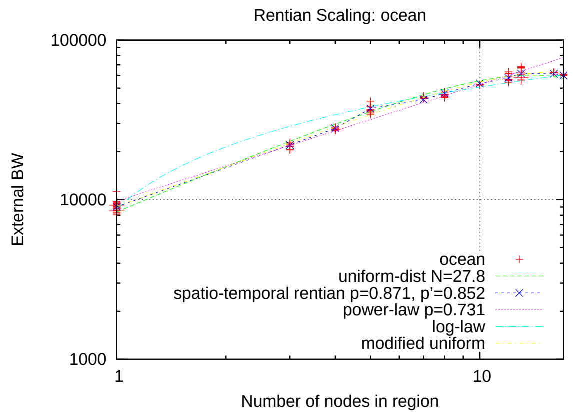

Hop Distribution: cholesky
Bandwidth
Number of hops
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
cholesky
generalised uniform N=25.2
spatio-temporal rentian p=0.807, p'=0.761
exponential fit - not a prediction, a=0.337

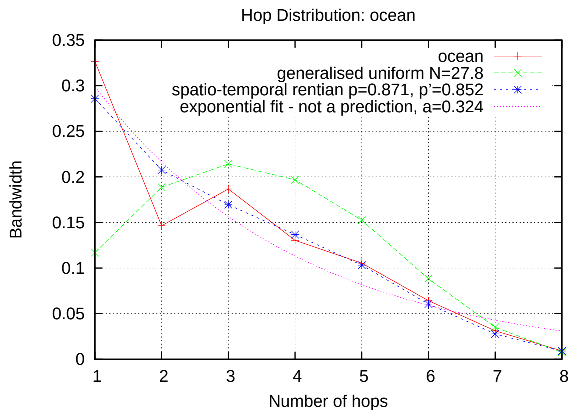

Rentian Scaling: water_nsq
External BW
Number of nodes in region
water_nsq
uniform-dist N=26.5
spatio-temporal rentian p=0.831, p'=0.796
power-law p=0.705
log-law
modified uniform

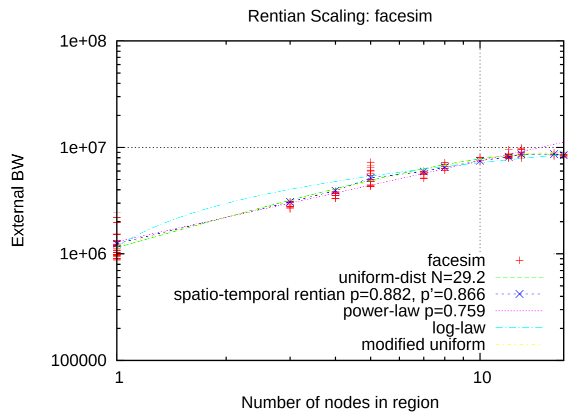

Hop Distribution: water_nsq
Bandwidth
Number of hops
water_nsq
generalised uniform N=26.5
spatio-temporal rentian p=0.831, p'=0.796
exponential fit - not a prediction, a=0.35

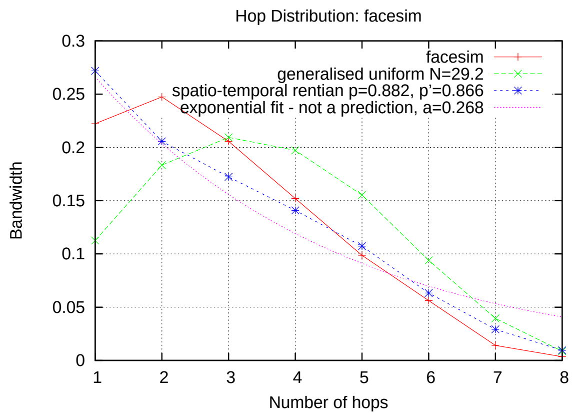

Rentian Scaling: volrend
External BW
Number of nodes in region
volrend
uniform-dist N=28.5
spatio-temporal rentian p=0.855, p'=0.831
power-law p=0.749
log-law
modified uniform

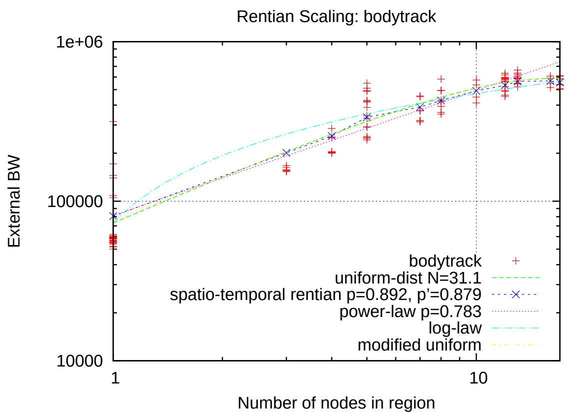

Hop Distribution: volrend
Bandwidth
Number of hops
volrend
generalised uniform N=28.5
spatio-temporal rentian p=0.855, p'=0.831
exponential fit - not a prediction, a=0.328

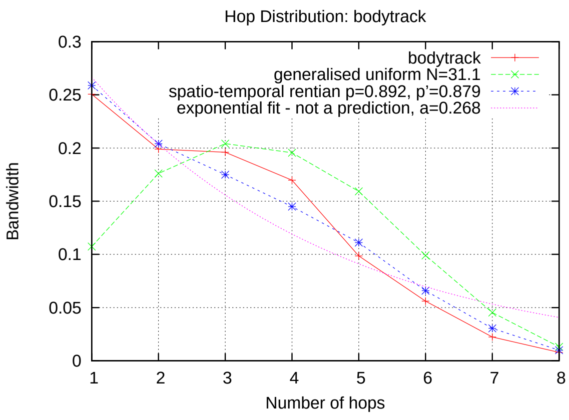

Rentian Scaling: ocean
External BW
Number of nodes in region
ocean
uniform-dist N=27.8
spatio-temporal rentian p=0.871, p'=0.852
power-law p=0.731
log-law
modified uniform

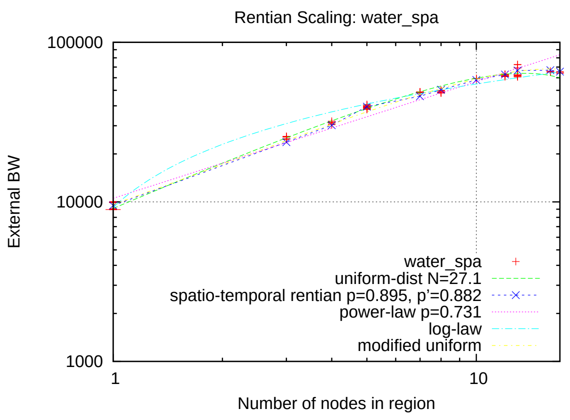

Hop Distribution: ocean
Bandwidth
Number of hops
ocean
generalised uniform N=27.8
spatio-temporal rentian p=0.871, p'=0.852
exponential fit - not a prediction, a=0.324

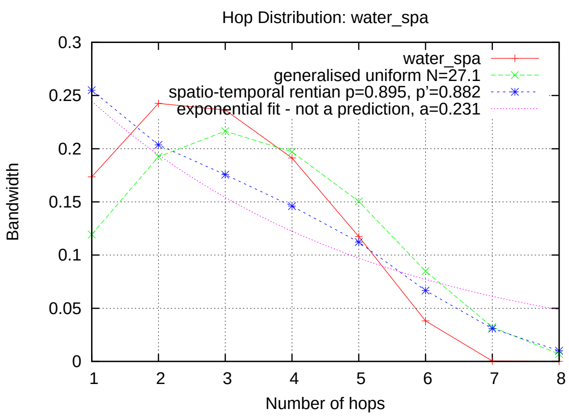

Rentian Scaling: facesim
External BW
Number of nodes in region
facesim
uniform-dist N=29.2
spatio-temporal rentian p=0.882, p'=0.866
power-law p=0.759
log-law
modified uniform

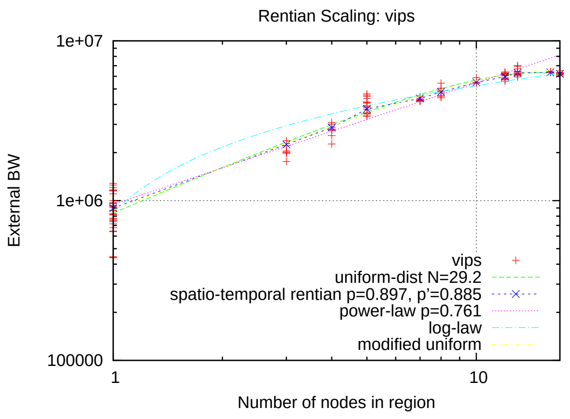

Hop Distribution: facesim
Bandwidth
Number of hops
facesim
generalised uniform N=29.2
spatio-temporal rentian p=0.882, p'=0.866
exponential fit - not a prediction, a=0.268

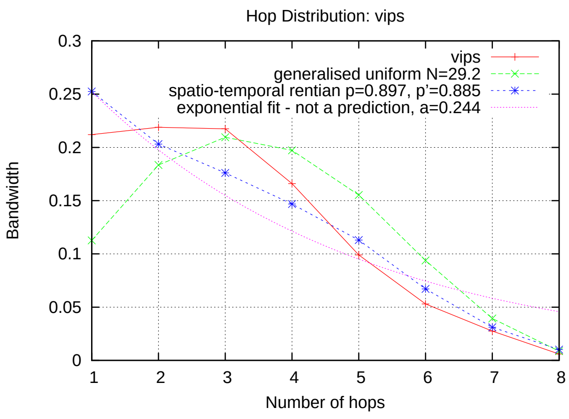

Rentian Scaling: bodytrack
External BW
Number of nodes in region
1e+06
100000
10000
1
10
bodytrack
uniform-dist N=31.1
spatio-temporal rentian p=0.892, p'=0.879
power-law p=0.783
log-law
modified uniform

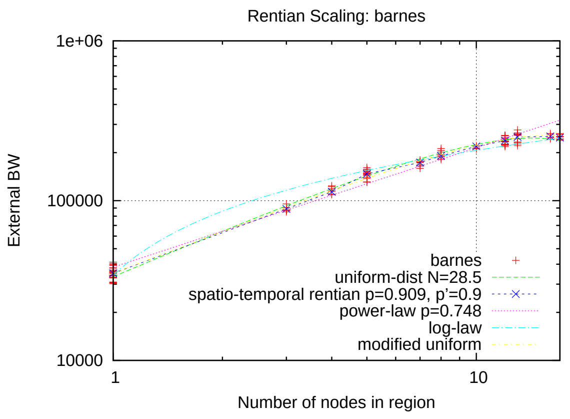

Hop Distribution: bodytrack
Bandwidth
Number of hops
bodytrack
generalised uniform N=31.1
spatio-temporal rentian p=0.892, p'=0.879
exponential fit - not a prediction, a=0.268

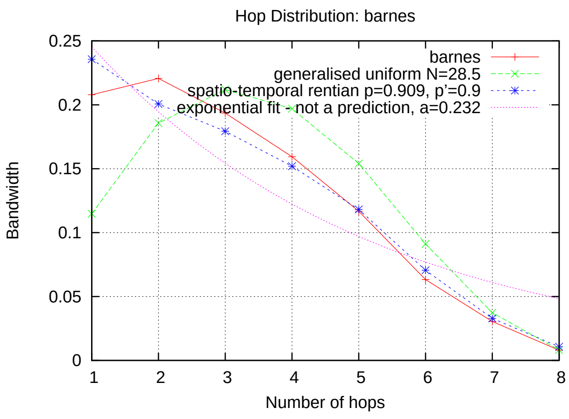

Rentian Scaling: water_spa
External BW
Number of nodes in region
100000
10000
1000
1
10
water_spa
uniform-dist N=27.1
spatio-temporal rentian p=0.895, p'=0.882
power-law p=0.731
log-law
modified uniform

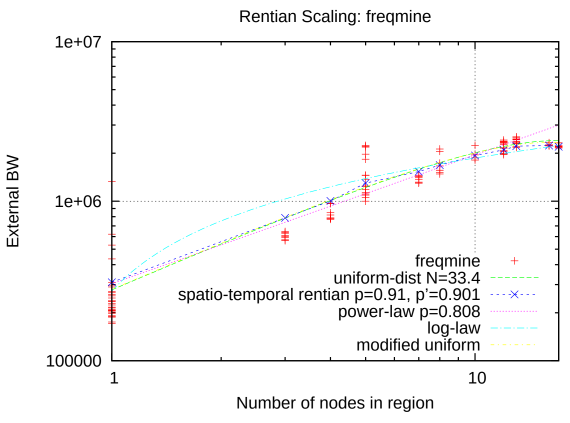

Hop Distribution: water_spa
Bandwidth
Number of hops
water_spa
generalised uniform N=27.1
spatio-temporal rentian p=0.895, p'=0.882
exponential fit - not a prediction, a=0.231

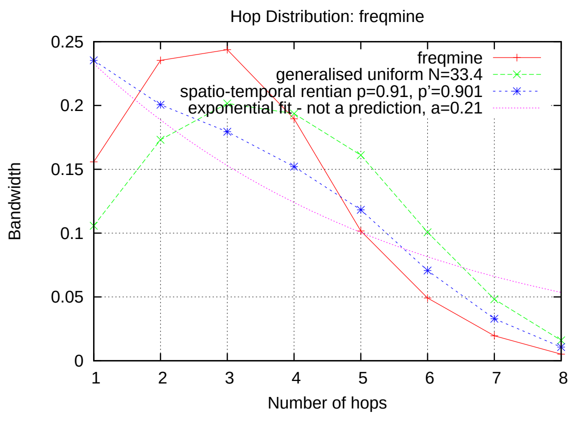

Rentian Scaling: vips
External BW
Number of nodes in region
1e+07
1e+06
100000
1
10
vips
uniform-dist N=29.2
spatio-temporal rentian p=0.897, p'=0.885
power-law p=0.761
log-law
modified uniform

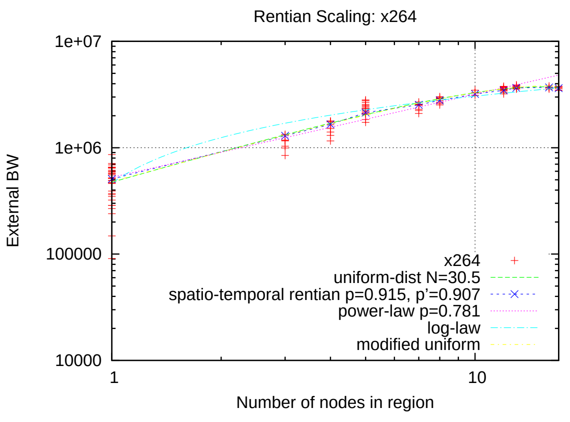

Hop Distribution: vips
Bandwidth
Number of hops
vips
generalised uniform N=29.2
spatio-temporal rentian p=0.897, p'=0.885
exponential fit - not a prediction, a=0.244

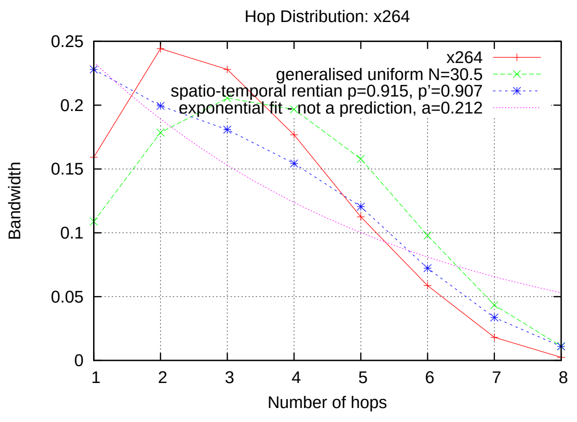

Rentian Scaling: barnes
External BW
Number of nodes in region
1e+06
100000
10000
1
10
barnes
uniform-dist N=28.5
spatio-temporal rentian p=0.909, p'=0.9
power-law p=0.748
log-law
modified uniform

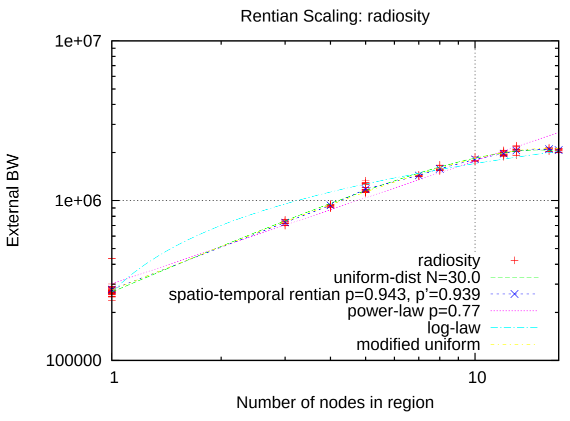

Hop Distribution: barnes
Bandwidth
Number of hops
barnes
generalised uniform N=28.5
spatio-temporal rentian p=0.909, p'=0.9
exponential fit - not a prediction, a=0.232

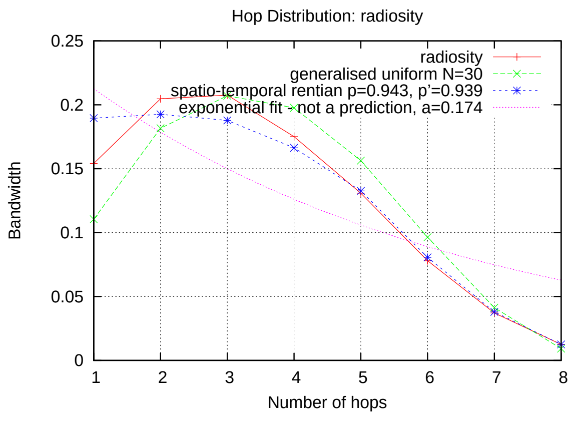

Rentian Scaling: freqmine
1e+07
1e+06
100000
External BW
1
10
Number of nodes in region
freqmine
uniform-dist N=33.4
spatio-temporal rentian p=0.91, p'=0.901
power-law p=0.808
log-law
modified uniform

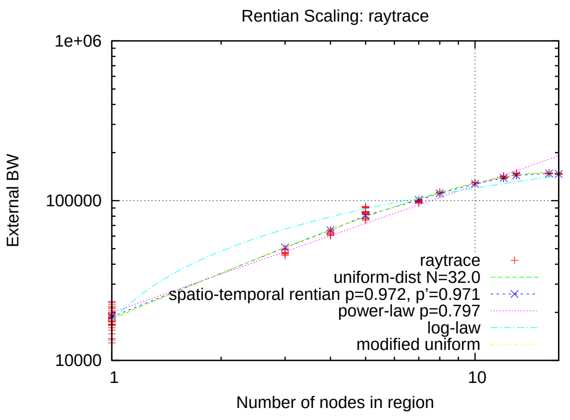

Hop Distribution: freqmine
0.25
0.2
0.15
0.1
0.05
0
Bandwidth
1
2
3
4
5
6
7
8
Number of hops
freqmine
generalised uniform N=33.4
spatio-temporal rentian p=0.91, p'=0.901
exponential fit - not a prediction, a=0.21

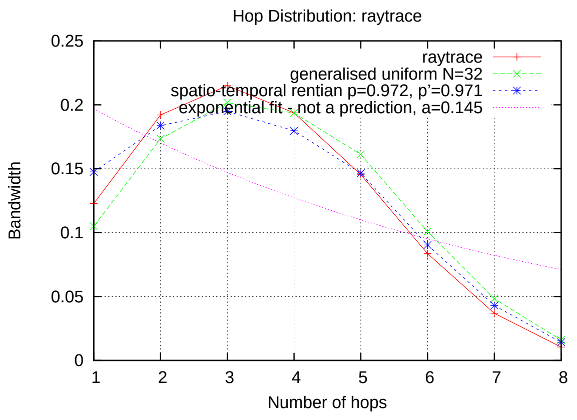

Rentian Scaling: x264
1e+07
1e+06
100000
10000
External BW
1
10
Number of nodes in region
x264
uniform-dist N=30.5
spatio-temporal rentian p=0.915, p'=0.907
power-law p=0.781
log-law
modified uniform

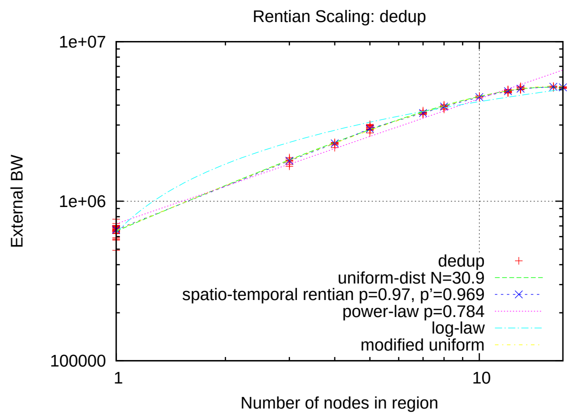

Hop Distribution: x264
0.25
0.2
0.15
0.1
0.05
0
Bandwidth
1
2
3
4
5
6
7
8
Number of hops
x264
generalised uniform N=30.5
spatio-temporal rentian p=0.915, p'=0.907
exponential fit - not a prediction, a=0.212

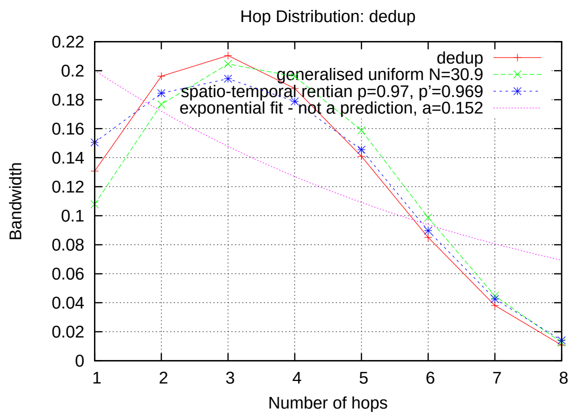

Rentian Scaling: radiosity
1e+07
1e+06
100000
External BW
1
10
Number of nodes in region
radiosity
uniform-dist N=30.0
spatio-temporal rentian p=0.943, p'=0.939
power-law p=0.77
log-law
modified uniform

