Scalability! But at what COST?

Frank McSherry, Michael Isard, Derek G. Murray (2015)

Jakub Bachurski (jkb55), 28 October 2024

Scalability!!!

5.2 GraphX Performance

Scaling: In Figure 8 we evaluate the strong scaling performance of GraphX running PageRank on the Twitter follower graph. As we move from 8 to 32 machines (a factor of 4) we see a 3x speedup. However as we move to 64 machines (a factor of 8) we only see a 3.5x speedup. While this is hardly linear scaling, it is actually slightly better than the 3.2x speedup reported by GraphLab [13]. The poor scaling performance of PageRank has been attributed by [13] to high communication overhead relative to computation for the PageRank algorithm.

[Gonzalez et al: GraphX, 2014]

But at what COST?

(Configuration to Outperform a Single Thread)

Scalability scales up with *carelessness*

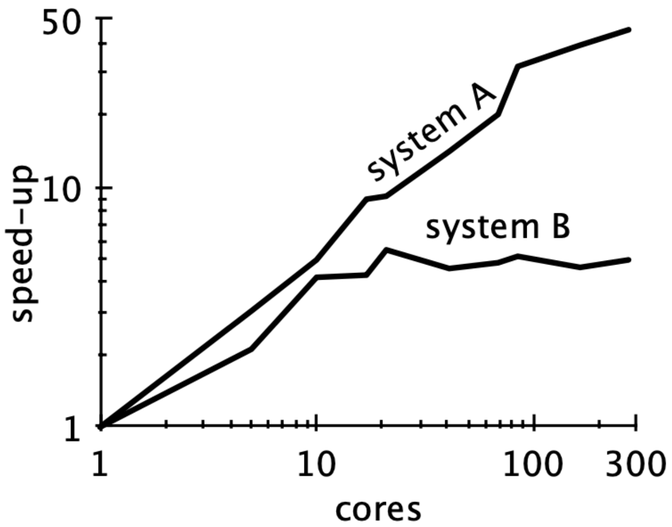
```
def main(num_threads, thread_idx):
    if thread_idx == 0:
        do_work()
    time.sleep(10.0 / num_threads)
```

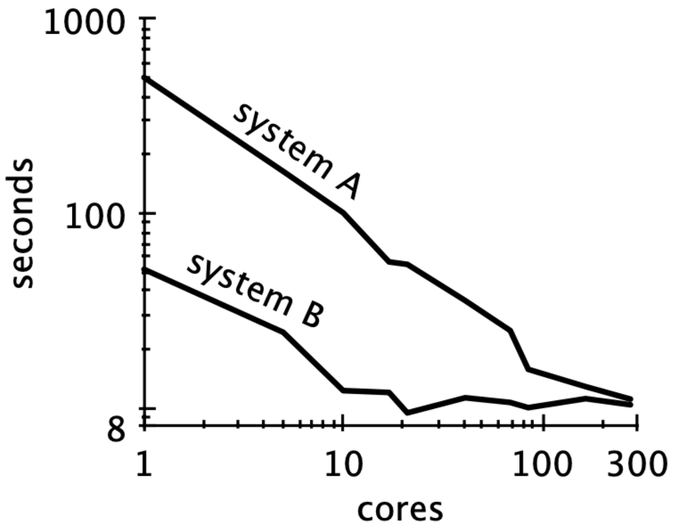
Scalability scales down with *carefulness*

```
def main(num_threads, thread_idx):
    if thread_idx == 0:
        do_work()
#   time.sleep(10.0 / num_threads)
```

Optimising a system breaks scalability

Authors' example on Naiad [McSherry et al., 2012]



