UNIVERSITY OF CAMBRIDGE
Computer Laboratory

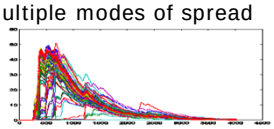
Data-Centric Systems and Networking

Contact: eiko.yoneki@cl.cam.ac.uk

Digital Epidemiology

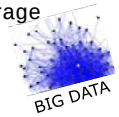
- Raspberry Pi based delay tolerant networks with satellite connectivity in developing countries

- Real world mobility data collection in Africa
- Analyse network structure to understand infectious disease spread
  - Multiple modes of spread



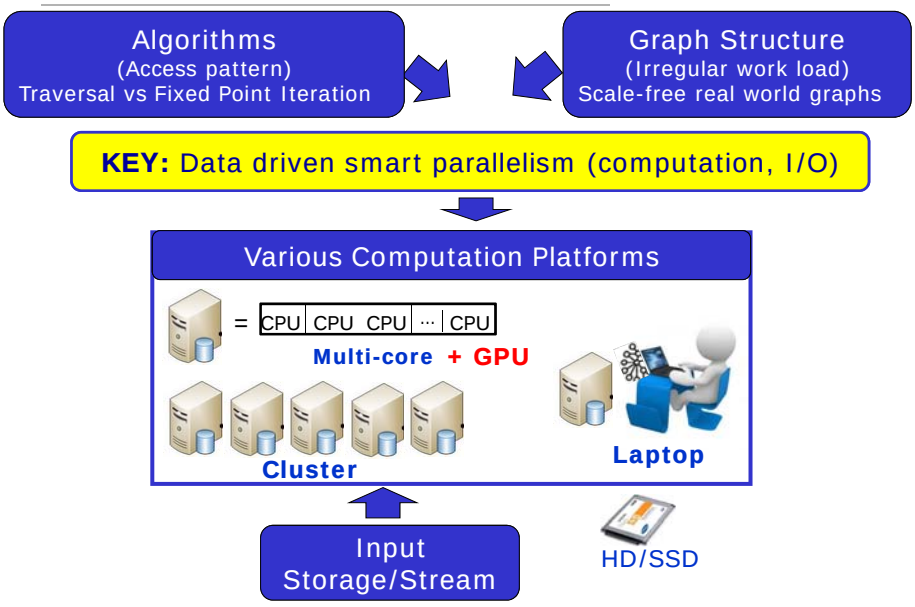
Graph Specific Data Parallel

- Fast, flexible, and programmable graph processing
- Cost effective but efficient storage
  - Move to SSDs from RAM
- Reduce latency
  - Runtime prefetching
  - Graph algorithm specific runtime
  - Dynamic CPU/GPU scheduling
- Super efficient parallel processing
- Reduce storage requirements
  - Compressed adjacency lists
- Build efficient data analytic framework without huge computing resources
- Search/update real time (Graph DB)



UNIVERSITY OF CAMBRIDGE
Computer Laboratory

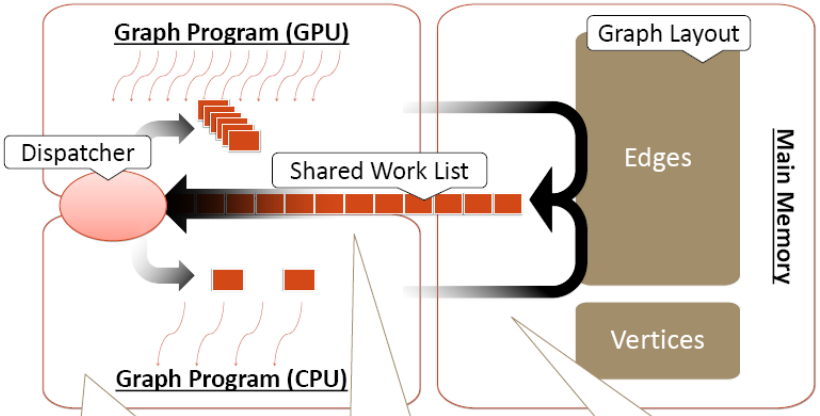
Large-scale Graph Processing



UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Heterogeneous Workload Management for Large-Scale Graph Processing