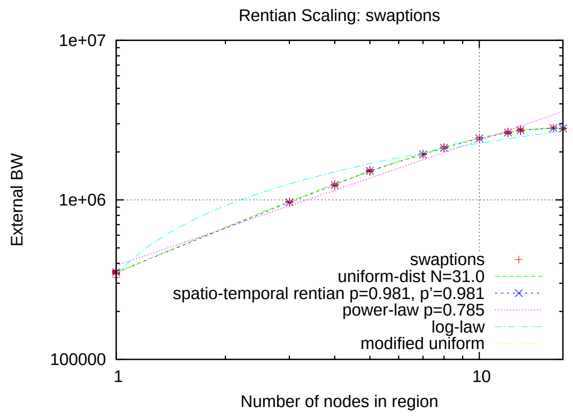

Hop Distribution: radiosity
0.25
0.2
0.15
0.1
0.05
0
Bandwidth
1
2
3
4
5
6
7
8
Number of hops
radiosity
generalised uniform N=30
spatio-temporal rentian p=0.943, p'=0.939
exponential fit - not a prediction, a=0.174

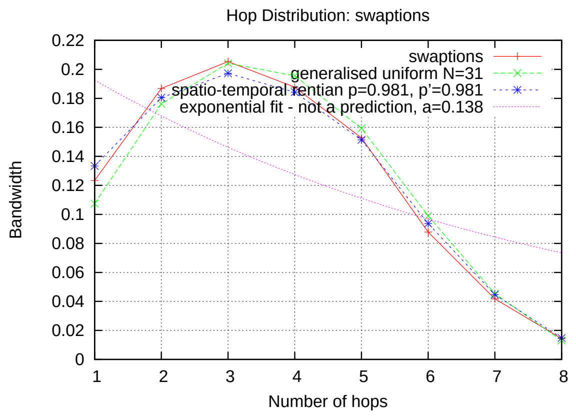

Rentian Scaling: raytrace
1e+06
100000
10000
External BW
1
10
Number of nodes in region
raytrace
uniform-dist N=32.0
spatio-temporal rentian p=0.972, p'=0.971
power-law p=0.797
log-law
modified uniform

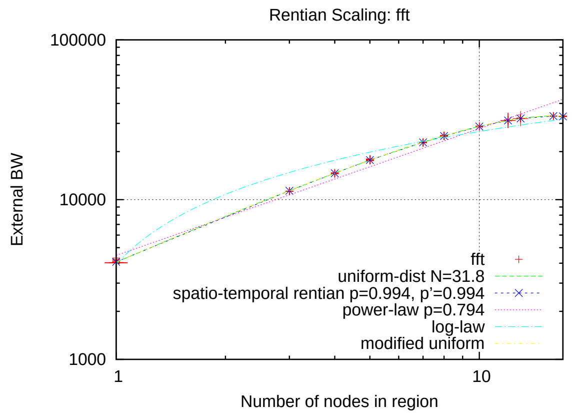

Hop Distribution: raytrace
0.25
0.2
0.15
0.1
0.05
0
Bandwidth
1
2
3
4
5
6
7
8
Number of hops
raytrace
generalised uniform N=32
spatio-temporal rentian p=0.972, p'=0.971
exponential fit - not a prediction, a=0.145

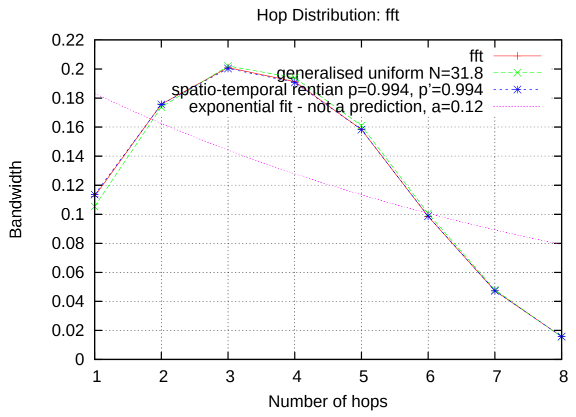

Rentian Scaling: dedup
External BW
Number of nodes in region
1e+07
1e+06
100000
1
10
dedup
uniform-dist N=30.9
spatio-temporal rentian p=0.97, p'=0.969
power-law p=0.784
log-law
modified uniform

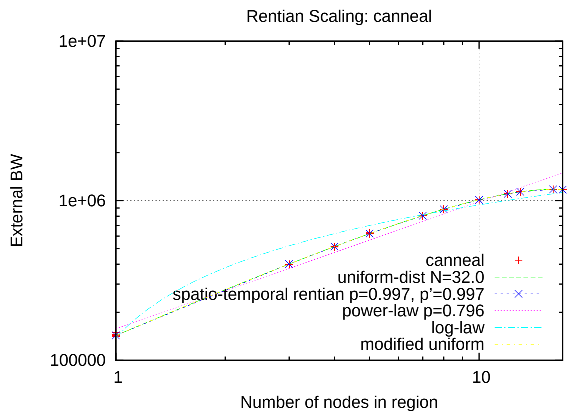

Hop Distribution: dedup
Bandwidth
Number of hops
dedup
generalised uniform N=30.9
spatio-temporal rentian p=0.97, p'=0.969
exponential fit - not a prediction, a=0.152

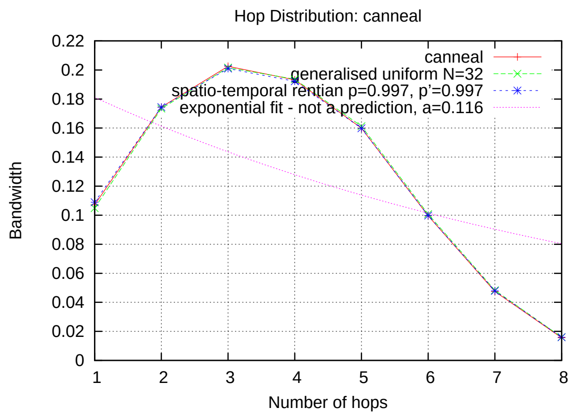

Rentian Scaling: swaptions
External BW
Number of nodes in region
1e+07
1e+06
100000
1
10
swaptions
uniform-dist N=31.0
spatio-temporal rentian p=0.981, p'=0.981
power-law p=0.785
log-law
modified uniform


Hop Distribution: swaptions
Bandwidth
Number of hops
swaptions
generalised uniform N=31
spatio-temporal rentian p=0.981, p'=0.981
exponential fit - not a prediction, a=0.138


Rentian Scaling: fft
External BW
Number of nodes in region
100000
10000
1000
1
10
fft
uniform-dist N=31.8
spatio-temporal rentian p=0.994, p'=0.994
power-law p=0.794
log-law
modified uniform


Hop Distribution: fft
Bandwidth
Number of hops
fft
generalised uniform N=31.8
spatio-temporal rentian p=0.994, p'=0.994
exponential fit - not a prediction, a=0.12


Rentian Scaling: canneal
External BW
Number of nodes in region
1e+07
1e+06
100000
1
10
canneal
uniform-dist N=32.0
spatio-temporal rentian p=0.997, p'=0.997
power-law p=0.796
log-law
modified uniform

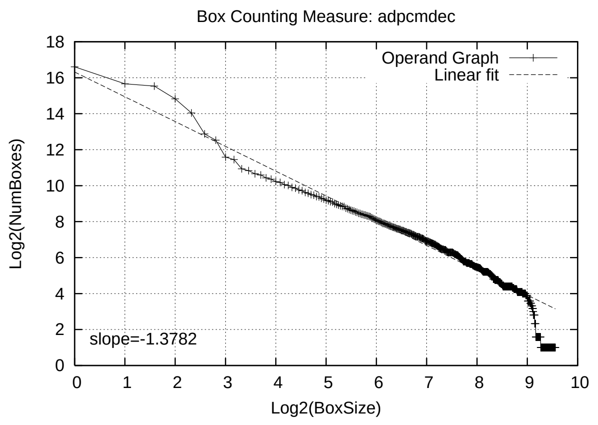

Hop Distribution: canneal
Bandwidth
Number of hops
canneal
generalised uniform N=32
spatio-temporal rentian p=0.997, p'=0.997
exponential fit - not a prediction, a=0.116

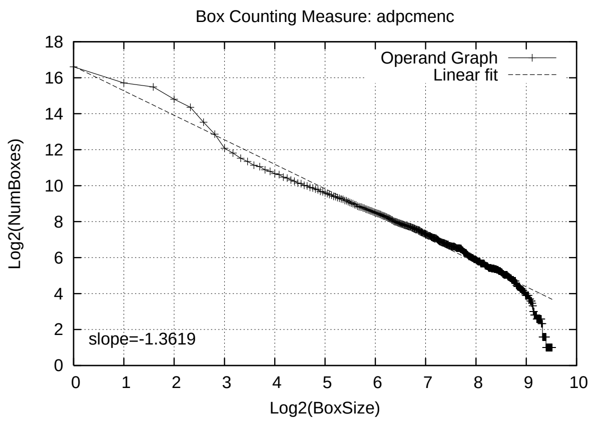
Appendix B

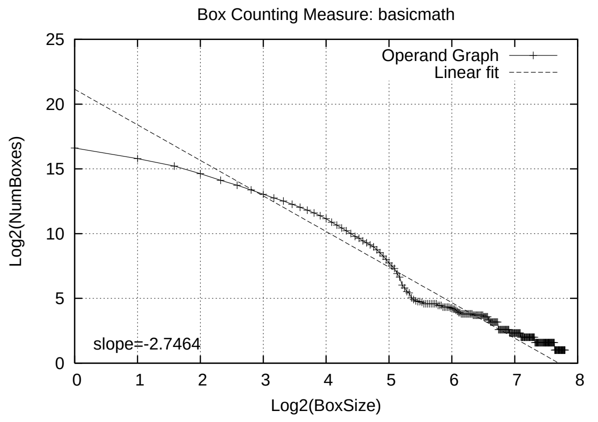
Fractal Dimensionality: MiBench Software

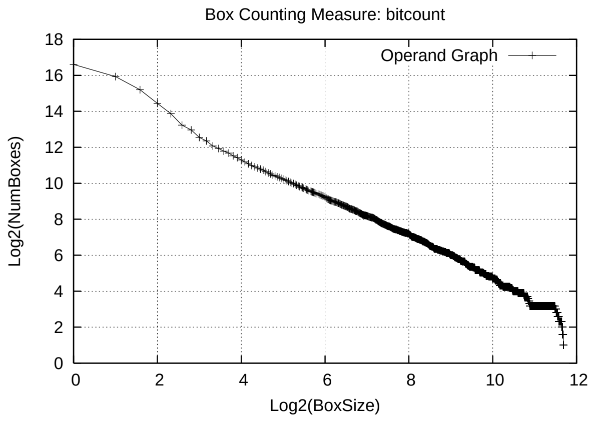
The following plots for MiBench benchmark applications show the box-counting scaling behaviour for the data-dependency graph of 100,000 contiguous instructions. A method adapted from Concas et al [31], itself based on the work of Song et al [95] was used for doing the incremental box-counting.

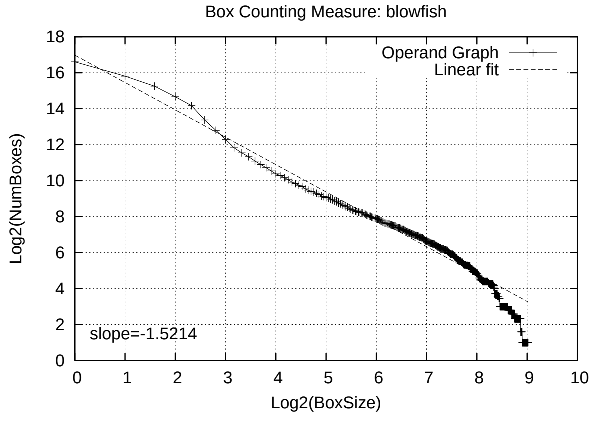
All plots are in log-log form for power-law fitting. Two benchmark graphs (*tiff2bw* and *susan.smoothing*) are not shown as they failed box-counting analysis, due to small-world behaviour. Indeed, there are important differences between box-counting dimension and the Rentian dimensionality, especially when it concerns small-world networks. See the discussion in Chapter 9 for more details.

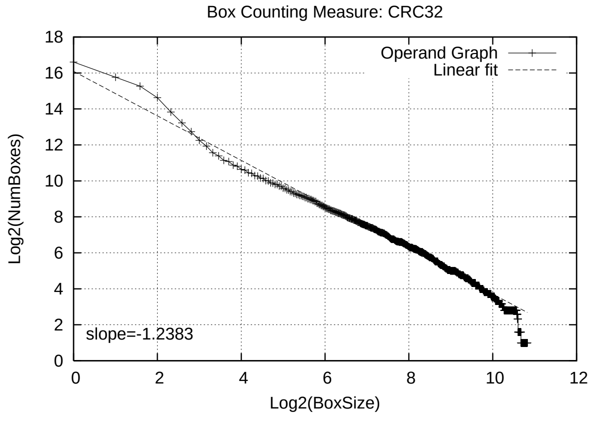

Box Counting Measure: adpcmdec
Operand Graph
Linear fit
slope=-1.3782
Log2(NumBoxes)
Log2(BoxSize)

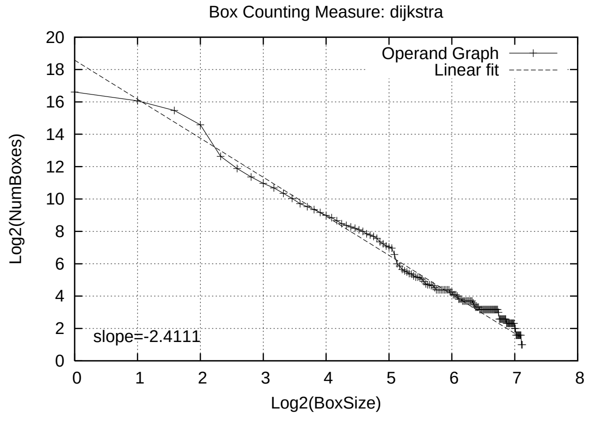

Box Counting Measure: adpcmenc
Operand Graph
Linear fit
slope=-1.3619
Log2(NumBoxes)
Log2(BoxSize)

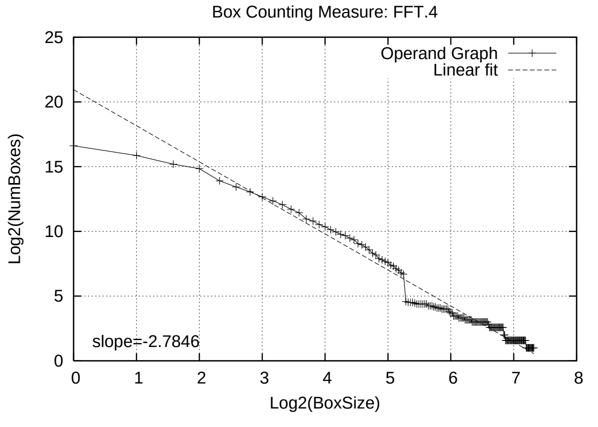

Box Counting Measure: basicmath
Operand Graph
Linear fit
slope=-2.7464
Log2(NumBoxes)
Log2(BoxSize)

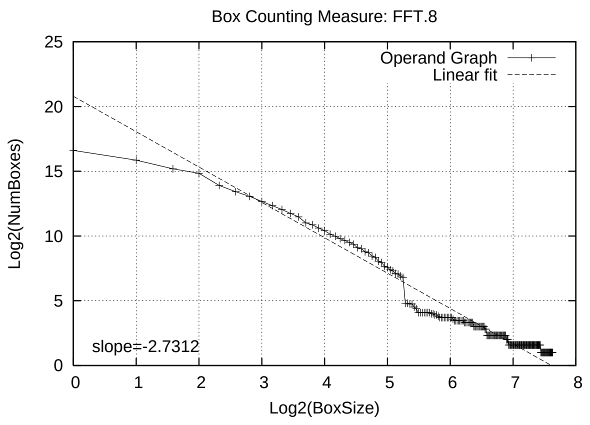

Box Counting Measure: bitcount
Operand Graph
Log2(NumBoxes)
Log2(BoxSize)

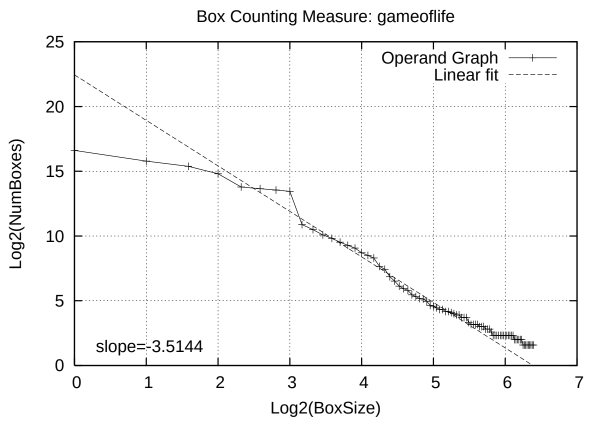

Box Counting Measure: blowfish
Operand Graph
Linear fit
slope=-1.5214
Log2(NumBoxes)
Log2(BoxSize)

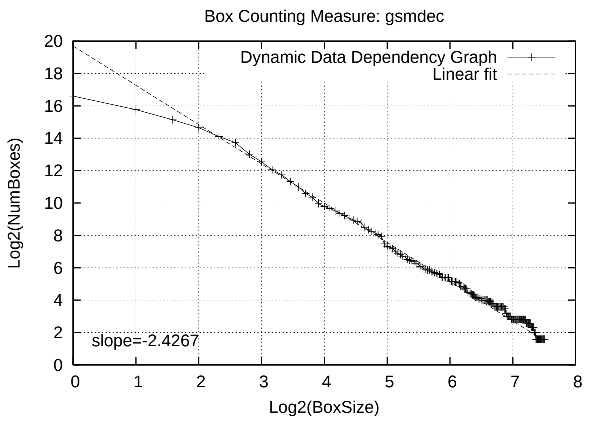

Box Counting Measure: CRC32
Operand Graph
Linear fit
slope=-1.2383
Log2(NumBoxes)
Log2(BoxSize)

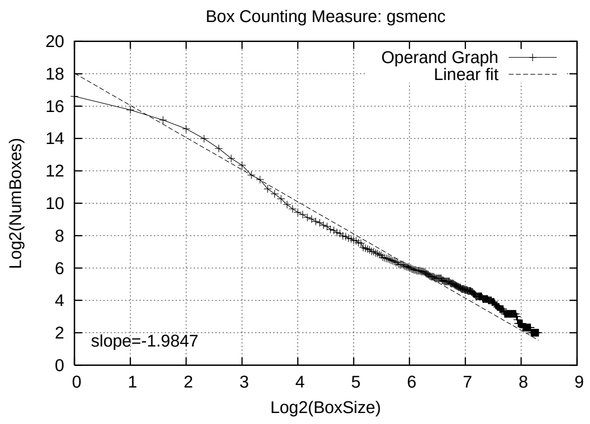

Box Counting Measure: dijkstra
Operand Graph
Linear fit
slope=-2.4111
Log2(NumBoxes)
Log2(BoxSize)

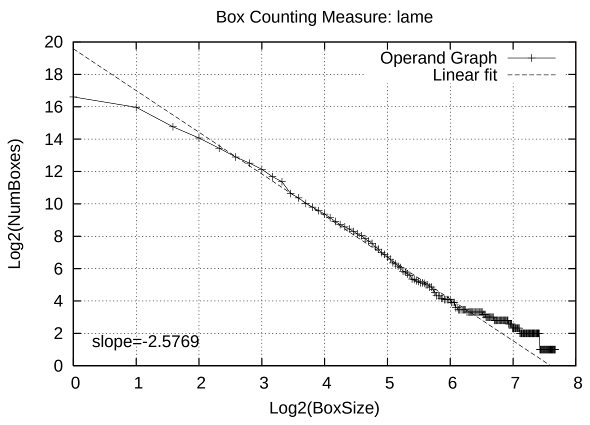

Box Counting Measure: FFT.4
Operand Graph
Linear fit
slope=-2.7846
Log2(NumBoxes)
Log2(BoxSize)

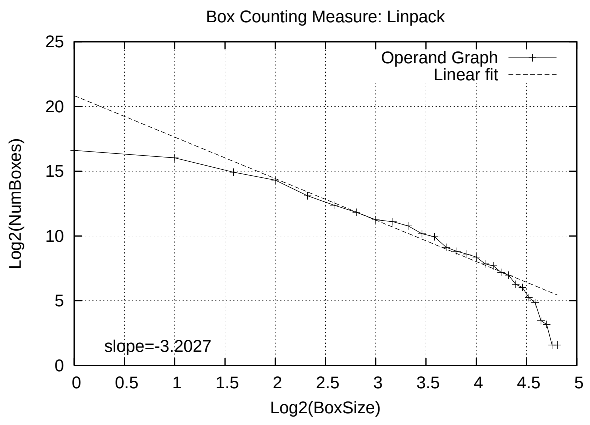

Box Counting Measure: FFT.8
Operand Graph
Linear fit
slope=-2.7312
Log2(NumBoxes)
Log2(BoxSize)