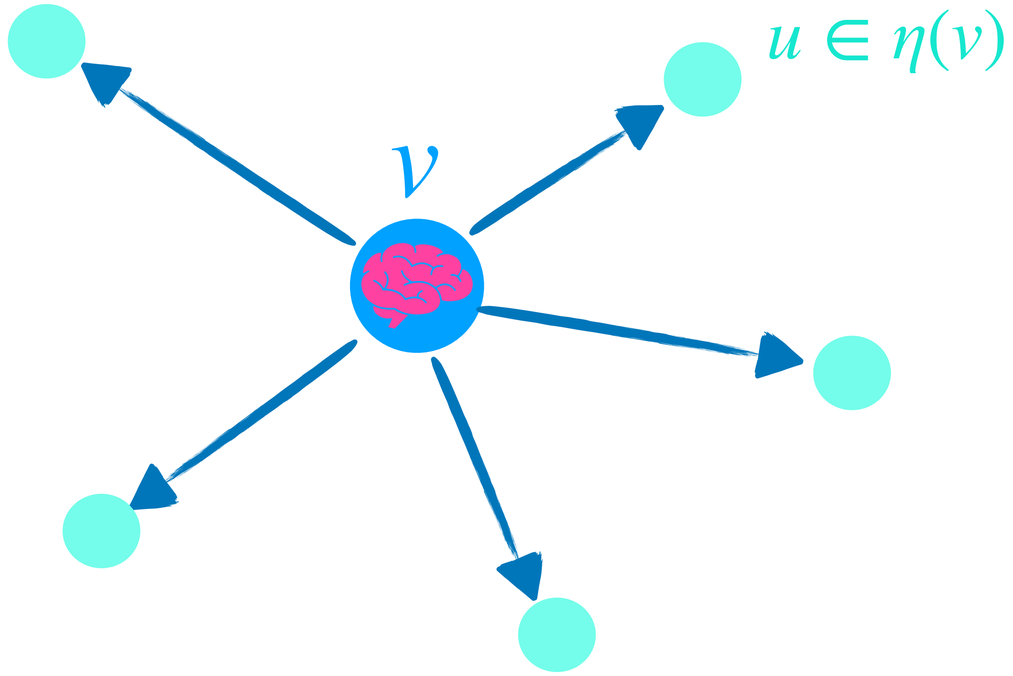


Parallelisation is hard…

Case in point: graph algorithms

«*Think like a vertex*» model

[Malewicz et al.: Pregel, 2010]



PageRank

Iterated vertex accumulation

$$a_v \quad \mapsto^* \quad \frac{1-\alpha}{|V|} + \alpha \sum_{u \in \eta(v)} \frac{a_u}{\deg u}$$

Label propagation

Finding connected components… like a vertex

$$\ell_v \quad \mapsto^* \quad \min\left(\ell_v, \min_{u \in \eta(v)} \ell_u\right)$$

Parallelisation *is* hard!

Most parallel systems need orders of magnitude more resources to match a single-thread.