Karthik Nilakant and Dr Eiko Yoneki
Karthik.Nilakant@cl.cam.ac.uk / Eiko.Yoneki@cl.cam.ac.uk

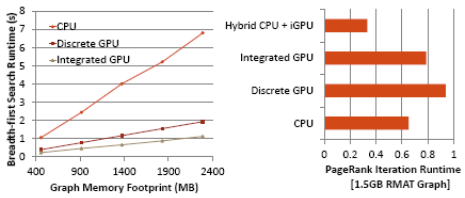


Heterogeneous Partitioning
- Run similar program instances on each device
- Aim to dispatch data to best-suited device
- Analyse work list elements to characterise data

Batch Management
- Trade off element analysis complexity with efficiency gains
- Elements can be classified individually, aggregated in batches, or dispatched speculatively

Memory Interface
- Batching needs to take into account local / global memory layout
- Attempt to hide GPU / CPU bus latency with higher parallelism

Early Results – APU Architecture
- Accelerated Processing Units provide higher-bandwidth DMA
- Left figure shows relative performance advantage of integrated GPU
- Right figure shows results of basic implementation of above workflow on APU



Research Systems group srg@cambridge:~$

UNIVERSITY OF CAMBRIDGE

Microsoft Research

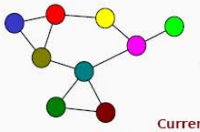
This work is supported by Microsoft Research through its PhD Scholarship Programme.

Optimizing Graph Computations: Trading Communications for Computations

Valentin Dalibard, Eiko Yoneki
Cambridge University Computer Laboratory
valentin.daliba@cl.cam.ac.uk

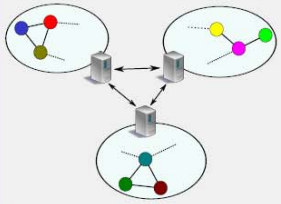
UNIVERSITY OF CAMBRIDGE

Example problem: Compute PageRank on a large graph

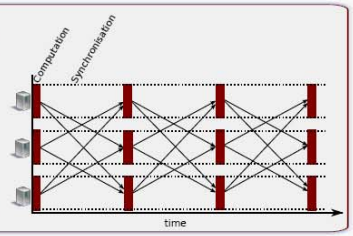


$$R(i) = \frac{\alpha}{N} + (1-\alpha) \times \sum_{j \to i} \frac{R(j)}{N_j}$$

Current Solution:
Distributed Bulk Synchronous Processing



- Split the graph among machines
- **Synchronization** between each iteration
- Imbalance between **computations** and **communcations**
- Overhead of about 10X on 64 machines (PowerGraph)



Solution proposed: Treat the computation as a distributed optimization

Change Computation Model

Allow computations on **not up-to-date values**

Allow **dynamic computation order** to optimise convergence

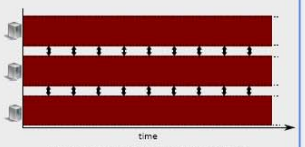
Change View

Goal of PageRank is to find a **fixed point**.

Treat it as a **distributed optimization**: aim is to minimize change between iterations

Make **full use** of resources to converge to this fixed point.

Change Execution Model



Constant Computation and Communication

Execution model

Use spare computations to:
- Perform **more graph computations** to converge faster towards the fixed point
- Assign **priorities to computations** to perform the more important ones first
- Assign **priorities to messages** to transfer the more important ones first

Computations

Transfer highest priority messages at **full bandwidth** to other machines

Communications

Challenges

Find the subset of graph computations that fit this model:
- Works well for **Shortest Paths** algorithms, **Greedy Graph Coloring** or **Connected Components**
- Doesn't work for **Gibbs Sampling** which always needs up-to-date inputs

Find the right **balance** between actual graph computations and meta-computations

Avoid making the model **difficult to use** for the programmer

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

SAKYOMI: SSD Prefetcher for Large-Scale Graph Traversal

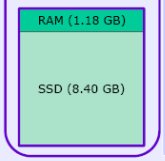
Eiko Yoneki, Karthik Nilakant, Valentin Dalibard (University of Cambridge) and Amitabha Roy (EPFL)