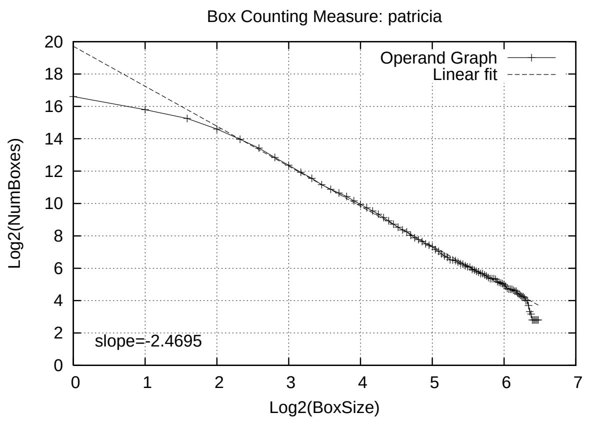

Box Counting Measure: gameoflife
Operand Graph
Linear fit
slope=-3.5144
Log2(NumBoxes)
Log2(BoxSize)

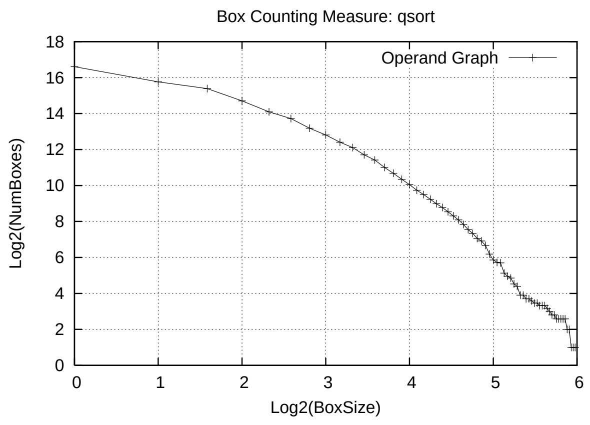

Box Counting Measure: gsmdec
Dynamic Data Dependency Graph
Linear fit
slope=-2.4267
Log2(NumBoxes)
Log2(BoxSize)

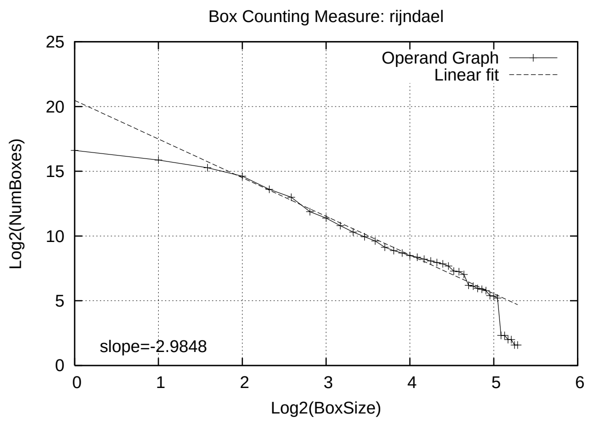

Box Counting Measure: gsmenc
Operand Graph
Linear fit
slope=-1.9847
Log2(NumBoxes)
Log2(BoxSize)

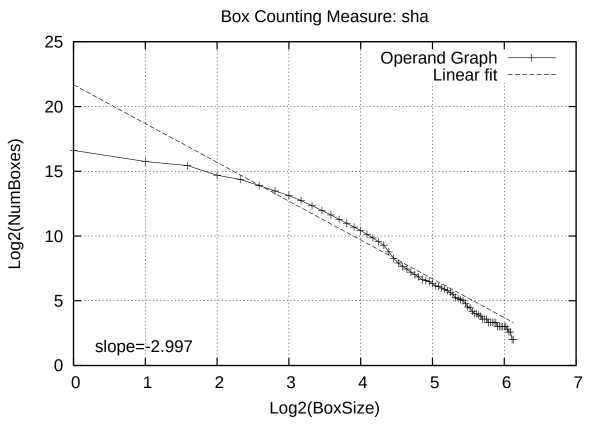

Box Counting Measure: lame
Operand Graph
Linear fit
slope=-2.5769
Log2(NumBoxes)
Log2(BoxSize)

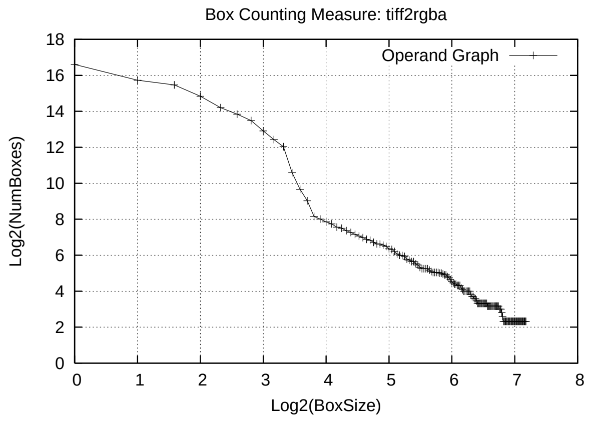

Box Counting Measure: Linpack
Operand Graph
Linear fit
slope=-3.2027
Log2(NumBoxes)
Log2(BoxSize)

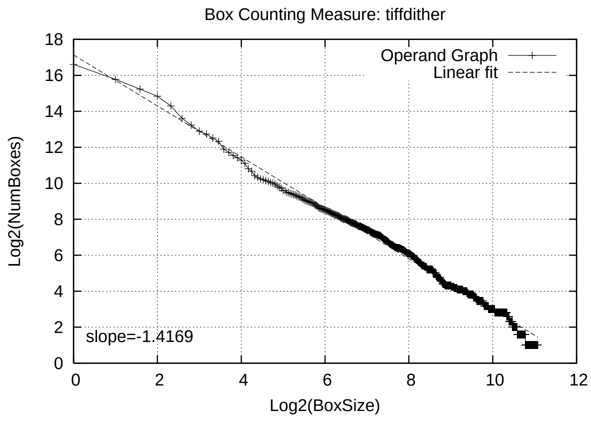

Box Counting Measure: patricia
Operand Graph
Linear fit
slope=-2.4695
Log2(NumBoxes)
Log2(BoxSize)

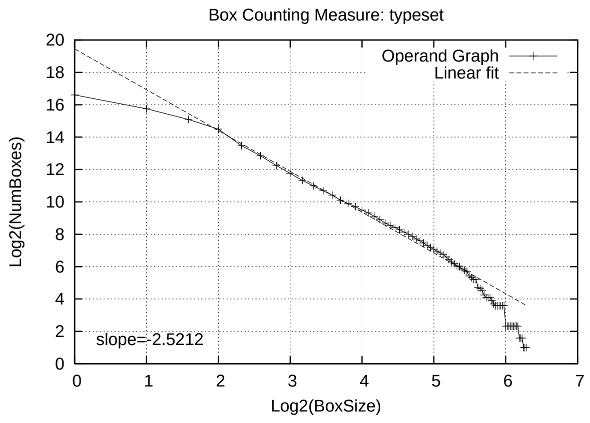

Box Counting Measure: qsort
Operand Graph
Log2(NumBoxes)
Log2(BoxSize)

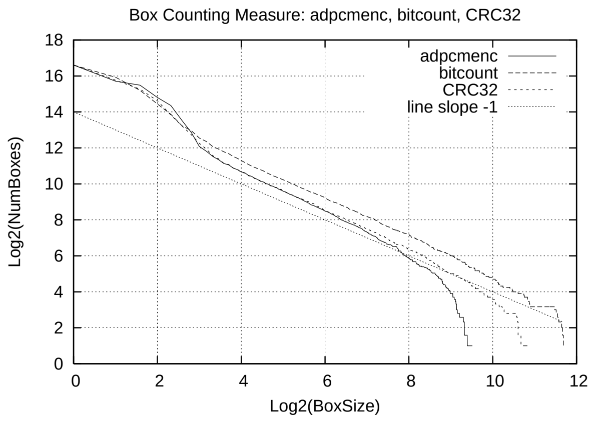

Box Counting Measure: rijndael
Operand Graph
Linear fit
slope=-2.9848
Log2(NumBoxes)
Log2(BoxSize)


Box Counting Measure: sha
Operand Graph
Linear fit
slope=-2.997
Log2(NumBoxes)
Log2(BoxSize)


Box Counting Measure: tiff2rgba
Operand Graph
Log2(NumBoxes)
Log2(BoxSize)


Box Counting Measure: tiffdither
Operand Graph
Linear fit
slope=-1.4169
Log2(NumBoxes)
Log2(BoxSize)


Box Counting Measure: typeset
Operand Graph
Linear fit
slope=-2.5212
Log2(NumBoxes)
Log2(BoxSize)


Box Counting Measure: adpcmenc, bitcount, CRC32
adpcmenc
bitcount
CRC32
line slope -1
Log2(NumBoxes)
Log2(BoxSize)

Appendix C

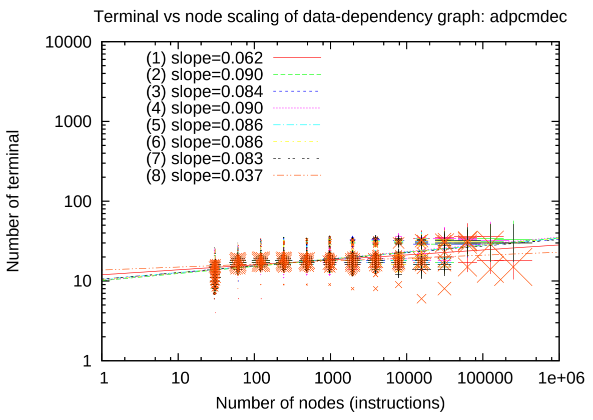
Temporal Scaling: MiBench Software

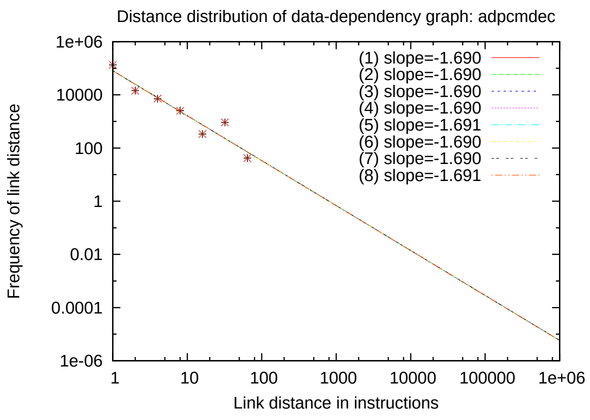
The following plots for MiBench benchmark applications show the scaling behaviour for 2 million instructions of their data-dependency graphs. The meaning of 'placement' here is the actual instruction execution position in time – so the first instruction has position one and the last has instruction position two-million. The communication graph is then segmented approximately equally in time into eight regions starting from segment 1, and ending at segment 8, and these are labelled separately within each plot. The Rentian scaling properties of 'terminals' (cut hyper-edges) to nodes (instructions) is shown on the left. The scaling of the distance distribution of accesses is shown on the right hand side.

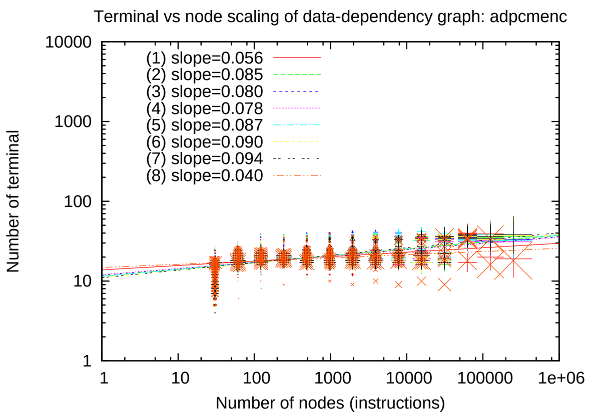
Some correspondence between the Rent's exponent and the distance distribution can be observed. Although the fits are not ideal, with unknown boundaries of Region II effects, most of these have the distance distribution slope about $\sim \hat{p} - 2$, where $\hat{p}$ is the slope of the Rentian fit, except for those with truncated distance distributions (last points ending at an average frequency of about one or more instead of fractional average frequencies of less than one).

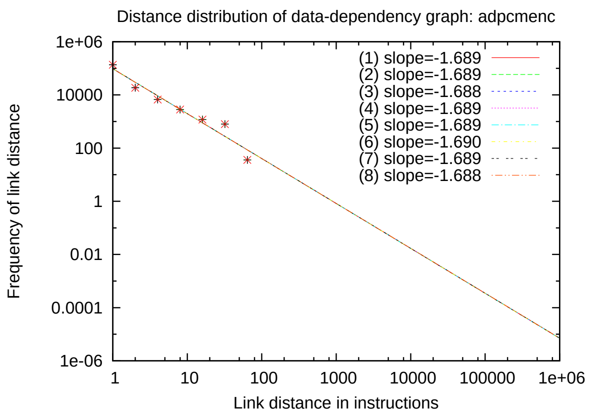
All plots are in log-log form for power-law fitting. Distance distribution frequencies are logarithmically binned. A shallower distance distribution slope implies less locality, as it means there is more communication at further distances. For Rentian distributions, there is a limiting value in the slope of -1 which corresponds to unbounded dimensionality in the communication graph's connectivity. One-dimensional graphs (serial communication) have a limiting-case distance distribution slope of -2, or steeper. For the Rentian plots on the left, the size of the cross corresponds to the degeneracy (number of identical samples) at that sample point.

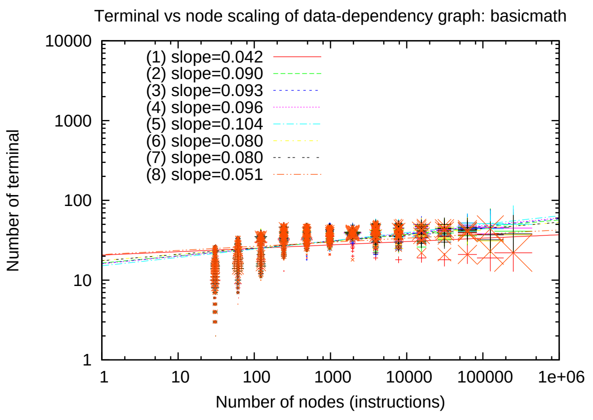
We should note that all fits are over the entire data-range, and therefore may include Region II effects that may alter the values of the slopes.

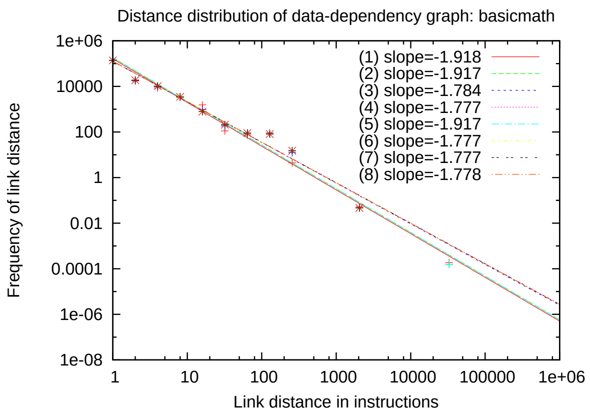

Terminal vs node scaling of data-dependency graph: adpcmdec
(1) slope=0.062
(2) slope=0.090
(3) slope=0.084
(4) slope=0.090
(5) slope=0.086
(6) slope=0.086
(7) slope=0.083
(8) slope=0.037
Number of terminal
Number of nodes (instructions)

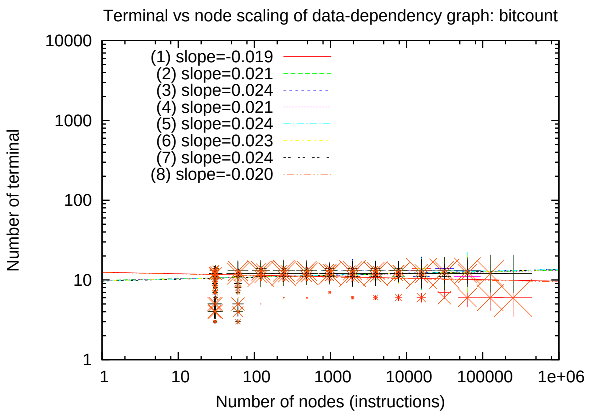

Distance distribution of data-dependency graph: adpcmdec
(1) slope=-1.690
(2) slope=-1.690
(3) slope=-1.690
(4) slope=-1.690
(5) slope=-1.691
(6) slope=-1.690
(7) slope=-1.690
(8) slope=-1.691
Frequency of link distance
Link distance in instructions

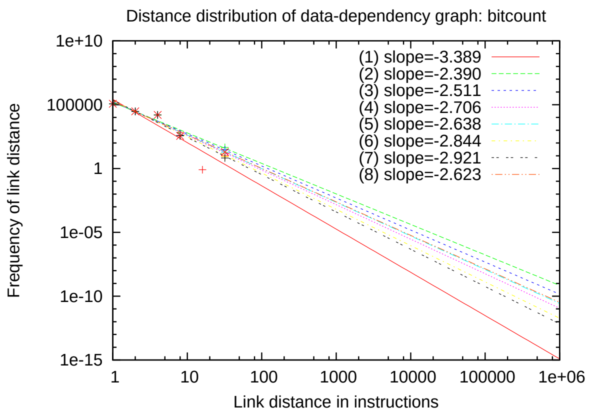

Terminal vs node scaling of data-dependency graph: adpcmenc
(1) slope=0.056
(2) slope=0.085
(3) slope=0.080
(4) slope=0.078
(5) slope=0.087
(6) slope=0.090
(7) slope=0.094
(8) slope=0.040
Number of terminal
Number of nodes (instructions)

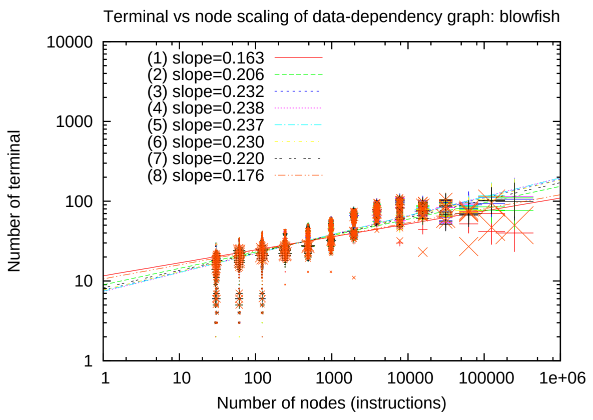

Distance distribution of data-dependency graph: adpcmenc
(1) slope=-1.689
(2) slope=-1.689
(3) slope=-1.688
(4) slope=-1.689
(5) slope=-1.689
(6) slope=-1.690
(7) slope=-1.689
(8) slope=-1.688
Frequency of link distance
Link distance in instructions

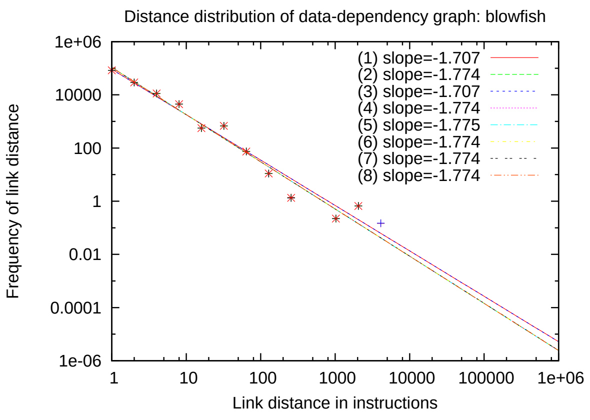

Terminal vs node scaling of data-dependency graph: basicmath
(1) slope=0.042
(2) slope=0.090
(3) slope=0.093
(4) slope=0.096
(5) slope=0.104
(6) slope=0.080
(7) slope=0.080
(8) slope=0.051
Number of terminal
Number of nodes (instructions)

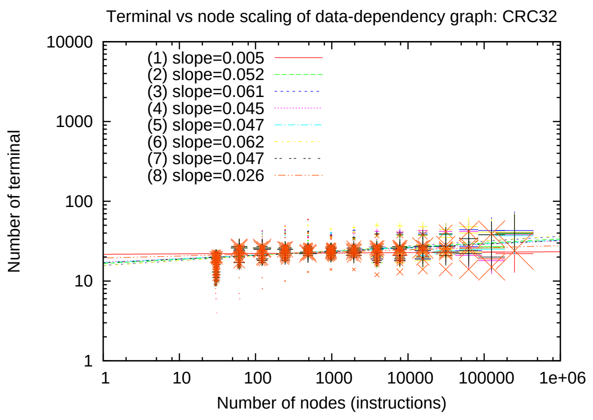

Distance distribution of data-dependency graph: basicmath
(1) slope=-1.918
(2) slope=-1.917
(3) slope=-1.784
(4) slope=-1.777
(5) slope=-1.917
(6) slope=-1.777
(7) slope=-1.777
(8) slope=-1.778
Frequency of link distance
Link distance in instructions

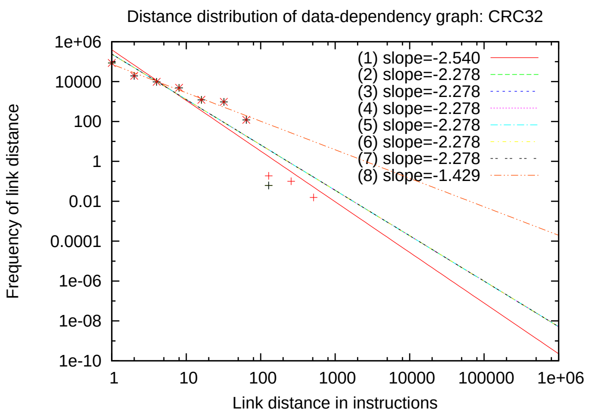

Terminal vs node scaling of data-dependency graph: bitcount
(1) slope=-0.019
(2) slope=0.021
(3) slope=0.024
(4) slope=0.021
(5) slope=0.024
(6) slope=0.023
(7) slope=0.024
(8) slope=-0.020
Number of terminal
Number of nodes (instructions)