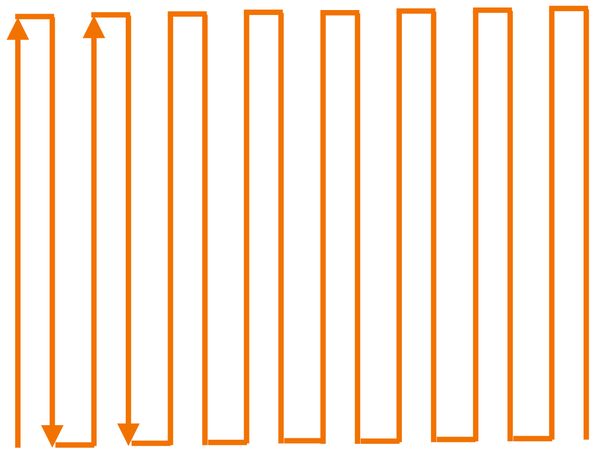
COST ≈ 10, 100

Going above and beyond

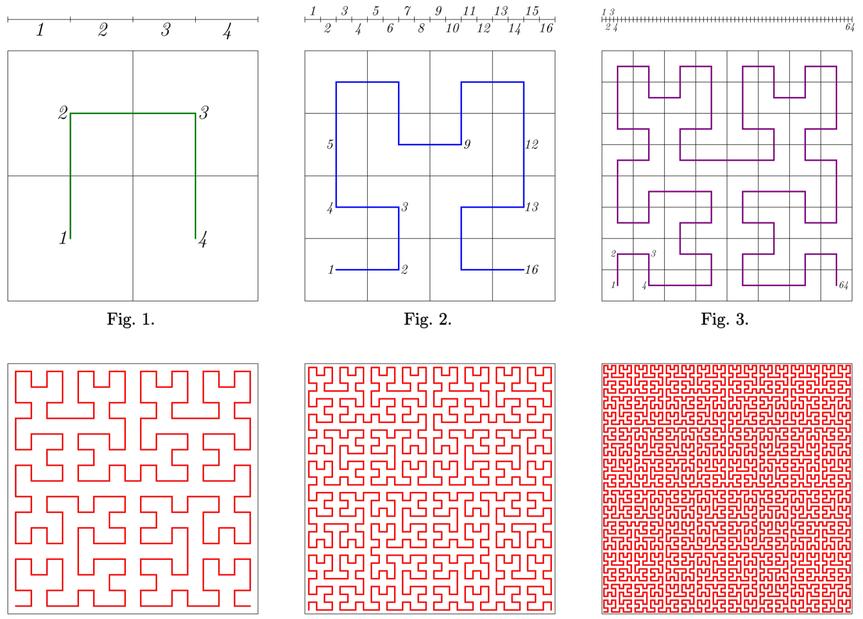
Improving memory layout

It’s always the memory

- Accessing data, especially in distributed systems, is **really expensive**
- Pre-processing work *(Hilbert space-filling curve)* ⇒ Performance gains!



⟹



Abandon yer models!

A foolish consistency is the hobgoblin of little minds [Emerson]

- Move beyond the *«think like a vertex»* model [2010] and refer to literature
- Algorithms for connected components in near-linear time (vs quadratic)
  - Find & union (disjoint sets) data structure [Galler and Fischer, 1964]
  - Borůvka's algorithm [1926] → 1938, 1951, 1965, …
- Much of existing literature won’t fit recent simple models…

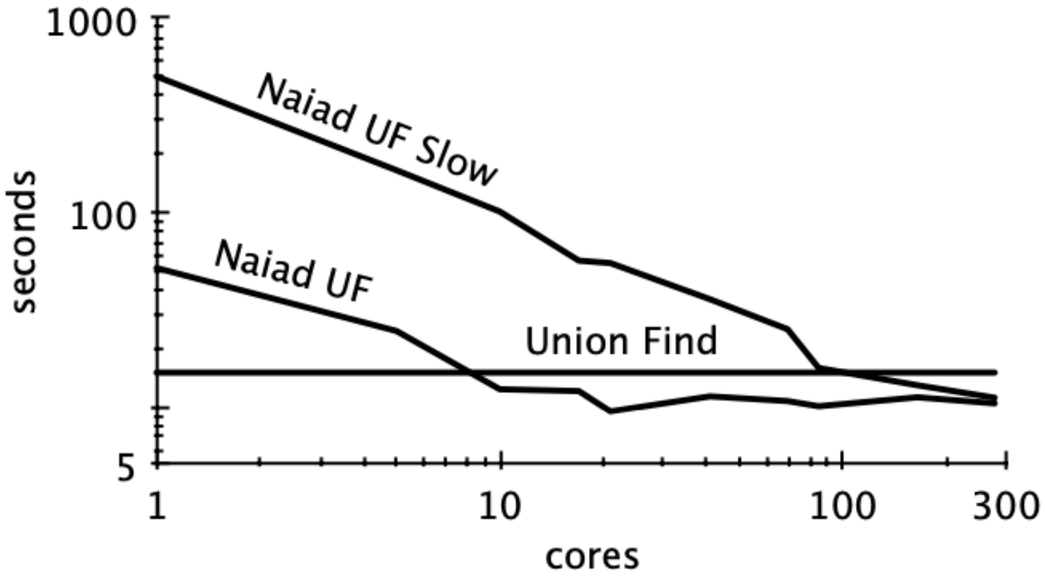
Parallelisation is *really* hard!