SAKYOMI: Cost-effective Graph Processing with Semi-External Memory

Main Memory (RAM) — Can hold program state, iterator state (work list), adjacency list index and disk cache

External Storage (SSD) — Can hold adjacency list, edge weights and other edge-specific data

Example: Memory layout for Twitter graph



Issues

- With random access to disk-based data, standard caching is inefficient
- Poor I/O queue depth and storage bandwidth utilisation with most synchronous graph programs

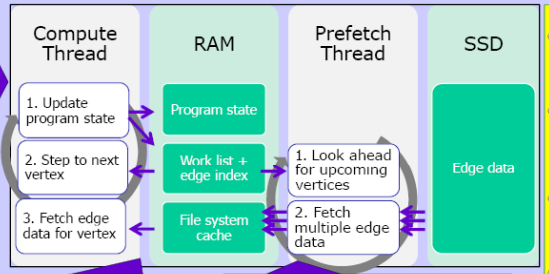
Insight

- Use the work list to predict future accesses to edge data
- Issue multiple concurrent I/O requests to load this data into cache

I/O Latency Mitigation with SAKYOMI

Example: BFS

1. Mark current vertex visited, add unvisited neighbours to queue
2. Pop next vertex from queue
3. Find neighbours of current vertex Repeat until queue is empty



- Computation stalls not-yet cached requested page of memory-mapped edge file
- Prefetcher reduces such page faults by fetching edge data directly from memory

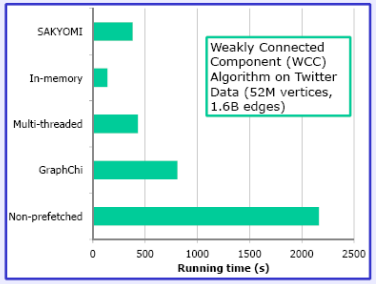
- SAKYOMI has predefined look-ahead functions for common work lists (Sequential / FIFO / Priority Queue) and allows custom look-ahead via callbacks

Advantages

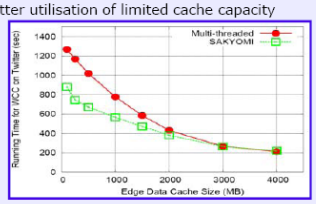
- Dijkstra's algorithm completed on a graph (315GB edge data) in 10 hours, using only 30GB RAM
- Unlike GraphChi, SAKYOMI can easily adapt to changing resource conditions w/o re processing edge data
- Various algorithms adapted (connected components, SSSP, K-Cores, SSSP and A* search)
- Simpler alternative to multi-threaded graph traversal, avoiding need to implement concurrency control

Key Results

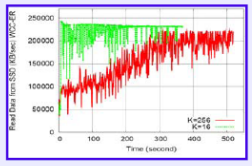
- Runtime only 1.5x to 5x slower than in-memory programs, despite requiring hardware platform at least 10x cheaper



- Better utilisation of limited cache capacity



- Immediate saturation of SSD random read throughput capacity with WCC on ER with high K value (average degree)



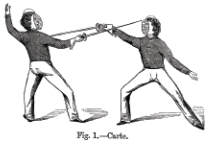
THE GAME-THEORY OF TECHNOLOGY USE

A. RIBEIRO, E. YONEKI

Technology and media as instruments for cooperation and competition. What problems groups solve with them, and how they can improve. New models for GUI, Website and Knowledge Repository use.

Graphical Games: Human-Computer Communication, Game-Theory and Applications - UIST'12

A Model of Learning in Low Income Communities: Coordination and Recommendation Systems - ICMLA'12



The 'Super-Cool' Formal Player - w L. Gu

How Cambridge students choose parties. Model of Power and popularity. "Super-Cool" player strategy. A role-playing game verifies his power empirically.

Coordination Games for Graph Visualization - AAAI'13 (submitted)

Real time Game-theoretical visualization of large networks. Application on how communities self-distribute to best represent knowledge. Study and model of Wikipedia evolution.