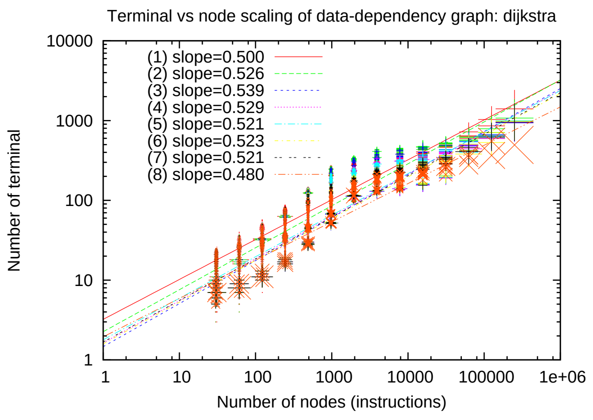

Distance distribution of data-dependency graph: bitcount
(1) slope=-3.389
(2) slope=-2.390
(3) slope=-2.511
(4) slope=-2.706
(5) slope=-2.638
(6) slope=-2.844
(7) slope=-2.921
(8) slope=-2.623
Frequency of link distance
Link distance in instructions

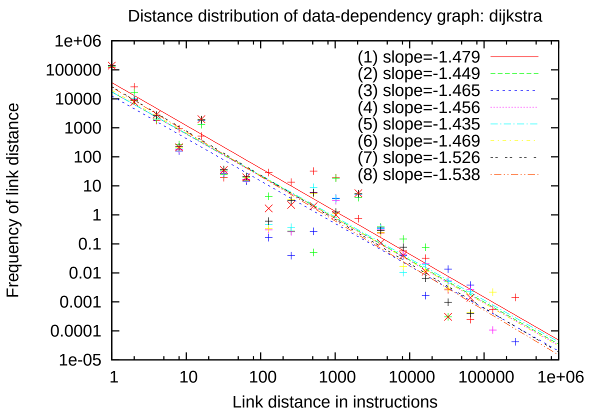

Terminal vs node scaling of data-dependency graph: blowfish
(1) slope=0.163
(2) slope=0.206
(3) slope=0.232
(4) slope=0.238
(5) slope=0.237
(6) slope=0.230
(7) slope=0.220
(8) slope=0.176
Number of terminal
Number of nodes (instructions)

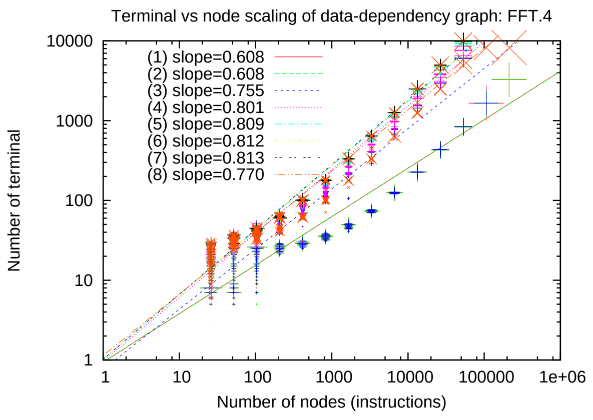

Distance distribution of data-dependency graph: blowfish
(1) slope=-1.707
(2) slope=-1.774
(3) slope=-1.707
(4) slope=-1.774
(5) slope=-1.775
(6) slope=-1.774
(7) slope=-1.774
(8) slope=-1.774
Frequency of link distance
Link distance in instructions

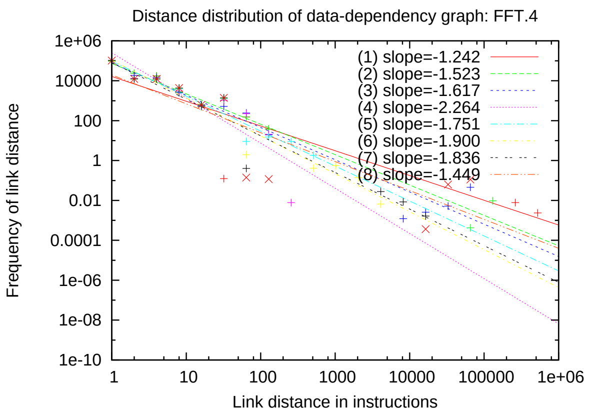

Terminal vs node scaling of data-dependency graph: CRC32
(1) slope=0.005
(2) slope=0.052
(3) slope=0.061
(4) slope=0.045
(5) slope=0.047
(6) slope=0.062
(7) slope=0.047
(8) slope=0.026
Number of terminal
Number of nodes (instructions)

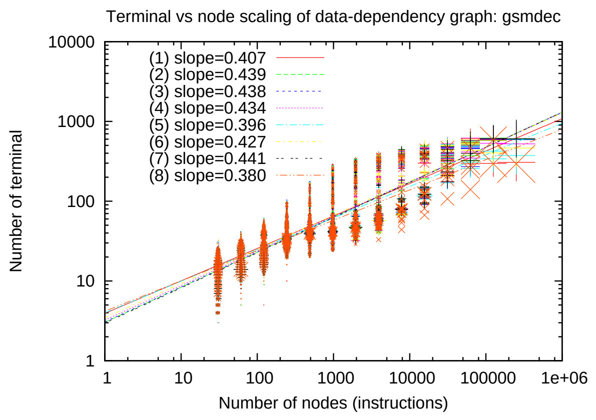

Distance distribution of data-dependency graph: CRC32
(1) slope=-2.540
(2) slope=-2.278
(3) slope=-2.278
(4) slope=-2.278
(5) slope=-2.278
(6) slope=-2.278
(7) slope=-2.278
(8) slope=-1.429
Frequency of link distance
Link distance in instructions

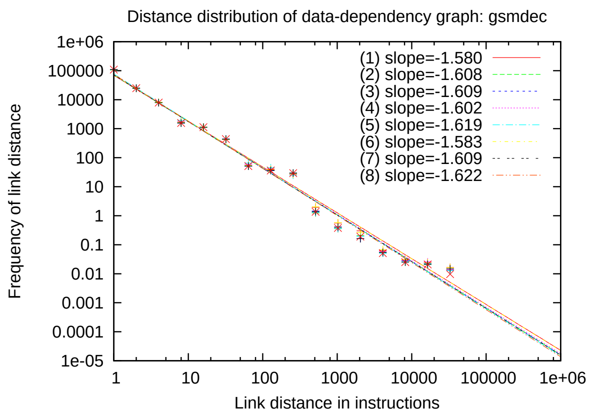

Terminal vs node scaling of data-dependency graph: dijkstra
(1) slope=0.500
(2) slope=0.526
(3) slope=0.539
(4) slope=0.529
(5) slope=0.521
(6) slope=0.523
(7) slope=0.521
(8) slope=0.480
Number of terminal
Number of nodes (instructions)

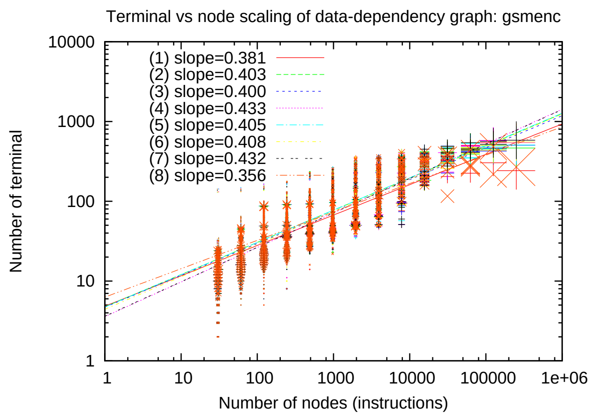

Distance distribution of data-dependency graph: dijkstra
(1) slope=-1.479
(2) slope=-1.449
(3) slope=-1.465
(4) slope=-1.456
(5) slope=-1.435
(6) slope=-1.469
(7) slope=-1.526
(8) slope=-1.538
Frequency of link distance
Link distance in instructions

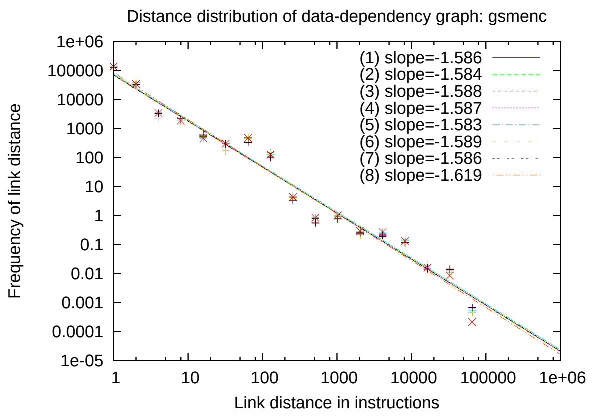

Terminal vs node scaling of data-dependency graph: FFT.4
(1) slope=0.608
(2) slope=0.608
(3) slope=0.755
(4) slope=0.801
(5) slope=0.809
(6) slope=0.812
(7) slope=0.813
(8) slope=0.770
Number of terminal
Number of nodes (instructions)

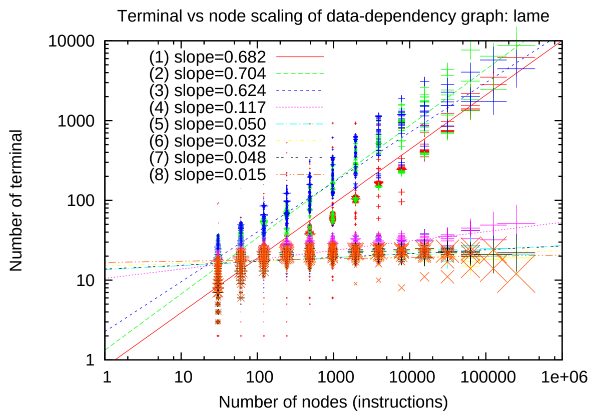

Distance distribution of data-dependency graph: FFT.4
(1) slope=-1.242
(2) slope=-1.523
(3) slope=-1.617
(4) slope=-2.264
(5) slope=-1.751
(6) slope=-1.900
(7) slope=-1.836
(8) slope=-1.449
Frequency of link distance
Link distance in instructions

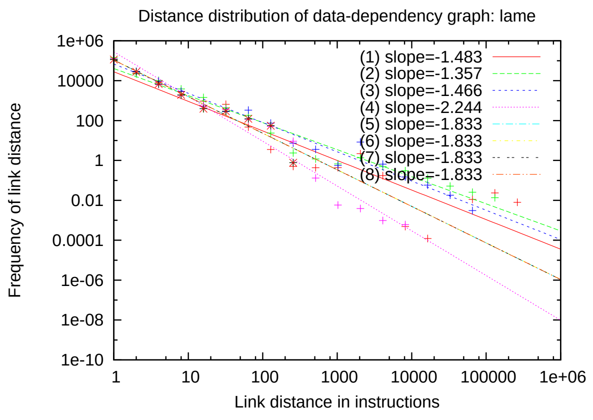

Terminal vs node scaling of data-dependency graph: gsmdec
(1) slope=0.407
(2) slope=0.439
(3) slope=0.438
(4) slope=0.434
(5) slope=0.396
(6) slope=0.427
(7) slope=0.441
(8) slope=0.380
Number of terminal
Number of nodes (instructions)

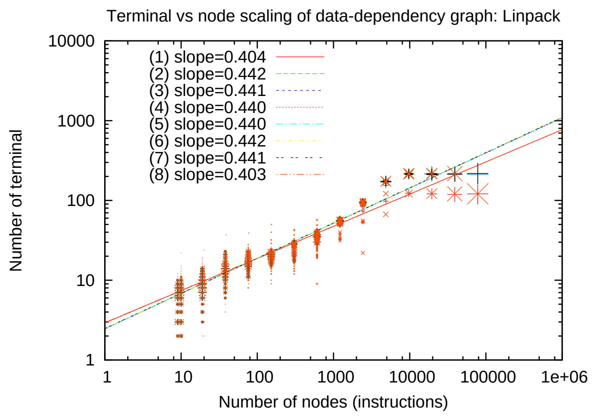

Distance distribution of data-dependency graph: gsmdec
(1) slope=-1.580
(2) slope=-1.608
(3) slope=-1.609
(4) slope=-1.602
(5) slope=-1.619
(6) slope=-1.583
(7) slope=-1.609
(8) slope=-1.622
Frequency of link distance
Link distance in instructions

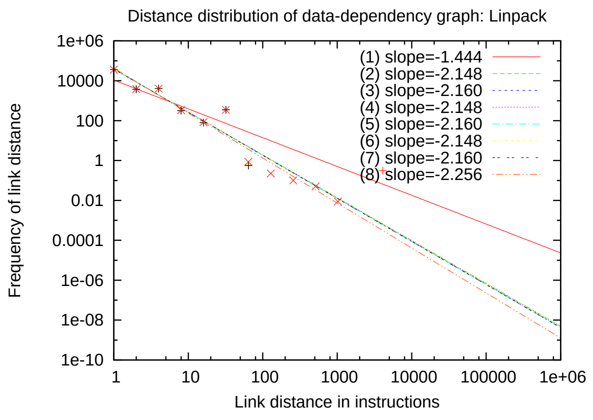

Terminal vs node scaling of data-dependency graph: gsmenc
(1) slope=0.381
(2) slope=0.403
(3) slope=0.400
(4) slope=0.433
(5) slope=0.405
(6) slope=0.408
(7) slope=0.432
(8) slope=0.356
Number of terminal
Number of nodes (instructions)

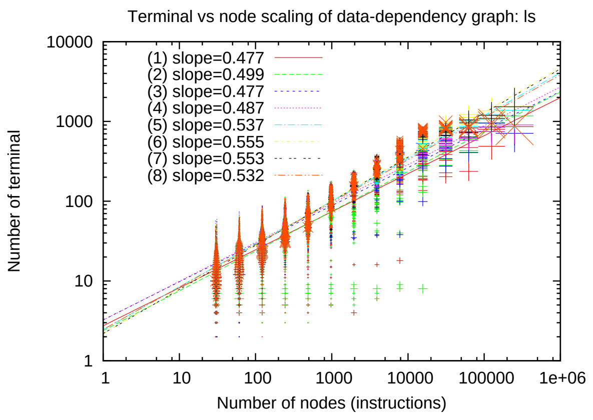

Distance distribution of data-dependency graph: gsmenc
(1) slope=-1.586
(2) slope=-1.584
(3) slope=-1.588
(4) slope=-1.587
(5) slope=-1.583
(6) slope=-1.589
(7) slope=-1.586
(8) slope=-1.619
Frequency of link distance
Link distance in instructions

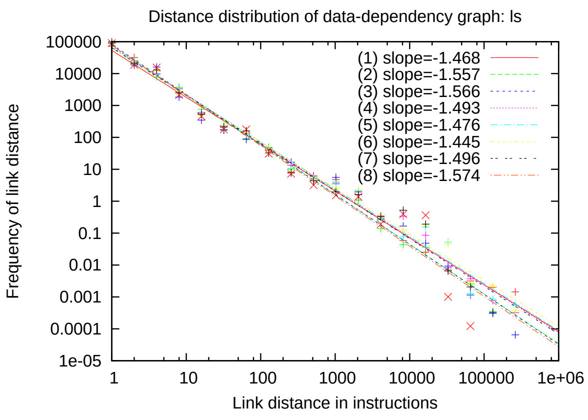

Terminal vs node scaling of data-dependency graph: lame
(1) slope=0.682
(2) slope=0.704
(3) slope=0.624
(4) slope=0.117
(5) slope=0.050
(6) slope=0.032
(7) slope=0.048
(8) slope=0.015
Number of terminal
Number of nodes (instructions)

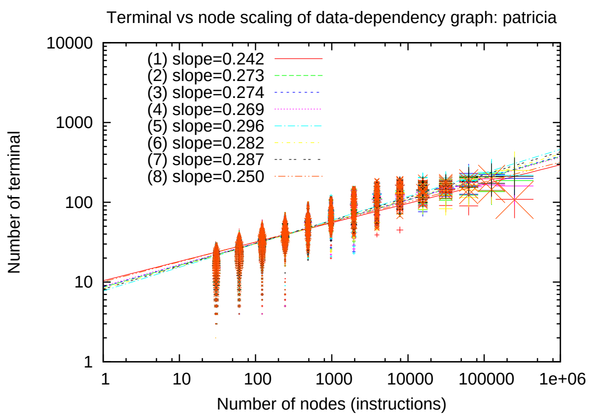

Distance distribution of data-dependency graph: lame
(1) slope=-1.483
(2) slope=-1.357
(3) slope=-1.466
(4) slope=-2.244
(5) slope=-1.833
(6) slope=-1.833
(7) slope=-1.833
(8) slope=-1.833
Frequency of link distance
Link distance in instructions

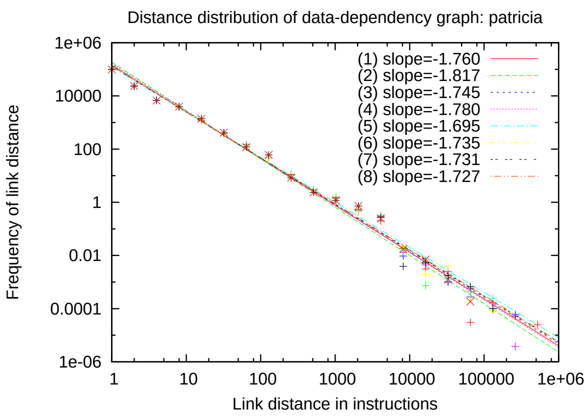

Terminal vs node scaling of data-dependency graph: Linpack
(1) slope=0.404
(2) slope=0.442
(3) slope=0.441
(4) slope=0.440
(5) slope=0.440
(6) slope=0.442
(7) slope=0.441
(8) slope=0.403
Number of terminal
Number of nodes (instructions)

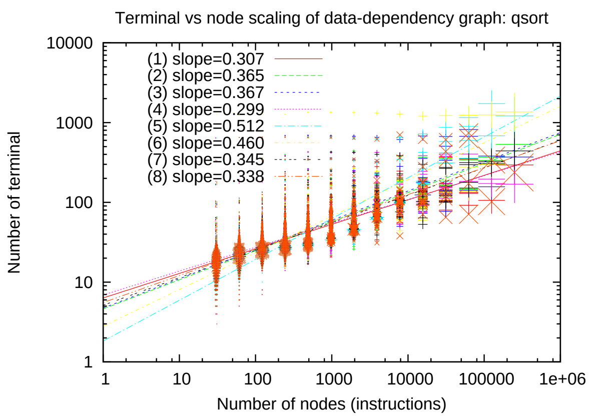

Distance distribution of data-dependency graph: Linpack
(1) slope=-1.444
(2) slope=-2.148
(3) slope=-2.160
(4) slope=-2.148
(5) slope=-2.160
(6) slope=-2.148
(7) slope=-2.160
(8) slope=-2.256
Frequency of link distance
Link distance in instructions

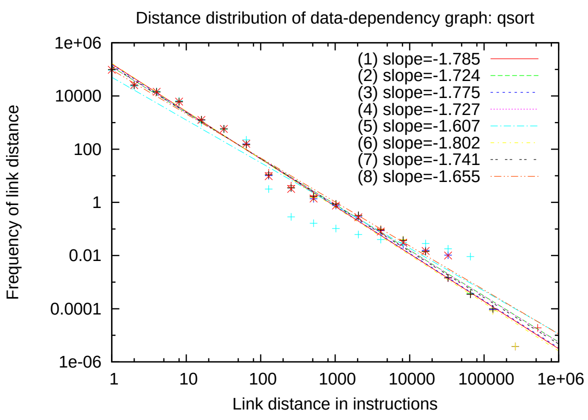

Terminal vs node scaling of data-dependency graph: ls
Number of terminal
Number of nodes (instructions)
(1) slope=0.477
(2) slope=0.499
(3) slope=0.477
(4) slope=0.487
(5) slope=0.537
(6) slope=0.555
(7) slope=0.553
(8) slope=0.532
1
10
100
1000
10000
100000
1e+06

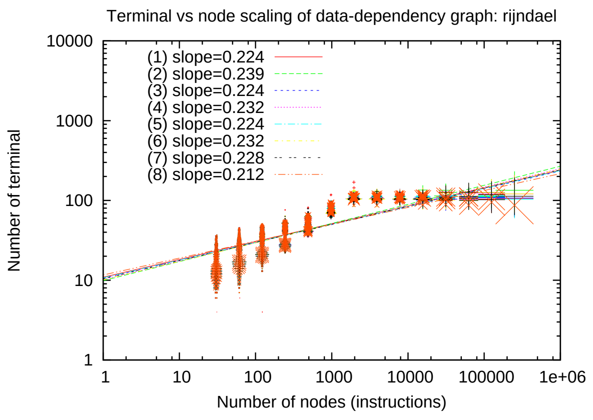

Distance distribution of data-dependency graph: ls
Frequency of link distance
Link distance in instructions
(1) slope=-1.468
(2) slope=-1.557
(3) slope=-1.566
(4) slope=-1.493
(5) slope=-1.476
(6) slope=-1.445
(7) slope=-1.496
(8) slope=-1.574
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05
1e+06

