Distributed Computing

17. Scalability! But at What COST?

An Intervention

Scalability! But at what COST?

Frank McSherry
Unaffiliated

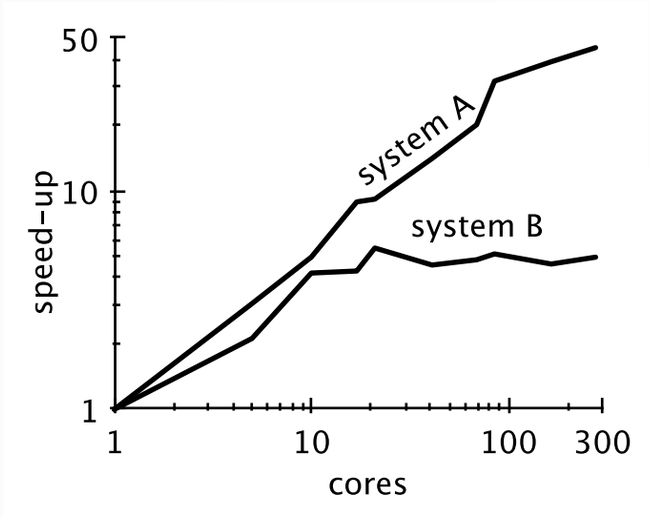
Michael Isard
Unaffiliated*

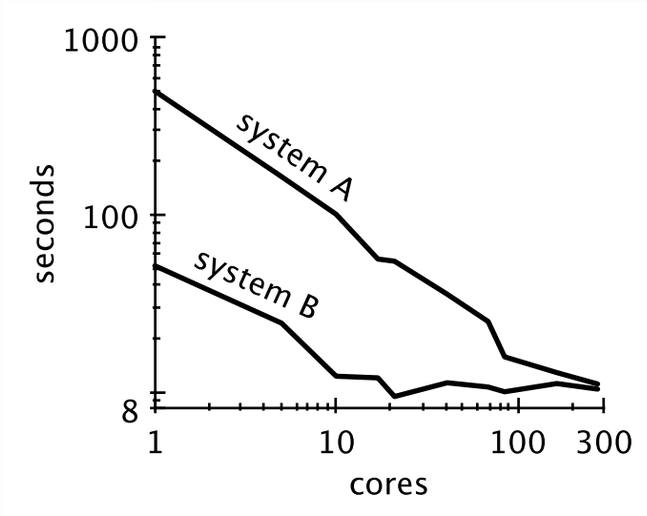
Derek G. Murray
Unaffiliated†

- Based on the paper by McSherry et al. at USENIX HotOS 2015
- Please also watch the video on YouTube recorded in 2017

Scalability vs. Performance

Scalability vs. Performance





- Speed-up: how much faster the system becomes when you add resources
- Performance: how fast the system is—period! :)

Comparing PageRank

scalable system	cores	twitter	uk-2007-05
GraphChi [12]	2	3160s	6972s
Stratosphere [8]	16	2250s	-
X-Stream [21]	16	1488s	-
Spark [10]	128	857s	1759s
Giraph [10]	128	596s	1235s
GraphLab [10]	128	249s	833s
GraphX [10]	128	419s	462s
Single thread (SSD)	1	300s	651s
Single thread (RAM)	1	275s	-

Single-Thread PageRank in Rust

```
fn PageRank20(graph: GraphIterator, alpha: f32) {
  let mut a = vec![0f32; graph.nodes()];
  let mut b = vec![0f32; graph.nodes()];
  let mut d = vec![0f32; graph.nodes()];

  graph.map_edges(|x, y| { d[x] += 1; });

  for iter in 0..20 {
    for i in 0..graph.nodes() {
      b[i] = alpha * a[i] / d[i];
      a[i] = 1f32 - alpha;
    }

    graph.map_edges(|x, y| { a[y] += b[x]; });
  }
}
```

Fancy Optimization: Hilbert Curve

- Not happy with this, authors looked for a further humiliation
- Additional fancy optimization: re-sort edges using a way that gets more locality in memory access
- Gets an additional ~2x speedup