COST' ≈ 100, 1000, …, +∞

Results

Parallelisation overheads

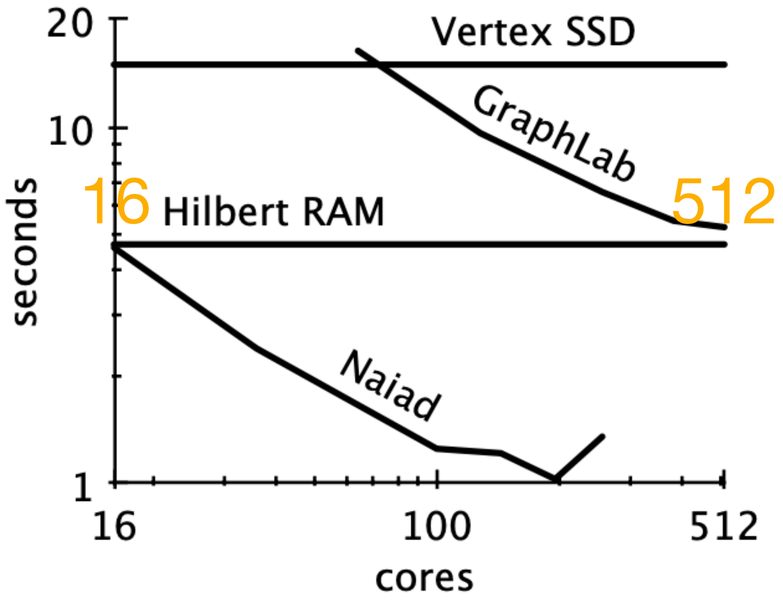
Connected components



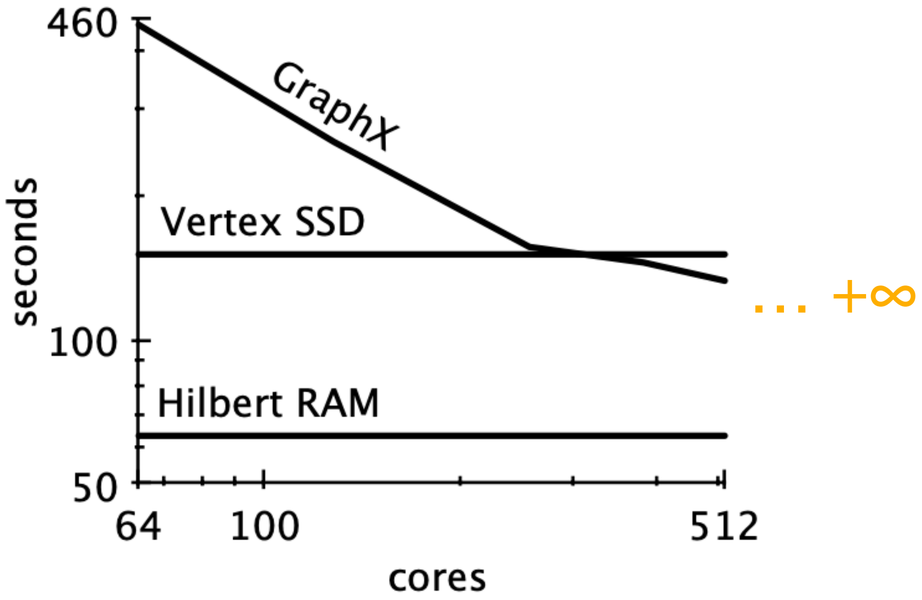
Warm & cold benchmarking

PageRank

Cost per warm iteration



Cold start



… +∞

(Reported data only)

Observations

And lessons

- Many distributed systems slower than single-threaded implementations
- Scalability ≠ efficient use of resources
- Models (including metrics) constrain your thinking — in an **un/helpful** way?

Question your models!

Ask the right questions.

Thank you!

Questions?