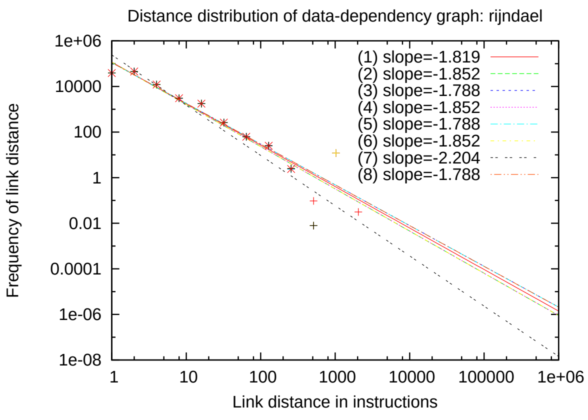

Terminal vs node scaling of data-dependency graph: patricia
Number of terminal
Number of nodes (instructions)
(1) slope=0.242
(2) slope=0.273
(3) slope=0.274
(4) slope=0.269
(5) slope=0.296
(6) slope=0.282
(7) slope=0.287
(8) slope=0.250
1
10
100
1000
10000
100000
1e+06

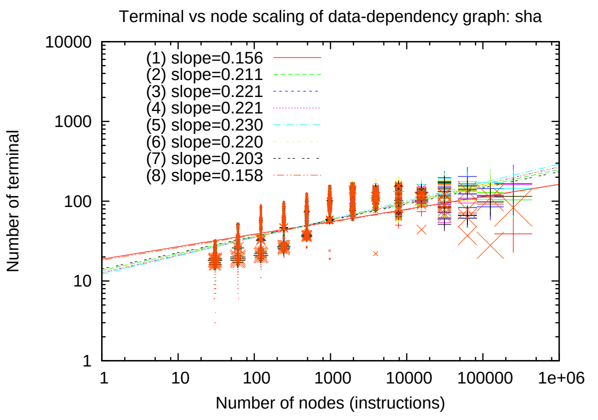

Distance distribution of data-dependency graph: patricia
Frequency of link distance
Link distance in instructions
(1) slope=-1.760
(2) slope=-1.817
(3) slope=-1.745
(4) slope=-1.780
(5) slope=-1.695
(6) slope=-1.735
(7) slope=-1.731
(8) slope=-1.727
1e+06
10000
100
1
0.01
0.0001
1e-06
10
1000
100000

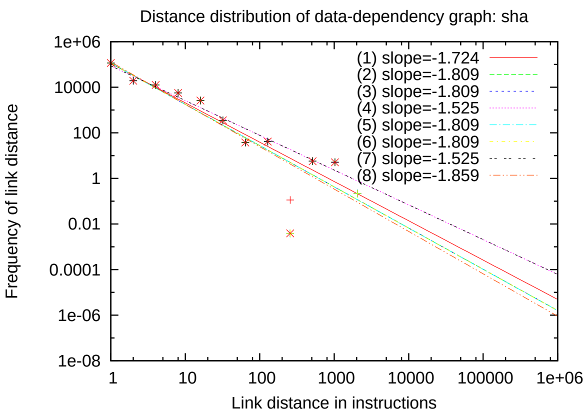

Terminal vs node scaling of data-dependency graph: qsort
Number of terminal
Number of nodes (instructions)
(1) slope=0.307
(2) slope=0.365
(3) slope=0.367
(4) slope=0.299
(5) slope=0.512
(6) slope=0.460
(7) slope=0.345
(8) slope=0.338
1
10
100
1000
10000
100000
1e+06

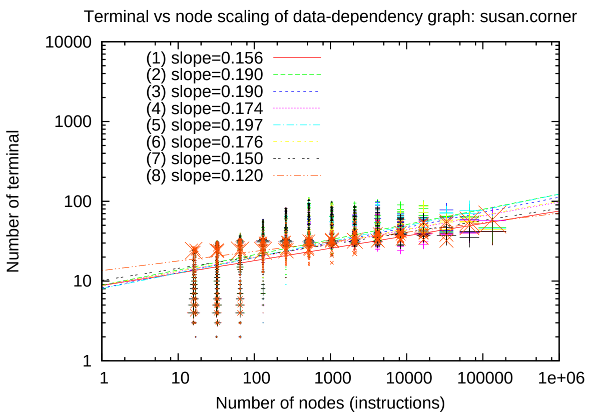

Distance distribution of data-dependency graph: qsort
Frequency of link distance
Link distance in instructions
(1) slope=-1.785
(2) slope=-1.724
(3) slope=-1.775
(4) slope=-1.727
(5) slope=-1.607
(6) slope=-1.802
(7) slope=-1.741
(8) slope=-1.655
1e+06
10000
100
1
0.01
0.0001
1e-06
10
1000
100000

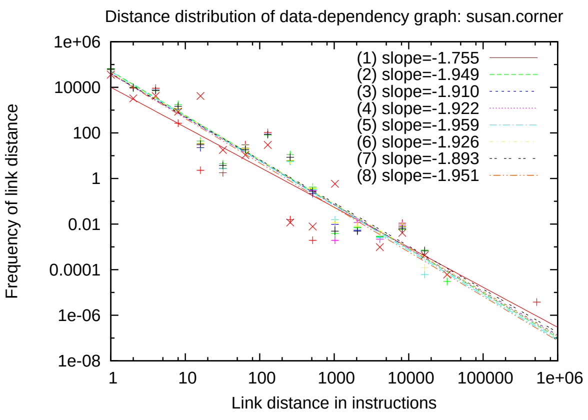

Terminal vs node scaling of data-dependency graph: rijndael
Number of terminal
Number of nodes (instructions)
(1) slope=0.224
(2) slope=0.239
(3) slope=0.224
(4) slope=0.232
(5) slope=0.224
(6) slope=0.232
(7) slope=0.228
(8) slope=0.212
1
10
100
1000
10000
100000
1e+06

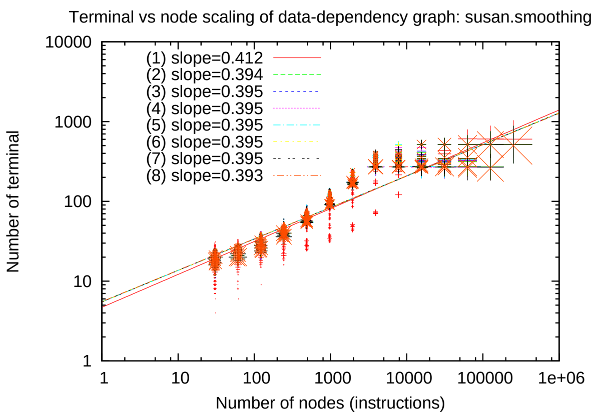

Distance distribution of data-dependency graph: rijndael
Frequency of link distance
Link distance in instructions
(1) slope=-1.819
(2) slope=-1.852
(3) slope=-1.788
(4) slope=-1.852
(5) slope=-1.788
(6) slope=-1.852
(7) slope=-2.204
(8) slope=-1.788
1e+06
10000
100
1
0.01
0.0001
1e-06
1e-08
10
1000
100000

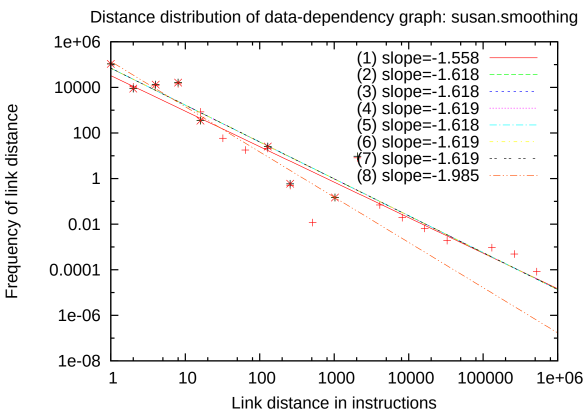

Terminal vs node scaling of data-dependency graph: sha
(1) slope=0.156
(2) slope=0.211
(3) slope=0.221
(4) slope=0.221
(5) slope=0.230
(6) slope=0.220
(7) slope=0.203
(8) slope=0.158
Number of terminal
Number of nodes (instructions)
1
10
100
1000
10000
100000
1e+06

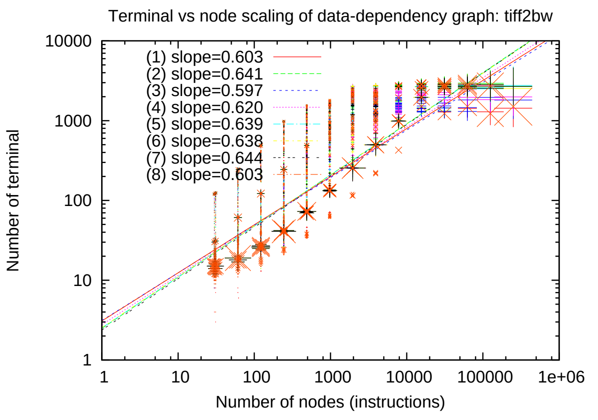

Distance distribution of data-dependency graph: sha
(1) slope=-1.724
(2) slope=-1.809
(3) slope=-1.809
(4) slope=-1.525
(5) slope=-1.809
(6) slope=-1.809
(7) slope=-1.525
(8) slope=-1.859
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06
1e-08

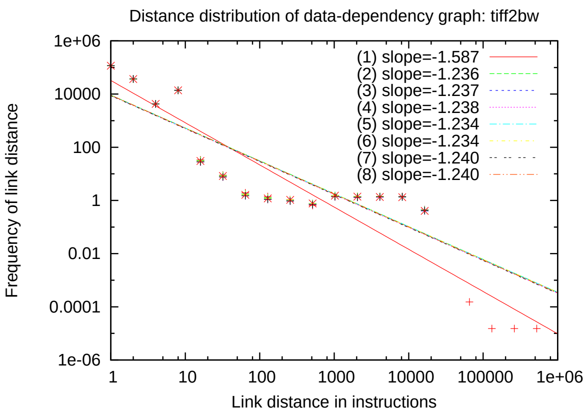

Terminal vs node scaling of data-dependency graph: susan.corner
(1) slope=0.156
(2) slope=0.190
(3) slope=0.190
(4) slope=0.174
(5) slope=0.197
(6) slope=0.176
(7) slope=0.150
(8) slope=0.120
Number of terminal
Number of nodes (instructions)

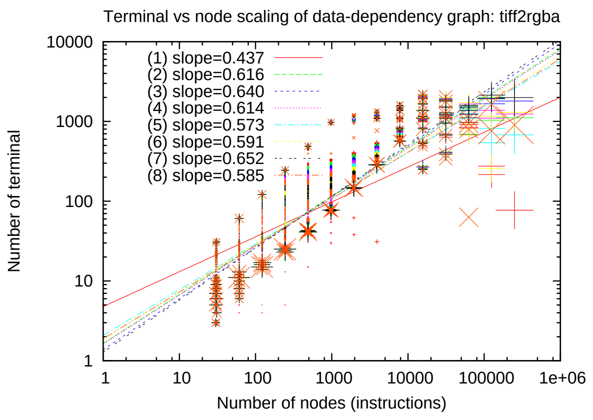

Distance distribution of data-dependency graph: susan.corner
(1) slope=-1.755
(2) slope=-1.949
(3) slope=-1.910
(4) slope=-1.922
(5) slope=-1.959
(6) slope=-1.926
(7) slope=-1.893
(8) slope=-1.951
Frequency of link distance
Link distance in instructions

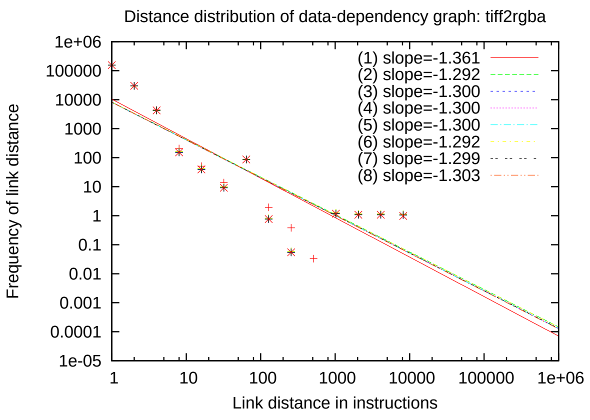

Terminal vs node scaling of data-dependency graph: susan.smoothing
(1) slope=0.412
(2) slope=0.394
(3) slope=0.395
(4) slope=0.395
(5) slope=0.395
(6) slope=0.395
(7) slope=0.395
(8) slope=0.393
Number of terminal
Number of nodes (instructions)

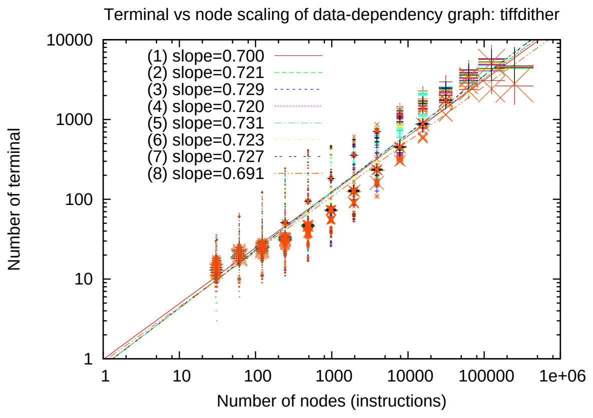

Distance distribution of data-dependency graph: susan.smoothing
(1) slope=-1.558
(2) slope=-1.618
(3) slope=-1.618
(4) slope=-1.619
(5) slope=-1.618
(6) slope=-1.619
(7) slope=-1.619
(8) slope=-1.985
Frequency of link distance
Link distance in instructions

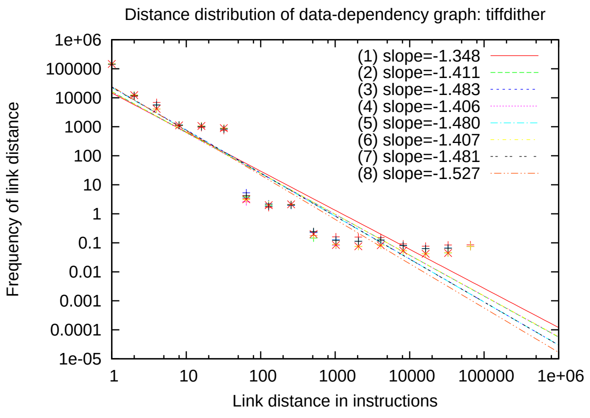

Terminal vs node scaling of data-dependency graph: tiff2bw
(1) slope=0.603
(2) slope=0.641
(3) slope=0.597
(4) slope=0.620
(5) slope=0.639
(6) slope=0.638
(7) slope=0.644
(8) slope=0.603
Number of terminal
Number of nodes (instructions)

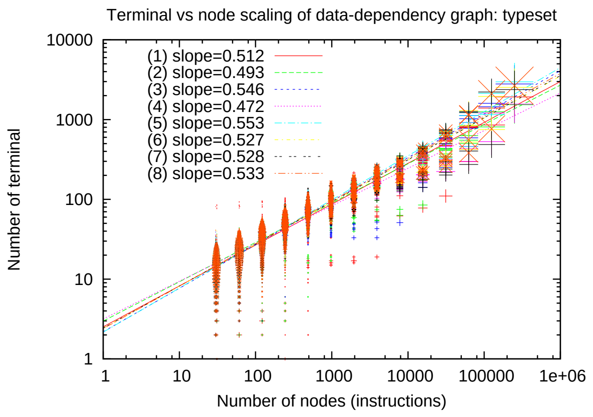

Distance distribution of data-dependency graph: tiff2bw
(1) slope=-1.587
(2) slope=-1.236
(3) slope=-1.237
(4) slope=-1.238
(5) slope=-1.234
(6) slope=-1.234
(7) slope=-1.240
(8) slope=-1.240
Frequency of link distance
Link distance in instructions

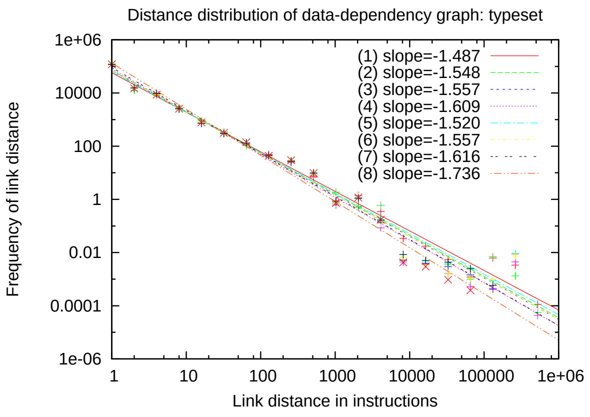

Terminal vs node scaling of data-dependency graph: tiff2rgba
(1) slope=0.437
(2) slope=0.616
(3) slope=0.640
(4) slope=0.614
(5) slope=0.573
(6) slope=0.591
(7) slope=0.652
(8) slope=0.585
Number of terminal
Number of nodes (instructions)
1
10
100
1000
10000
100000
1e+06


Distance distribution of data-dependency graph: tiff2rgba
(1) slope=-1.361
(2) slope=-1.292
(3) slope=-1.300
(4) slope=-1.300
(5) slope=-1.300
(6) slope=-1.292
(7) slope=-1.299
(8) slope=-1.303
Frequency of link distance
Link distance in instructions
1e+06
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05


Terminal vs node scaling of data-dependency graph: tiffdither
(1) slope=0.700
(2) slope=0.721
(3) slope=0.729
(4) slope=0.720
(5) slope=0.731
(6) slope=0.723
(7) slope=0.727
(8) slope=0.691
Number of terminal
Number of nodes (instructions)


Distance distribution of data-dependency graph: tiffdither
(1) slope=-1.348
(2) slope=-1.411
(3) slope=-1.483
(4) slope=-1.406
(5) slope=-1.480
(6) slope=-1.407
(7) slope=-1.481
(8) slope=-1.527
Frequency of link distance
Link distance in instructions


Terminal vs node scaling of data-dependency graph: typeset
(1) slope=0.512
(2) slope=0.493
(3) slope=0.546
(4) slope=0.472
(5) slope=0.553
(6) slope=0.527
(7) slope=0.528
(8) slope=0.533
Number of terminal
Number of nodes (instructions)


Distance distribution of data-dependency graph: typeset
(1) slope=-1.487
(2) slope=-1.548
(3) slope=-1.557
(4) slope=-1.609
(5) slope=-1.520
(6) slope=-1.557
(7) slope=-1.616
(8) slope=-1.736
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06

Appendix D

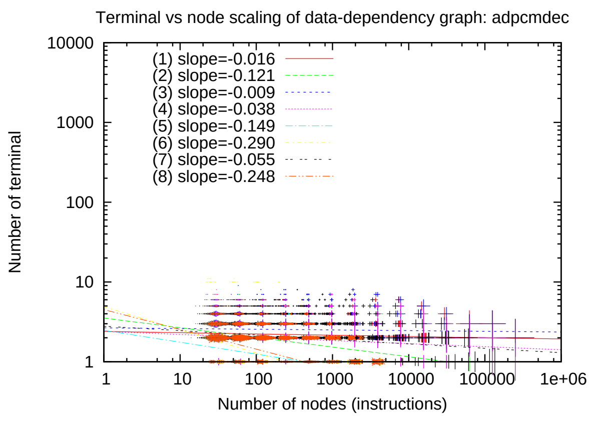
Partitioned Scaling: MiBench Software

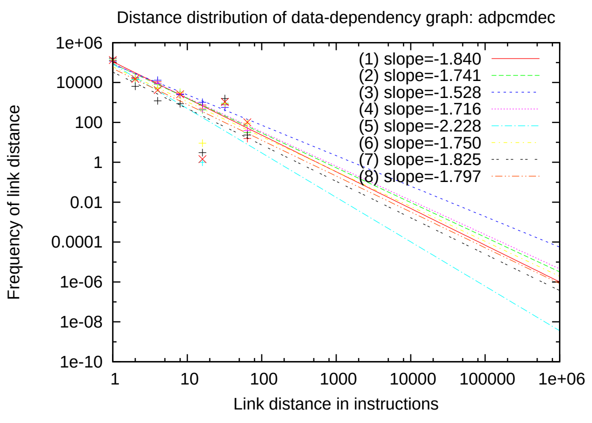
The following plots for MiBench benchmark applications show the scaling behaviour for 2 million instructions of their data-dependency graphs. The 'partitioning' here uses hMetis [62] which tries to find a min-cut bisection of the communication graph using hyper-edges. This is done recursively for sixteen levels. The top three levels of bisection form eight regions which are labelled segment 1 to segment 8, which although tend to also correspond to eight near-contiguous temporal segments, do not necessarily do so. These labels are shown within each plot. The Rentian scaling properties of 'terminals' (cut hyper-edges) to nodes (instructions) is shown on the left. The scaling of the distance distribution of accesses is shown on the right hand side.

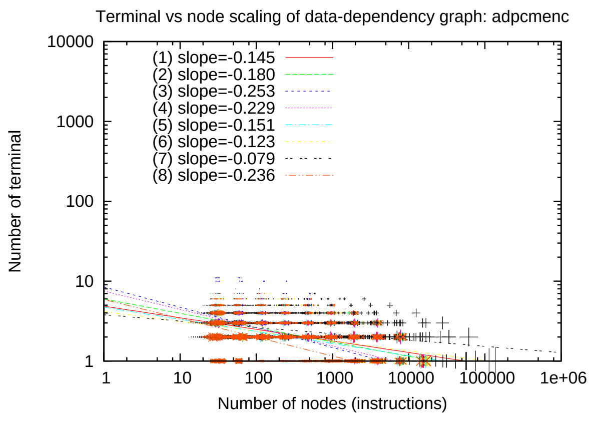
The distance distribution here, as in the previous chapter, uses the actual instruction time distances but using the partitioning found by hMetis. We should perhaps note that the min-cut problem is actually NP-complete, and hMetis is using heuristic approaches and annealing to achieve its result. As such hMetis is not guaranteed to find a minimal partitioning. One would hope that it could do better than the existing embedding of instructions in time from the previous chapter, but this is not necessarily the case.