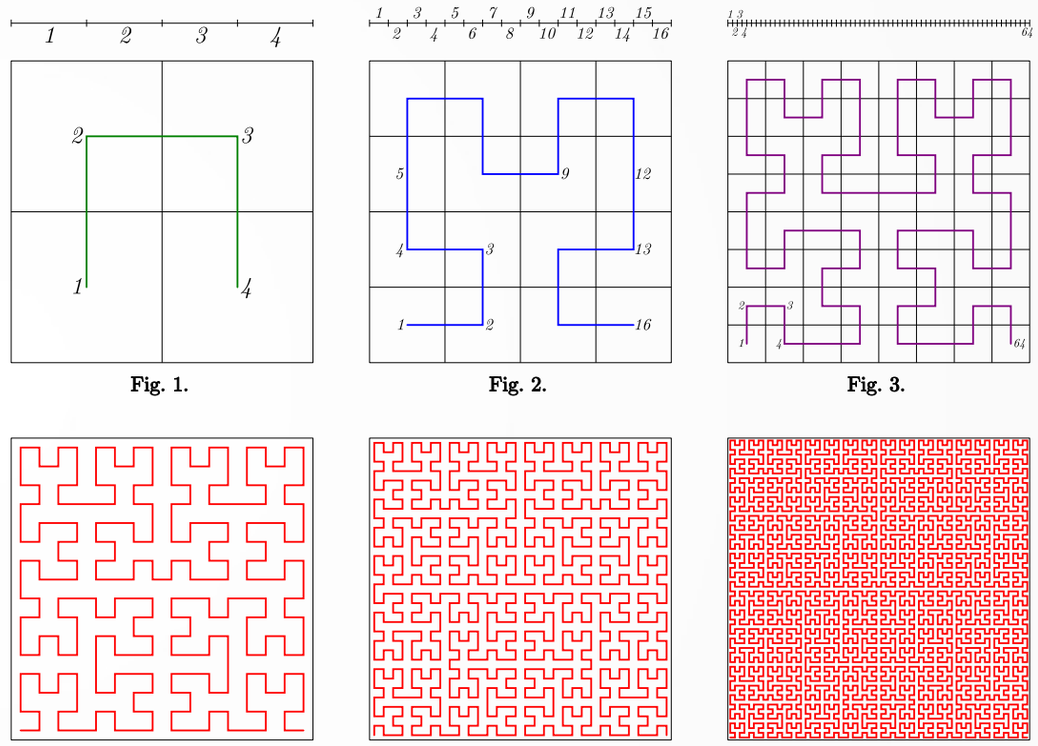
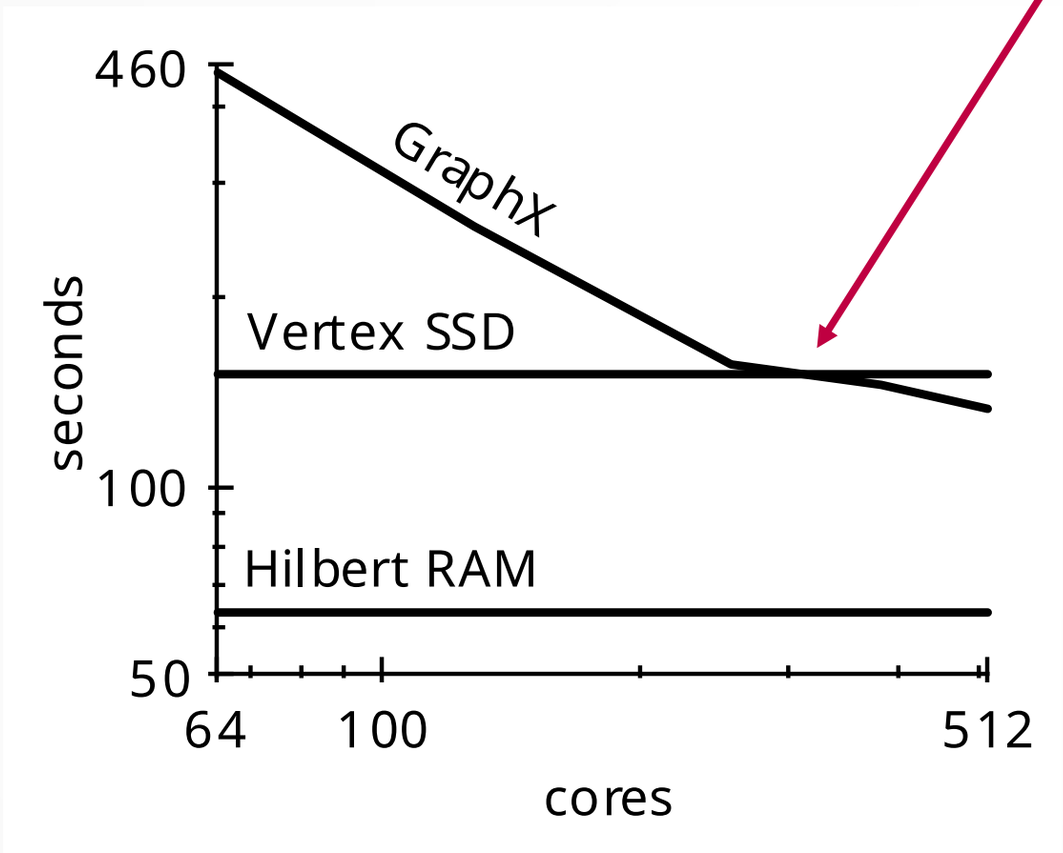
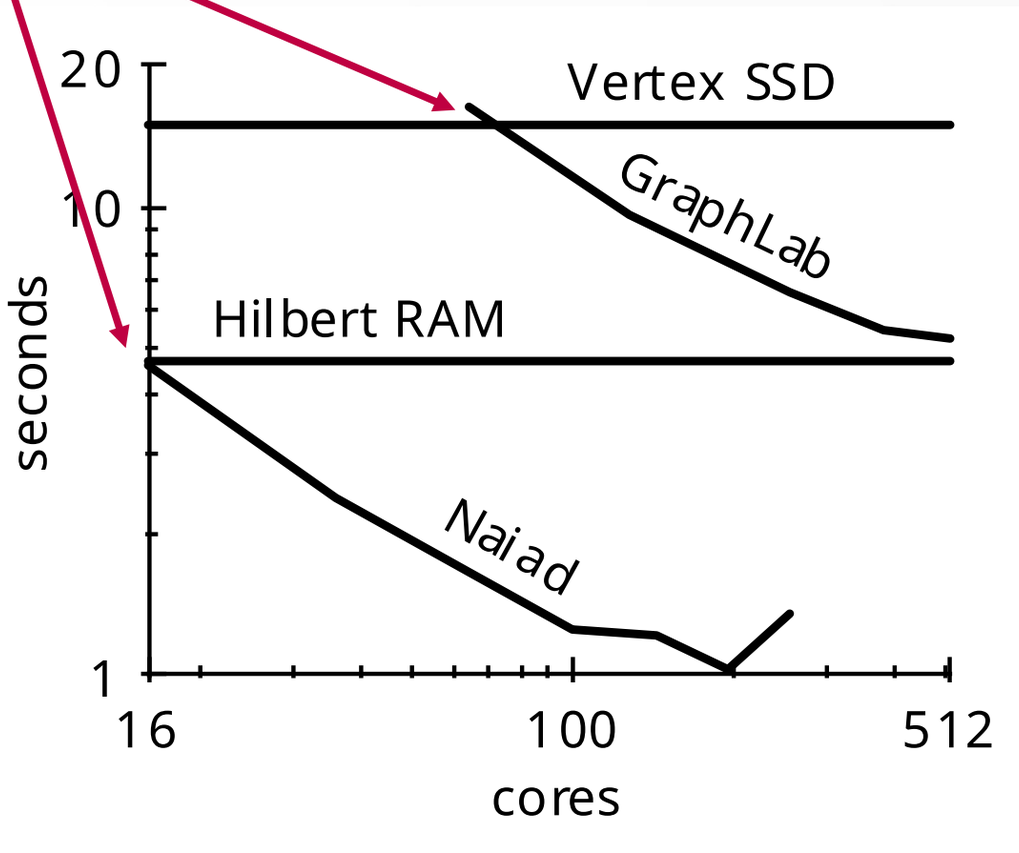


Image by user Braindrain0000 on Wikipedia. CC-BY-SA 3.0 license.

The Overall Results

COST: Configuration that Outperforms a Single Thread





Conclusions

- Systems have got better and they keep getting better
- These are applications that require a lot of CPU and information linking
  - Really a bad fit for distributed systems
  - We can’t tell the same story when **disks are the bottleneck**
  - Also, Java overheads—in particular, for serialization (converting objects to string)
- Don’t be mindless when designing the system and **think about your bottlenecks**