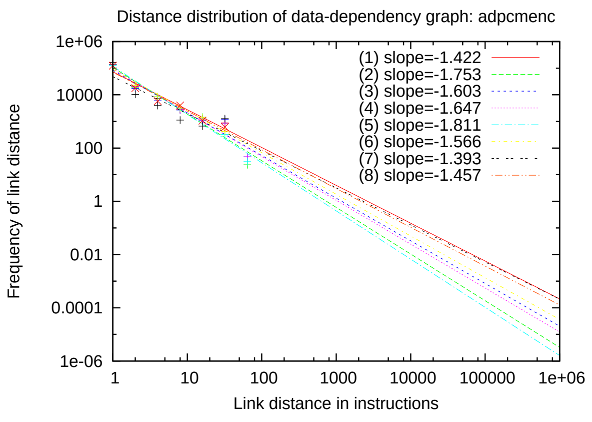
All plots are in log-log form for power-law fitting. Distance distribution frequencies are logarithmically binned. A shallower distance distribution slope implies less locality, as it means there is more communication at further distances. For Rentian distributions, there is a limiting value in the slope of -1 which corresponds to infinitely large dimensionality in the communication graph's connectivity. One-dimensional graphs (serial communication) have a limiting-case distance distribution slope of -2, or steeper. For the Rentian plots on the left, the size of the cross corresponds to the degeneracy (number of identical samples) at that sample point.

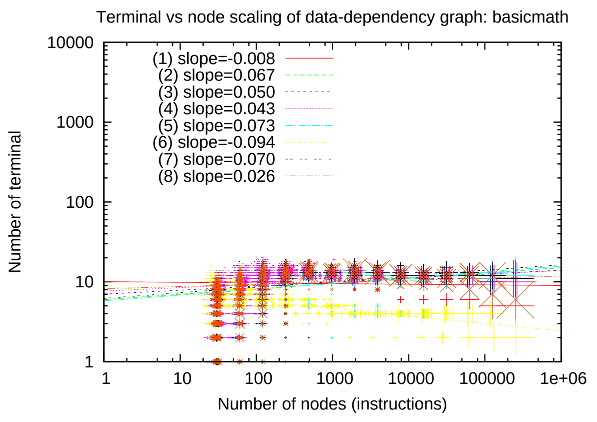
We should note that all fits are over the entire data-range, and therefore may include Region II effects that may alter the values of the slopes.

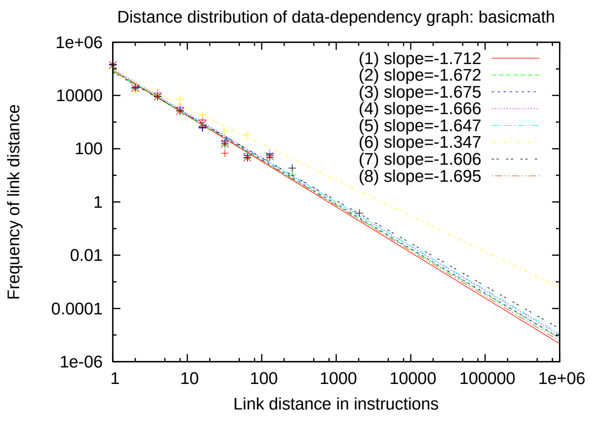

Terminal vs node scaling of data-dependency graph: adpcmdec
(1) slope=-0.016
(2) slope=-0.121
(3) slope=-0.009
(4) slope=-0.038
(5) slope=-0.149
(6) slope=-0.290
(7) slope=-0.055
(8) slope=-0.248
Number of terminal
Number of nodes (instructions)
10000
1000
100
10
1
1
10
100
1000
10000
100000
1e+06

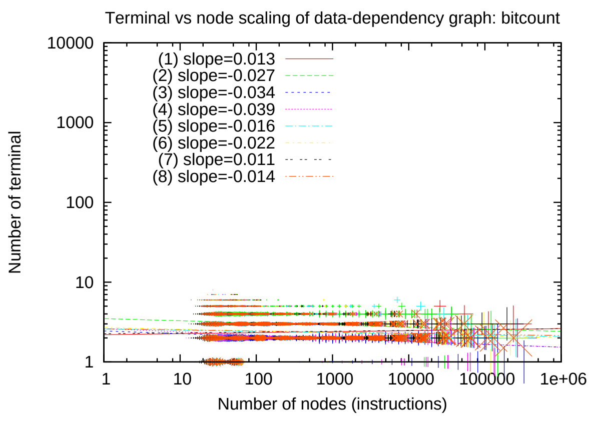

Distance distribution of data-dependency graph: adpcmdec
(1) slope=-1.840
(2) slope=-1.741
(3) slope=-1.528
(4) slope=-1.716
(5) slope=-2.228
(6) slope=-1.750
(7) slope=-1.825
(8) slope=-1.797
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06
1e-08
1e-10
1
10
100
1000
10000
100000
1e+06

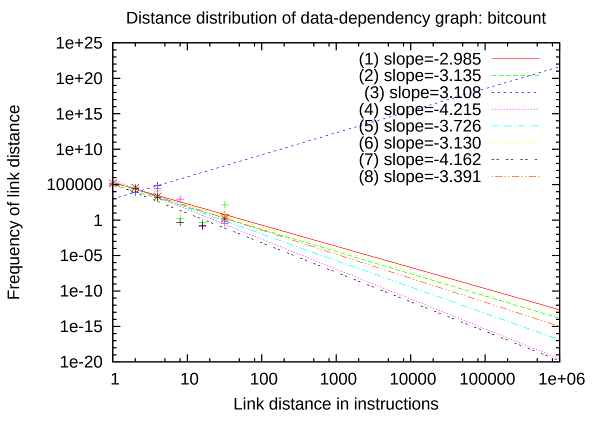

Terminal vs node scaling of data-dependency graph: adpcmenc
(1) slope=-0.145
(2) slope=-0.180
(3) slope=-0.253
(4) slope=-0.229
(5) slope=-0.151
(6) slope=-0.123
(7) slope=-0.079
(8) slope=-0.236
Number of terminal
Number of nodes (instructions)
10000
1000
100
10
1
1
10
100
1000
10000
100000
1e+06

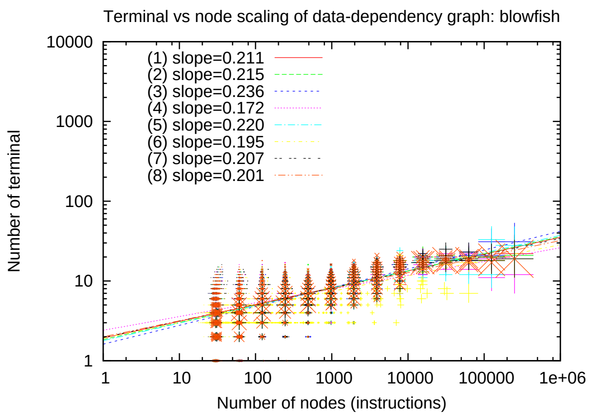

Distance distribution of data-dependency graph: adpcmenc
(1) slope=-1.422
(2) slope=-1.753
(3) slope=-1.603
(4) slope=-1.647
(5) slope=-1.811
(6) slope=-1.566
(7) slope=-1.393
(8) slope=-1.457
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06
1
10
100
1000
10000
100000
1e+06

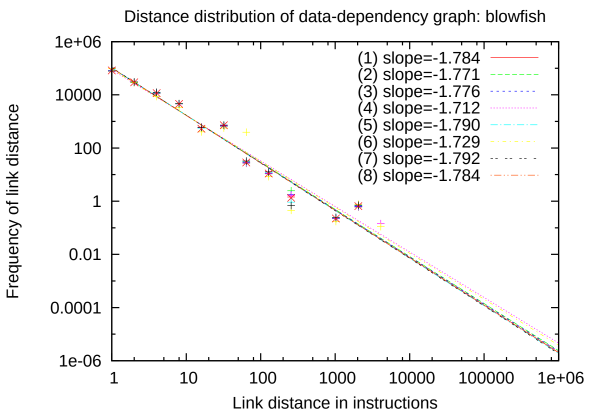

Terminal vs node scaling of data-dependency graph: basicmath
(1) slope=-0.008
(2) slope=0.067
(3) slope=0.050
(4) slope=0.043
(5) slope=0.073
(6) slope=-0.094
(7) slope=0.070
(8) slope=0.026
Number of terminal
Number of nodes (instructions)
10000
1000
100
10
1
1
10
100
1000
10000
100000
1e+06

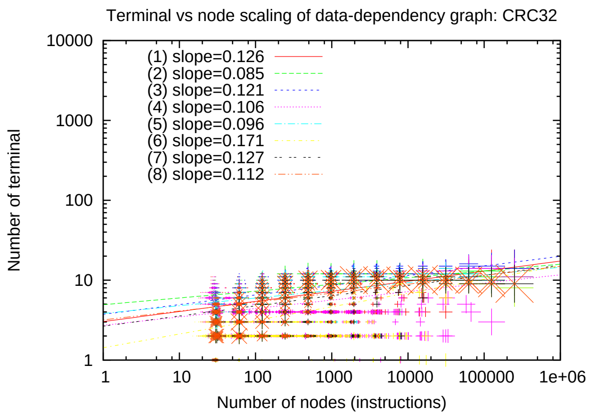

Distance distribution of data-dependency graph: basicmath
(1) slope=-1.712
(2) slope=-1.672
(3) slope=-1.675
(4) slope=-1.666
(5) slope=-1.647
(6) slope=-1.347
(7) slope=-1.606
(8) slope=-1.695
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06
1
10
100
1000
10000
100000
1e+06

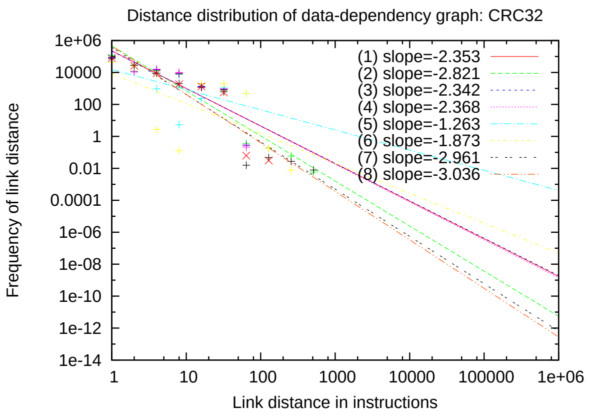

Terminal vs node scaling of data-dependency graph: bitcount
(1) slope=0.013
(2) slope=-0.027
(3) slope=-0.034
(4) slope=-0.039
(5) slope=-0.016
(6) slope=-0.022
(7) slope=0.011
(8) slope=-0.014
Number of terminal
Number of nodes (instructions)
10000
1000
100
10
1
1
10
100
1000
10000
100000
1e+06

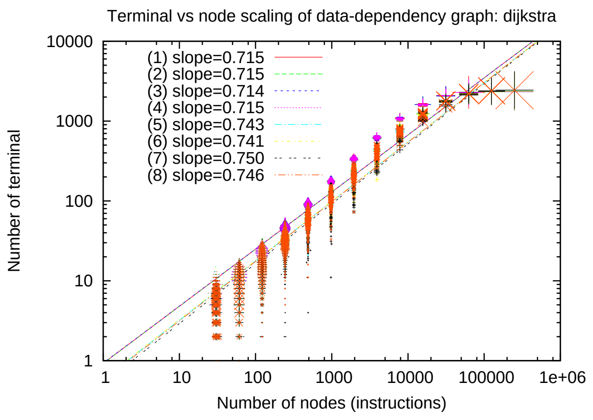

Distance distribution of data-dependency graph: bitcount
(1) slope=-2.985
(2) slope=-3.135
(3) slope=3.108
(4) slope=-4.215
(5) slope=-3.726
(6) slope=-3.130
(7) slope=-4.162
(8) slope=-3.391
Frequency of link distance
Link distance in instructions
1e+25
1e+20
1e+15
1e+10
100000
1
1e-05
1e-10
1e-15
1e-20
1
10
100
1000
10000
100000
1e+06

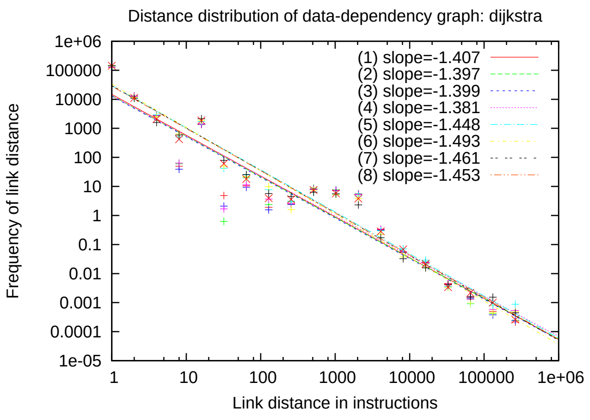

Terminal vs node scaling of data-dependency graph: blowfish
Number of terminal
Number of nodes (instructions)
(1) slope=0.211
(2) slope=0.215
(3) slope=0.236
(4) slope=0.172
(5) slope=0.220
(6) slope=0.195
(7) slope=0.207
(8) slope=0.201

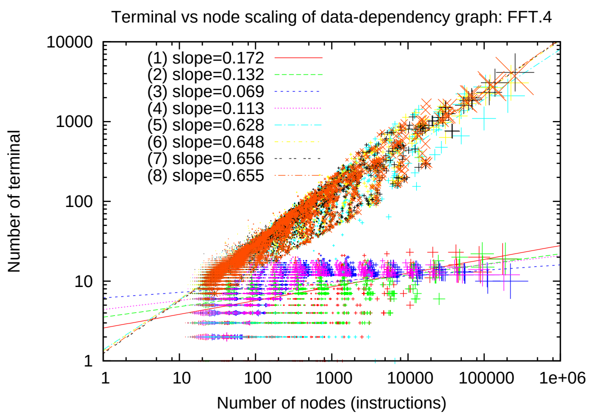

Distance distribution of data-dependency graph: blowfish
Frequency of link distance
Link distance in instructions
(1) slope=-1.784
(2) slope=-1.771
(3) slope=-1.776
(4) slope=-1.712
(5) slope=-1.790
(6) slope=-1.729
(7) slope=-1.792
(8) slope=-1.784

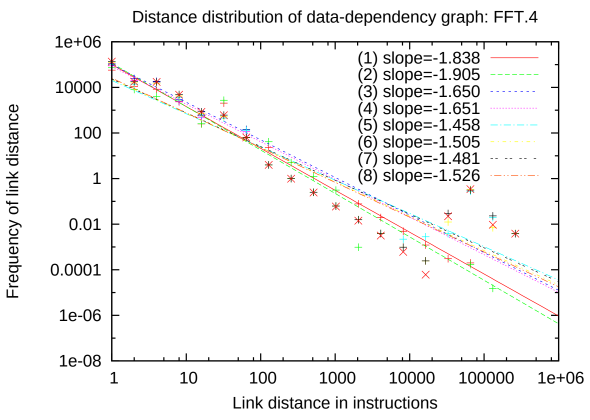

Terminal vs node scaling of data-dependency graph: CRC32
Number of terminal
Number of nodes (instructions)
(1) slope=0.126
(2) slope=0.085
(3) slope=0.121
(4) slope=0.106
(5) slope=0.096
(6) slope=0.171
(7) slope=0.127
(8) slope=0.112

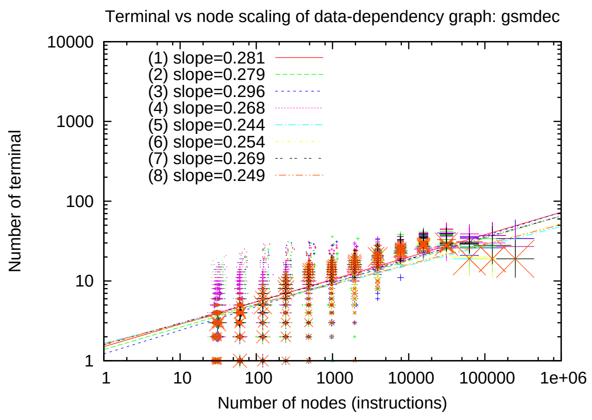

Distance distribution of data-dependency graph: CRC32
Frequency of link distance
Link distance in instructions
(1) slope=-2.353
(2) slope=-2.821
(3) slope=-2.342
(4) slope=-2.368
(5) slope=-1.263
(6) slope=-1.873
(7) slope=-2.961
(8) slope=-3.036

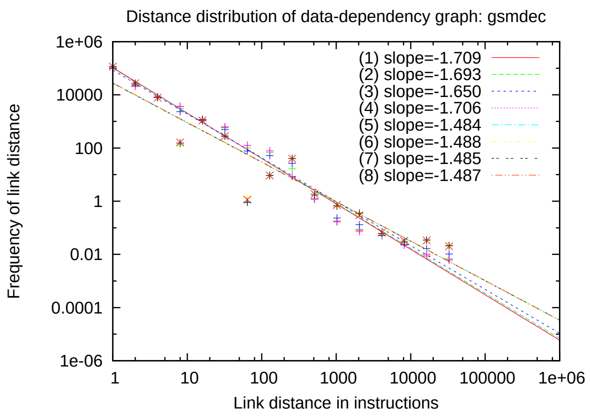

Terminal vs node scaling of data-dependency graph: dijkstra
Number of terminal
Number of nodes (instructions)
(1) slope=0.715
(2) slope=0.715
(3) slope=0.714
(4) slope=0.715
(5) slope=0.743
(6) slope=0.741
(7) slope=0.750
(8) slope=0.746

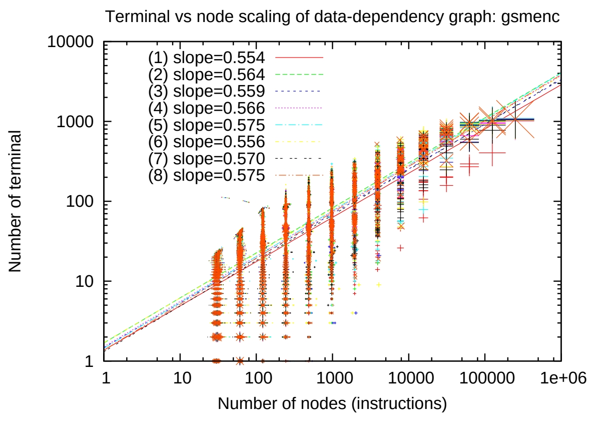

Distance distribution of data-dependency graph: dijkstra
Frequency of link distance
Link distance in instructions
(1) slope=-1.407
(2) slope=-1.397
(3) slope=-1.399
(4) slope=-1.381
(5) slope=-1.448
(6) slope=-1.493
(7) slope=-1.461
(8) slope=-1.453

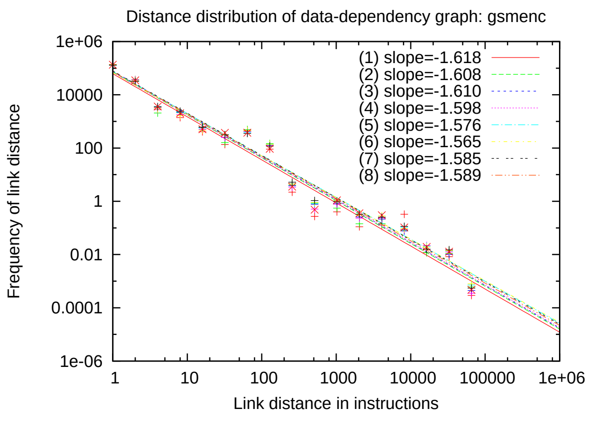

Terminal vs node scaling of data-dependency graph: FFT.4
Number of terminal
Number of nodes (instructions)
(1) slope=0.172
(2) slope=0.132
(3) slope=0.069
(4) slope=0.113
(5) slope=0.628
(6) slope=0.648
(7) slope=0.656
(8) slope=0.655

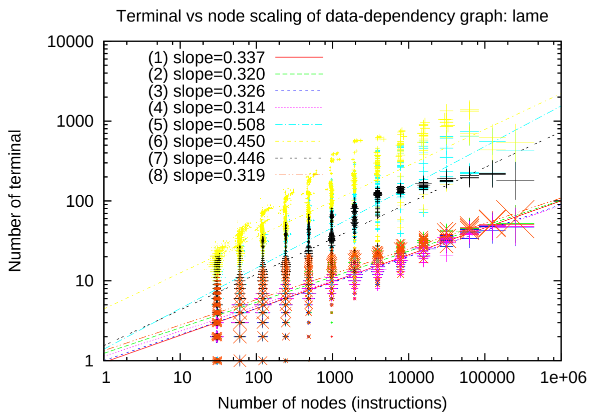

Distance distribution of data-dependency graph: FFT.4
Frequency of link distance
Link distance in instructions
(1) slope=-1.838
(2) slope=-1.905
(3) slope=-1.650
(4) slope=-1.651
(5) slope=-1.458
(6) slope=-1.505
(7) slope=-1.481
(8) slope=-1.526

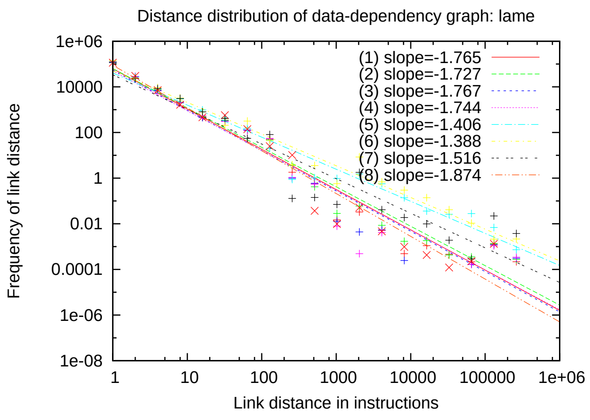

Terminal vs node scaling of data-dependency graph: gsmdec
(1) slope=0.281
(2) slope=0.279
(3) slope=0.296
(4) slope=0.268
(5) slope=0.244
(6) slope=0.254
(7) slope=0.269
(8) slope=0.249
Number of terminal
Number of nodes (instructions)
1
10
100
1000
10000
100000
1e+06

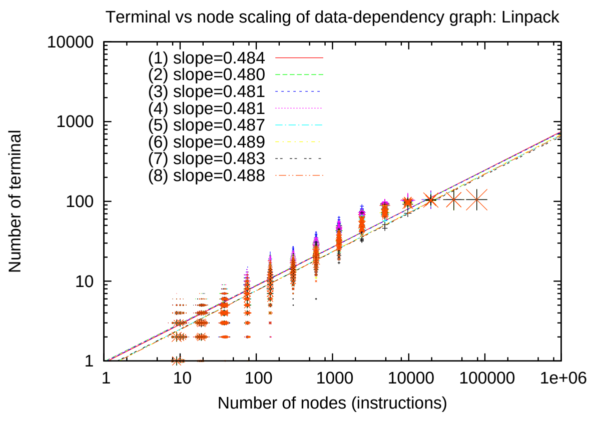

Distance distribution of data-dependency graph: gsmdec
(1) slope=-1.709
(2) slope=-1.693
(3) slope=-1.650
(4) slope=-1.706
(5) slope=-1.484
(6) slope=-1.488
(7) slope=-1.485
(8) slope=-1.487
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06

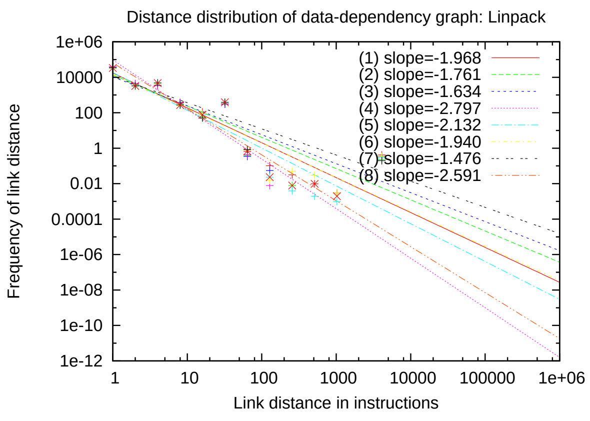

Terminal vs node scaling of data-dependency graph: gsmenc
(1) slope=0.554
(2) slope=0.564
(3) slope=0.559
(4) slope=0.566
(5) slope=0.575
(6) slope=0.556
(7) slope=0.570
(8) slope=0.575
Number of terminal
Number of nodes (instructions)

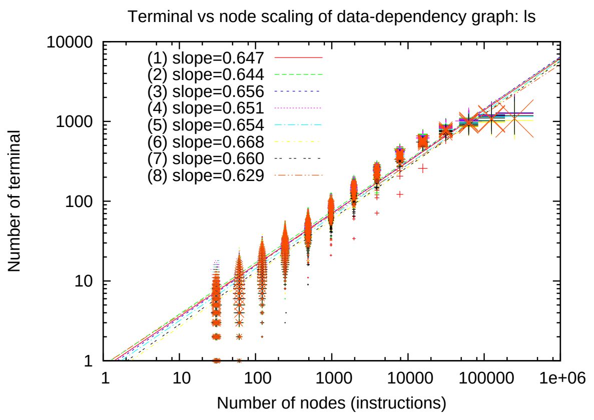

Distance distribution of data-dependency graph: gsmenc
(1) slope=-1.618
(2) slope=-1.608
(3) slope=-1.610
(4) slope=-1.598
(5) slope=-1.576
(6) slope=-1.565
(7) slope=-1.585
(8) slope=-1.589
Frequency of link distance
Link distance in instructions

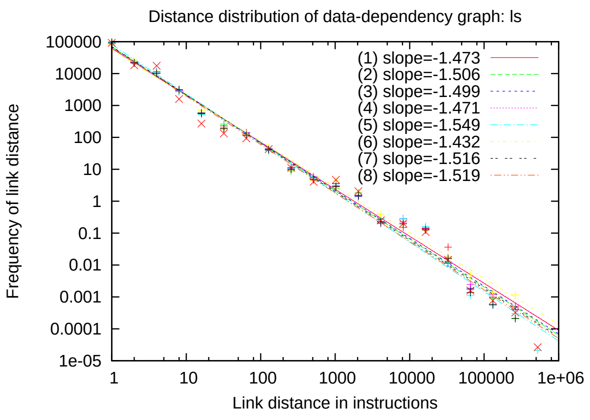

Terminal vs node scaling of data-dependency graph: lame
(1) slope=0.337
(2) slope=0.320
(3) slope=0.326
(4) slope=0.314
(5) slope=0.508
(6) slope=0.450
(7) slope=0.446
(8) slope=0.319
Number of terminal
Number of nodes (instructions)

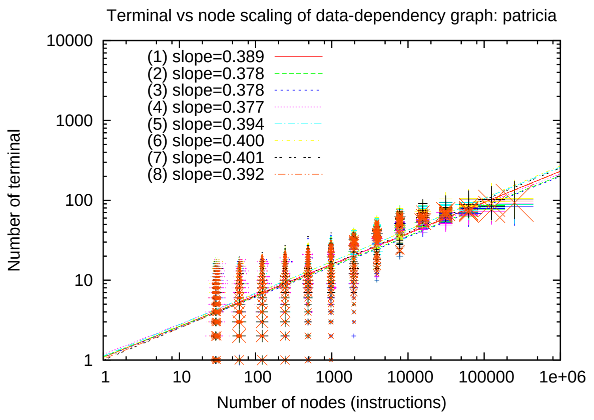

Distance distribution of data-dependency graph: lame
(1) slope=-1.765
(2) slope=-1.727
(3) slope=-1.767
(4) slope=-1.744
(5) slope=-1.406
(6) slope=-1.388
(7) slope=-1.516
(8) slope=-1.874
Frequency of link distance
Link distance in instructions
1e-08

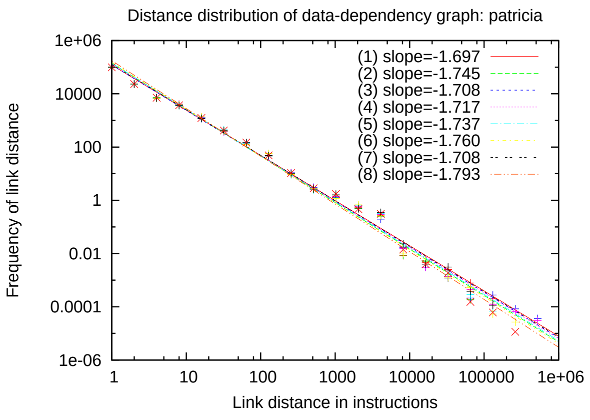

Terminal vs node scaling of data-dependency graph: Linpack
(1) slope=0.484
(2) slope=0.480
(3) slope=0.481
(4) slope=0.481
(5) slope=0.487
(6) slope=0.489
(7) slope=0.483
(8) slope=0.488
Number of terminal
Number of nodes (instructions)

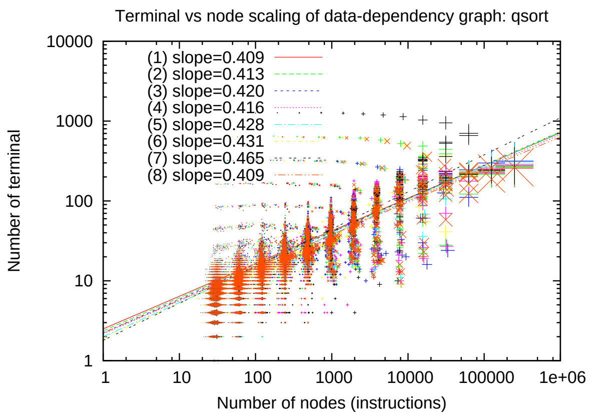

Distance distribution of data-dependency graph: Linpack
(1) slope=-1.968
(2) slope=-1.761
(3) slope=-1.634
(4) slope=-2.797
(5) slope=-2.132
(6) slope=-1.940
(7) slope=-1.476
(8) slope=-2.591
Frequency of link distance
Link distance in instructions
1e-10
1e-12

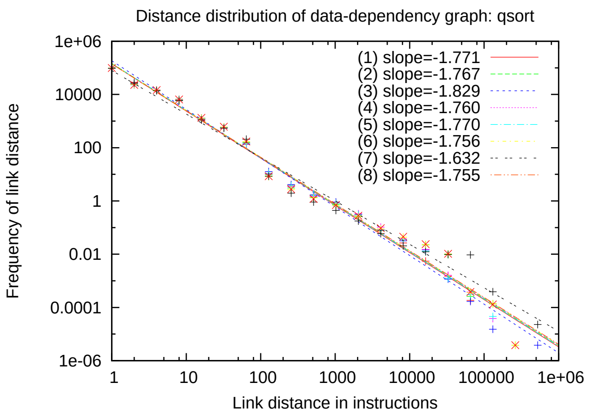

Terminal vs node scaling of data-dependency graph: ls
(1) slope=0.647
(2) slope=0.644
(3) slope=0.656
(4) slope=0.651
(5) slope=0.654
(6) slope=0.668
(7) slope=0.660
(8) slope=0.629
Number of terminal
Number of nodes (instructions)

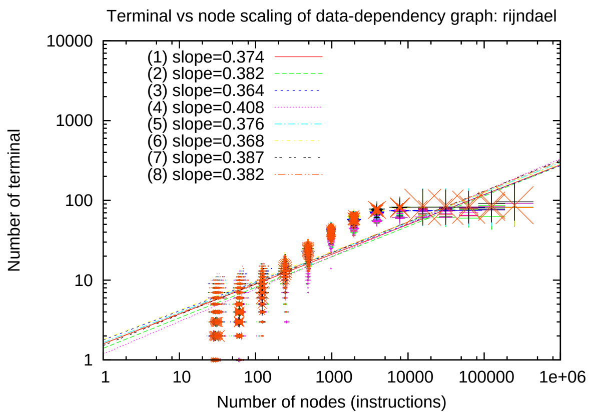

Distance distribution of data-dependency graph: ls
(1) slope=-1.473
(2) slope=-1.506
(3) slope=-1.499
(4) slope=-1.471
(5) slope=-1.549
(6) slope=-1.432
(7) slope=-1.516
(8) slope=-1.519
Frequency of link distance
Link distance in instructions

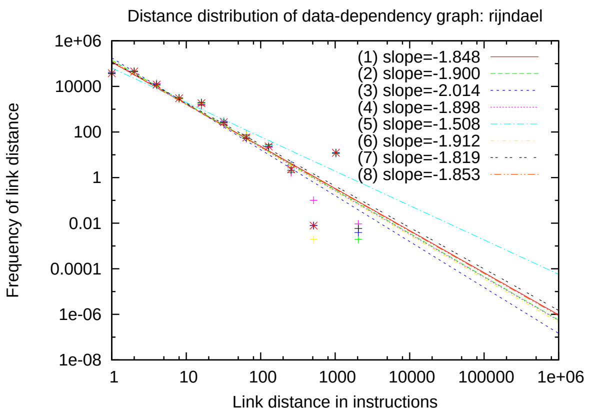

Terminal vs node scaling of data-dependency graph: patricia
(1) slope=0.389
(2) slope=0.378
(3) slope=0.378
(4) slope=0.377
(5) slope=0.394
(6) slope=0.400
(7) slope=0.401
(8) slope=0.392
Number of terminal
Number of nodes (instructions)

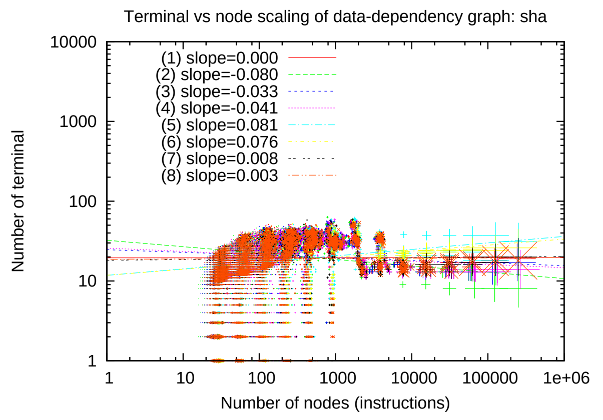

Distance distribution of data-dependency graph: patricia
(1) slope=-1.697
(2) slope=-1.745
(3) slope=-1.708
(4) slope=-1.717
(5) slope=-1.737
(6) slope=-1.760
(7) slope=-1.708
(8) slope=-1.793
Frequency of link distance
Link distance in instructions

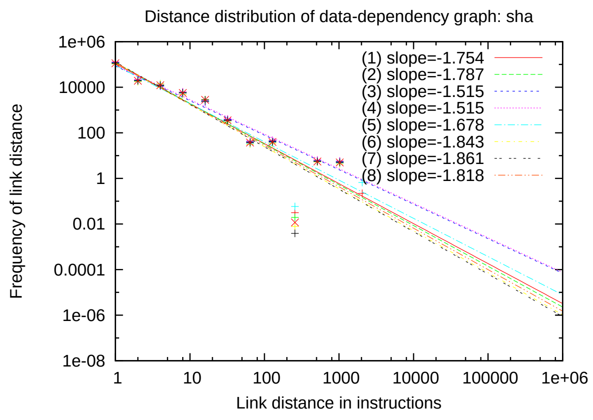

Terminal vs node scaling of data-dependency graph: qsort
(1) slope=0.409
(2) slope=0.413
(3) slope=0.420
(4) slope=0.416
(5) slope=0.428
(6) slope=0.431
(7) slope=0.465
(8) slope=0.409
Number of terminal
Number of nodes (instructions)

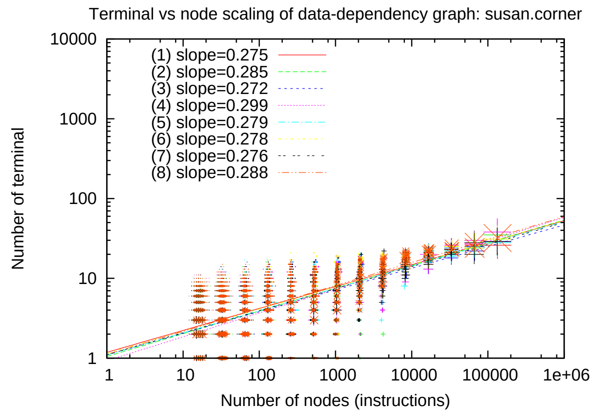

Distance distribution of data-dependency graph: qsort
(1) slope=-1.771
(2) slope=-1.767
(3) slope=-1.829
(4) slope=-1.760
(5) slope=-1.770
(6) slope=-1.756
(7) slope=-1.632
(8) slope=-1.755
Frequency of link distance
Link distance in instructions

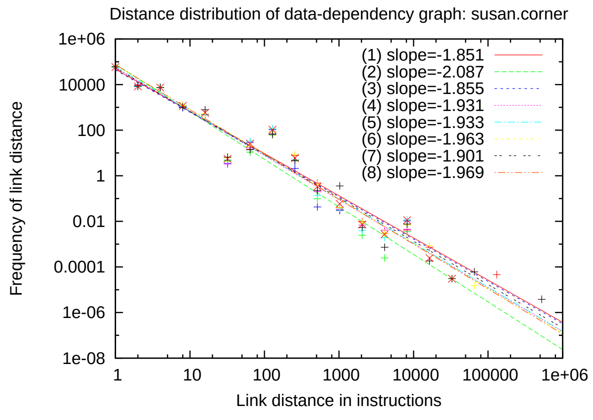

Terminal vs node scaling of data-dependency graph: rijndael
(1) slope=0.374
(2) slope=0.382
(3) slope=0.364
(4) slope=0.408
(5) slope=0.376
(6) slope=0.368
(7) slope=0.387
(8) slope=0.382
Number of terminal
Number of nodes (instructions)

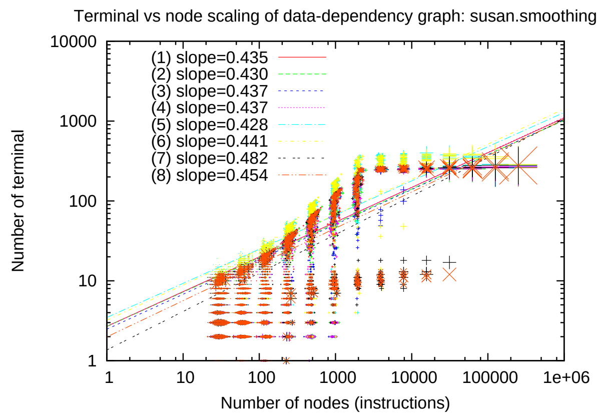

Distance distribution of data-dependency graph: rijndael
(1) slope=-1.848
(2) slope=-1.900
(3) slope=-2.014
(4) slope=-1.898
(5) slope=-1.508
(6) slope=-1.912
(7) slope=-1.819
(8) slope=-1.853
Frequency of link distance
Link distance in instructions

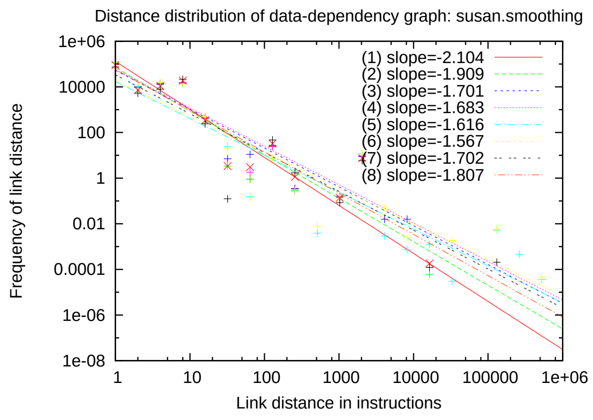

Terminal vs node scaling of data-dependency graph: sha
(1) slope=0.000
(2) slope=-0.080
(3) slope=-0.033
(4) slope=-0.041
(5) slope=0.081
(6) slope=0.076
(7) slope=0.008
(8) slope=0.003
Number of terminal
Number of nodes (instructions)

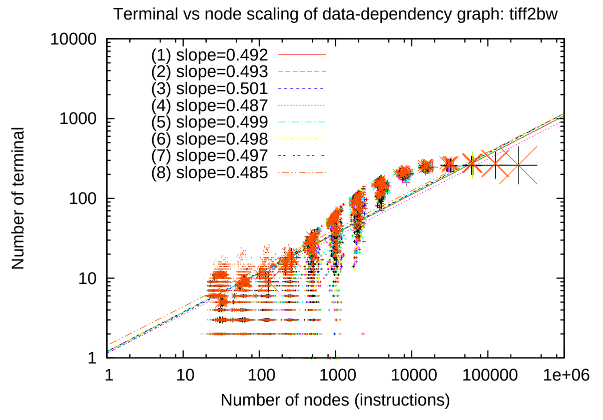

Distance distribution of data-dependency graph: sha
(1) slope=-1.754
(2) slope=-1.787
(3) slope=-1.515
(4) slope=-1.515
(5) slope=-1.678
(6) slope=-1.843
(7) slope=-1.861
(8) slope=-1.818
Frequency of link distance
Link distance in instructions

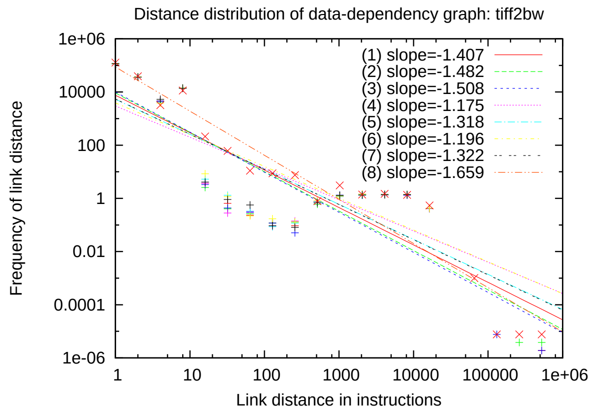

Terminal vs node scaling of data-dependency graph: susan.corner
(1) slope=0.275
(2) slope=0.285
(3) slope=0.272
(4) slope=0.299
(5) slope=0.279
(6) slope=0.278
(7) slope=0.276
(8) slope=0.288
Number of terminal
Number of nodes (instructions)

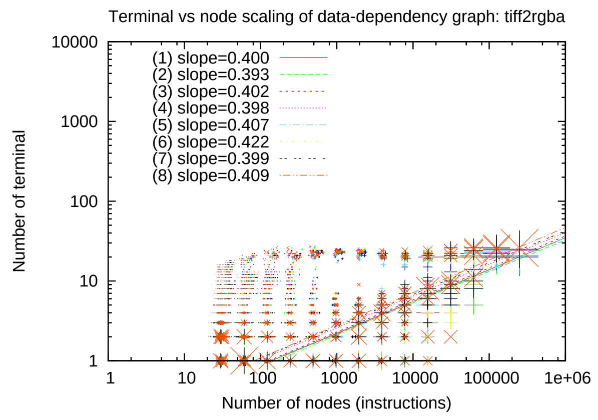

Distance distribution of data-dependency graph: susan.corner
(1) slope=-1.851
(2) slope=-2.087
(3) slope=-1.855
(4) slope=-1.931
(5) slope=-1.933
(6) slope=-1.963
(7) slope=-1.901
(8) slope=-1.969
Frequency of link distance
Link distance in instructions

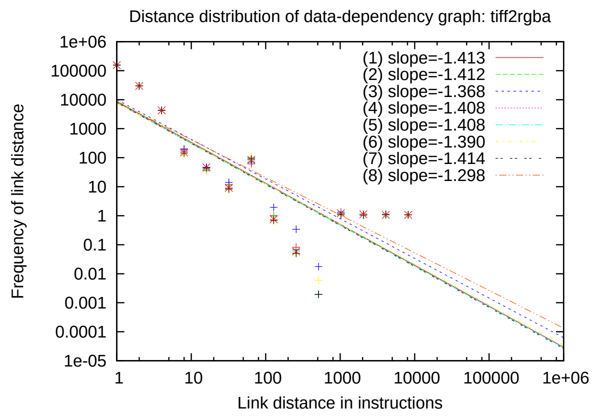

Terminal vs node scaling of data-dependency graph: susan.smoothing
(1) slope=0.435
(2) slope=0.430
(3) slope=0.437
(4) slope=0.437
(5) slope=0.428
(6) slope=0.441
(7) slope=0.482
(8) slope=0.454
Number of terminal
Number of nodes (instructions)

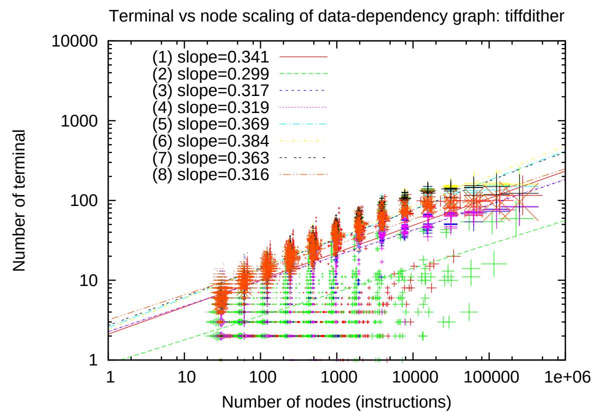

Distance distribution of data-dependency graph: susan.smoothing
(1) slope=-2.104
(2) slope=-1.909
(3) slope=-1.701
(4) slope=-1.683
(5) slope=-1.616
(6) slope=-1.567
(7) slope=-1.702
(8) slope=-1.807
Frequency of link distance
Link distance in instructions

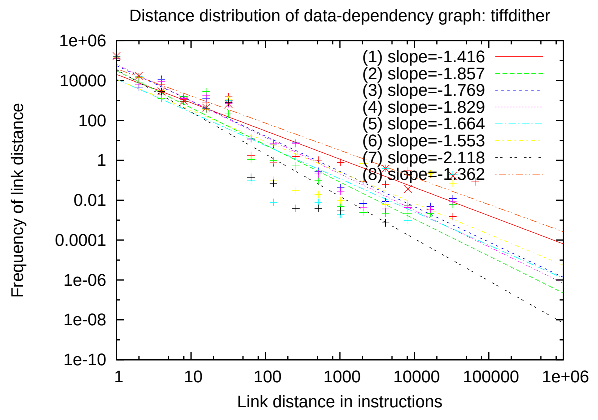

Terminal vs node scaling of data-dependency graph: tiff2bw
(1) slope=0.492
(2) slope=0.493
(3) slope=0.501
(4) slope=0.487
(5) slope=0.499
(6) slope=0.498
(7) slope=0.497
(8) slope=0.485
Number of terminal
Number of nodes (instructions)

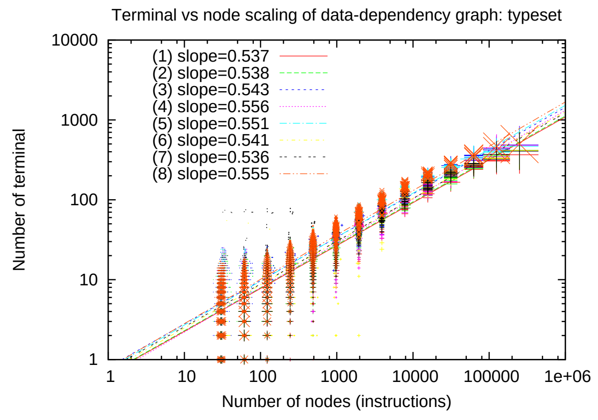

Distance distribution of data-dependency graph: tiff2bw
(1) slope=-1.407
(2) slope=-1.482
(3) slope=-1.508
(4) slope=-1.175
(5) slope=-1.318
(6) slope=-1.196
(7) slope=-1.322
(8) slope=-1.659
Frequency of link distance
Link distance in instructions

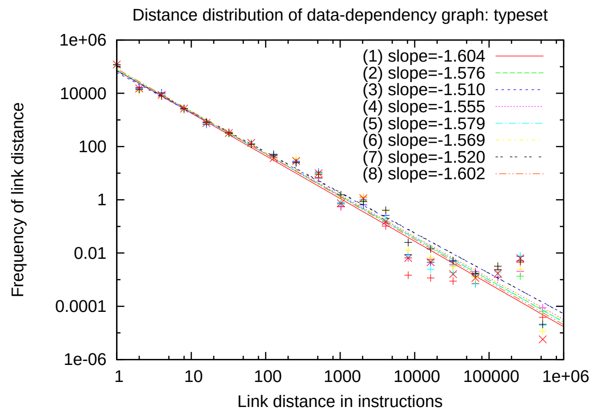

Terminal vs node scaling of data-dependency graph: tiff2rgba
(1) slope=0.400
(2) slope=0.393
(3) slope=0.402
(4) slope=0.398
(5) slope=0.407
(6) slope=0.422
(7) slope=0.399
(8) slope=0.409
Number of terminal
Number of nodes (instructions)
1
10
100
1000
10000
100000
1e+06


Distance distribution of data-dependency graph: tiff2rgba
(1) slope=-1.413
(2) slope=-1.412
(3) slope=-1.368
(4) slope=-1.408
(5) slope=-1.408
(6) slope=-1.390
(7) slope=-1.414
(8) slope=-1.298
Frequency of link distance
Link distance in instructions
1e+06
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05


Terminal vs node scaling of data-dependency graph: tiffdither
(1) slope=0.341
(2) slope=0.299
(3) slope=0.317
(4) slope=0.319
(5) slope=0.369
(6) slope=0.384
(7) slope=0.363
(8) slope=0.316
Number of terminal
Number of nodes (instructions)


Distance distribution of data-dependency graph: tiffdither
(1) slope=-1.416
(2) slope=-1.857
(3) slope=-1.769
(4) slope=-1.829
(5) slope=-1.664
(6) slope=-1.553
(7) slope=-2.118
(8) slope=-1.362
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06
1e-08
1e-10


Terminal vs node scaling of data-dependency graph: typeset
(1) slope=0.537
(2) slope=0.538
(3) slope=0.543
(4) slope=0.556
(5) slope=0.551
(6) slope=0.541
(7) slope=0.536
(8) slope=0.555
Number of terminal
Number of nodes (instructions)


Distance distribution of data-dependency graph: typeset
(1) slope=-1.604
(2) slope=-1.576
(3) slope=-1.510
(4) slope=-1.555
(5) slope=-1.579
(6) slope=-1.569
(7) slope=-1.520
(8) slope=-1.602
Frequency of link distance
Link distance in instructions
1e+06
10000
100
1
0.01
0.0001
1e-06

Appendix E

Spatio-Temporal Rentian Model: 32-core system

This section provides more detail on the Spatio-Temporal Rentian model utilised in Chapter 5 (section 5.2). In Chapter 7, a Spatio-Temporal Rentian model was examined for an $L \times L$ configuration of cores. For the 32-core system, this is instead accommodated by an arrangement of 6×6 with the four corners unoccupied. This alters the function $D\left(l_S\right)$ - the number of spatial links of length l_S which can instead be numerically calculated from this configuration. We then have the unnormalised configuration volume reachable at exactly distance r as:

$$Q\left(r\right) \equiv \sum_{k=0}^{r} D\left(k\right).$$

The unnormalised, total configuration volume reachable within distance q is then:

$$R\left(q\right) \equiv \sum_{r=0}^{q} Q\left(r\right).$$

The spatial bandwidth is calculated based on the number of cores. First, each node is taken as a seed node, then all nodes within a distance k of the seed node are counted as a region of S nodes. The bandwidth for this region of S nodes is given by the amount of communication from within the S nodes to outside the S nodes, which results in a weighted sum of powers of the configuration volume. There may be many such regions of size S nodes, so the average is taken. The expected amount of traffic between two spatial nodes i and j is given by a weighted sum of the configuration volume:

$$b \sum_{k} \lambda_{i,j,k} V_k^{p-1},$$

where b is the average bandwidth per node. Let d_{ij} be the distance between nodes i and j. Then $\lambda_{i,j,k}$ is:

$$\lambda_{i,j,k} = \begin{cases} 0 & , k < d_{ij} - 1 \\ \frac{D(d_{i,j})}{Q(d_{ij})} & , k = d_{ij} - 1 \\ \frac{D(d_{i,j})}{Q(k+1)} - \frac{D(d_{i,j})}{Q(k)} & , d_{ij} \leq k \end{cases}$$

With this configuration of 32 nodes, each node is reachable from every other node within 8 steps, so $\forall k > 8 : Q(k) = Q(8)$, trivially resulting in $\lambda_{i,j,k} = 0$. For the set of N nodes we have configuration volumes:

$$V_k = \frac{1}{|N|} R(k).$$

Let N_s be the set of regions with size of exactly s nodes (found by setting each node as a seed node). Then averaging over all regions $S \in N_s$ and summing over all nodes $i \in S$, $j \in N \setminus S$, the bandwidth $B_s(b, p)$ as a function of the average bandwidth b and spatio-temporal Rent's exponent p is:

$$B_s(b, p) = b \sum_k \left(\frac{1}{|N_s|} \sum_{S \in N_s} \sum_{i \in S} \sum_{j \in N \setminus S} \lambda_{i,j,k} \right) V_k^{p-1}.$$

We factor-out the parameter $\xi_{s,k}$ as:

$$\xi_{s,k} = \frac{1}{|N_s|} \sum_{S \in N_s} \sum_{i \in S} \sum_{j \in N \setminus S} \lambda_{i,j,k},$$

thus simplifying as:

$$B_s(b, p) = b \sum_k \xi_{s,k} V_k^{p-1}.$$

The best-fit is done in the logarithmic domain, which is:

$$\log(B_s(b, p)) = \log b + \log \left(\sum_k \xi_{s,k} V_k^{p-1} \right),$$

with partial derivatives:

$$\frac{\partial \log(B_s(b, p))}{\partial b} = \frac{1}{b}$$

$$\frac{\partial \log(B_s(b, p))}{\partial p} = \frac{\sum_k \xi_{s,k} V_k^{p-1} \log V_k}{\sum_k \xi_{s,k} V_k^{p-1}}.$$

We can then use Levenberg-Marquardt to fit this model to the experimental data.

Appendix F
Source code for numerical evaluation of Spatio-Temporal Rentian Model

Efficient numerical evaluation of the Spatio-Temporal Rentian model, as used in Chapter 7 is non-trivial, as the calculation spans many orders of magnitude that need to be successively approximated by integration. Thus the C++ source code is included below to help the reader.

RentianModel.cc

```
/*
 *  This program calculates the memory and I/O bandwidth requirements
 * using the Spatio-Temporal Rentian Model, given a side-length L of a square
 * arrangement of nodes, Rent's exponent and effective spatial cost psi.
 * Communication between nodes is as a 2-D mesh with simple Manhattan metric.
 * The number of nodes is N=L*L.
 *
 *  Command-line usage: RentianModel [L] [p] [psi]
 */
#include <iostream>
#include <math.h>

#define ERROR_TOLERANCE 0.0001


/* STRETCH specifies how many times further out to *directly* calculate beyond
 * the critical volume where each node can reach every other node. Beyond
 * this region, integration is used to approximate further, and the contribution
 * of the top 'cone' is neglected as negligible.
 */
#define STRETCH 100

#define MAX_MEM 2e10
```

```
using namespace std;

/*
 * D(ls, L) returns number of links with spatio-temporal distance ls for
 * side-length L
 */
double D(int _ls, int _L) {
  double ls = _ls;
  double L = _L;
  if (_ls==0)
    return L*L;
  if (_ls<_L)
    return ls*(ls*ls-1.0+6.0*L*(L-ls))*2.0/3.0;
  if (_ls<=2*_L)
    return (2.0*L-ls+1.0)*(2.0*L-ls)*(2.0*L-ls-1.0)*2.0/3.0;
  return 0.0;
}

/*
 * Q(r, L) returns non-normalised configuration volume reachable exactly at
 * distance r
 */
double Q(int _r, int _L) {
  double r = _r;
  double L = _L;
  double r2 = r*r;
  double r3 = r2*r;
  double r4 = r2*r2;
  double L2 = L*L;
  double L3 = L2*L;
  double L4 = L2*L2;
  if (_r < _L)
    return (6.0*L2-2.0*r-4.0*L*r+12.0*L2*r-r2-12.0*L*r2
            +12.0*L2*r2+2.0*r3-8.0*L*r3+r4)/6.0;
  if (_r < 2*_L)
    return (-4.0*L+4.0*L2+16.0*L3-10.0*L4+2.0*r-4.0*L*r-24.0*L2*r
            +32.0*L3*r+r2+12.0*L*r2-24.0*L2*r2-2.0*r3+8.0*L*r3-r4)/6.0;
  else return L4;
}
double *prob;   // Array of probabilities per 'era'
double *wprob;  // Array of length-weighted probabilities per 'era'
int critVal = 0; // Critical value where integral approximation is within
                 // ERROR_TOLERANCE of the summation

/*
 * Returns the weighted sum: Sum[((a+x)^(p-1)-(a+x+1)^(p-1))*x,{x,c,d}]
 */
double calcWeightedSum(double a, int c, int d, double p) {
  double sum = 0.0;
  for (int x=c; x<=d; ++x) {
    if (x!=0) {
      sum += (pow(a+x,p-1.0)-pow(a+x+1.0,p-1.0))*x;
    }
    }
    return sum;
  }

  /*
   * Approximates the calcWeightedSum function using an integral
   * returns Integral[((a+x)^(p-1)-(a+x+1)^(p-1))*x,{x,c-0.5,d+0.5}]
   */
  double calcWeightedIntegral(double a, double _c, double _d, double p) {
    double c = _c-0.5;
    double d = _d+0.5;
    return (pow(a+c,p)*(a-c*p)-pow(1.0+a+c,p)*(1.0+a-c*p)
            +pow(1.0+a+d,p)*(1.0+a-d*p)+pow(a+d,p)*(-a+d*p))/(p*(1.0+p));
  }

  /*
   * Uses calcWeightedIntegral, calcWeightedSum, or a linear combination of both
   * to come up with a fast approximation of calcWeightedSum that is within
   * ERROR_TOLERANCE of the correct value.
   */
  double calcWeightedHybrid(double a, double c, double d, double p) {
    if (a+(double)c > (double)critVal)
      return calcWeightedIntegral(a, c, d, p);
    if (a+(double)d <= (double)critVal)
      return calcWeightedSum(a, round(c), round(d), p);
    return calcWeightedSum(a, round(c), round(critVal-a), p)
            + calcWeightedHybrid(a, round(critVal-a+1.0), round(d), p);
  }

  /*
   * Finds the transition point for when the sum can be replaced with the
   * integral and be within ERROR_TOLERANCE of the correct value.
   */
  int findCriticalTransition(double p) {
    // the the value of 'l' for which thereafter it is within .1% error
    int a_x=1;
    while (true) {
      double X = calcWeightedIntegral(a_x, 0, 0, p);
      double Y = calcWeightedSum(a_x, 0, 0, p);
      double rel = 1.0-X/Y;
      if (rel < ERROR_TOLERANCE) {
        break;
      }
      a_x++;
    }
    return a_x;
  }

  /* Returns within ERROR_TOLERANCE tolerance the distance-weighted sum:
   *   Sum[((V0+x*Vdelta)^(p-1)-(V0+(x+1)*Vdelta)^(p-1))*x, {x, lstart, lend}]
   *     where V0+lstart*Vdelta = Vstart
   */
  double calcEither(double p, double Vstart, double Vdelta,
        double lstart, double lend) {
  if (critVal == 0) {
    critVal = findCriticalTransition(p);
  }
  double factor = pow(Vdelta, p-1.0); // factor out this term from the sum
  return factor*calcWeightedHybrid(Vstart/Vdelta-lstart, lstart, lend, p);
}

/*
 * Calculate and print out the details of the head region
 */
void calcHead(int L, double psi, double p) {
  double mem = 0;
  double cummProb = 0;
  int stepsize = 1;
  int count = 0;
  for (int l=1; l<=psi; l+=stepsize) {
    // When dealing with a large psi, it helps to exponentially grow the
    // stepsize, to speed up calculation
    if (count++ > 1000) {
      stepsize *= 2;
      count = 0;
    }
    double V0 = 1;  // initially each node can only reach itself
    double prob = 0;
    double wprob = 0;
    for (int r=0; r<=2*L; ++r) {
      double deltaV = Q(r,L)/pow(L,2);
      double V = V0+(double)(l-1.0)*deltaV;
      double Pbig = (pow(V,p-1.0)-pow(V+stepsize*deltaV,p-1.0));
      double prop = D(r,L)/Q(r,L);
      prob += prop * Pbig;
      wprob += prop * calcEither(p, V, deltaV, l, l+stepsize-1.0);
      V0 += psi*deltaV;
    }
    mem += wprob;
    cummProb += prob;
    cout << "0\t" << mem << "\t" << (1.0-cummProb) << "\t" << cummProb << endl;
  }
}

/*
 * Calculate and print out details of the transition region
 */
double calcTransition(int L, double psi, double p) {
  double V = 1.0; // initially each node can only reach itself
  for (int r=0; r<=2*L; ++r) {
    double deltaV = Q(r,L)/pow(L,2);
    double Pbig = (pow(V,p-1.0)-pow(V+psi*deltaV,p-1.0));
    for (int era=0; era<=r; ++era) {
      double prop = D(r-era,L)/Q(r,L);
      prob[era] += prop * Pbig;
      wprob[era] += prop
```

```
          * calcEither(p, V, deltaV, (double)era*psi+1.0, (double)(era+1.0)*psi);
      }
      V += psi*deltaV;
    }

// beyond the Vcrit region, we have a simpler relationship
  double deltaV = L*L;
  for (int r=2*L+1; r<STRETCH*2*L; ++r) {
    double Pbig = pow(V,p-1.0)-pow(V+psi*deltaV,p-1.0);
    for (int era = r-2*L; era<=r; ++era) {
      double prop = D(r-era,L)/Q(r,L);
      prob[era] += prop * Pbig;
      wprob[era] += prop *
           calcEither(p, V, deltaV, (double)era*psi+1.0, (double)(era+1.0)*psi);
    }
    V += psi*deltaV;
  }



// print out the results (excluding the 'top cone' that hasn't been filled)
  double mem = 0, cummProb = 0;
  for (int i=0; i<=(STRETCH-1)*2*L; ++i) {
    mem += wprob[i];
    cummProb += prob[i];
    cout <<i<<"\t"<<mem<<"\t"<<(1.0-cummProb)<<"\t"<<cummProb<<endl;
  }
  return V;
}



/*
 * Calculate and print out the details of the tail region
 */
void calcTail(double Vtail, int L, double psi, double p) {
  double mem = 0, cummProb = 0;
  // compute memory and BW for entire region including the ending 'cone region'
  for (int i=0; i<STRETCH*2*L; ++i) {
    mem += wprob[i];
    cummProb += prob[i];
  }



// starting at Vtail, we can now add a very large configuration volume
  // lets start with a spatio-temporal length of 8*L*psi
  double S = STRETCH*2.0*(double)L;
  double deltaS = 8.0*L;
  double deltaV = L*L;
  double correctionFactor = 0.0;
  for (int y=1; y<=2*L; ++y) {
    correctionFactor += D(y, L) * y;
  }
```

```
// Compute up to a memory size of MAX_MEM words
  while (mem < MAX_MEM) {
    cummProb += pow(Vtail,p-1.0)-pow(Vtail+psi*deltaS*deltaV,p-1.0);
    mem += calcEither(p, Vtail, deltaV, S*psi, (S+deltaS)*psi-1.0);
    // correct for the conical influence of spatio-temporal configuration space
    {
      double Vstart = Vtail;
      double lstart = S*psi;
      double a = Vstart/deltaV-lstart;
      double c = lstart-0.5;
      double d = lstart+deltaS*psi-1.0+0.5;
      double factor = pow(deltaV, p-1.0);
      mem -= correctionFactor * factor *
             (pow(a+d,p)-pow(a+d+1.0,p)-pow(a+c,p)+pow(a+c+1.0,p));
    }
    S += deltaS;
    cout << S<<"\t"<< mem<<"\t"<< (1.0-cummProb)<<"\t"<<cummProb<<"\n";
    Vtail += psi*deltaS*deltaV;
    deltaS *= 2.0;
  }
}


/*
 *  Command-line usage: RentianModel [L] [p] [psi]
 */
int main(int argc, char *argv[]) {
  if (argc < 4) {
     cerr << "Usage: RentianModel [L] [p] [psi]" << endl;
     return -1;
  }
  int L = atoi(argv[1]);
  double p = atof(argv[2]);
  double psi = atoi(argv[3]);
  int N = L*L;
  cout << "### Distance\tMemPerCore\tBWPerCore\tN = " << N << ", p = " << p
          <<", psi = " << psi << endl;
  prob = new double[STRETCH*2*L];
  wprob = new double[STRETCH*2*L];
  // Initialise
  for (int i=0; i<STRETCH*2*L; ++i) {
    prob[i] = wprob[i] = 0.0;
  }
  double Vtail; // configuration volume for starting the tail-region calculation
  calcHead(L, psi, p);
  Vtail = calcTransition(L, psi, p);
  calcTail(Vtail, L, psi, p);
  return 0;
}
```