Scalability! But at what COST?

Frank McSherry, Michael Isard, and Derek Murray

Presentation BY: Abdullah Alomar

Introduction

Motivation

"You can have a second computer once you've shown you know how to use the first one."

— Paul Barham

The authors observe that many published works on big data systems are evaluated in terms of their **scalability**.

Only **few** directly compare their performance in terms of absolute performance (compute time) against reasonable benchmarks.

The problem is that scalability is a rather misleading metric; any system can scale arbitrarily well when it is not implemented well .

Example of misleading scalability

The problem is that scalability is a rather misleading metric; any system can scale arbitrarily well when it is not implemented well .

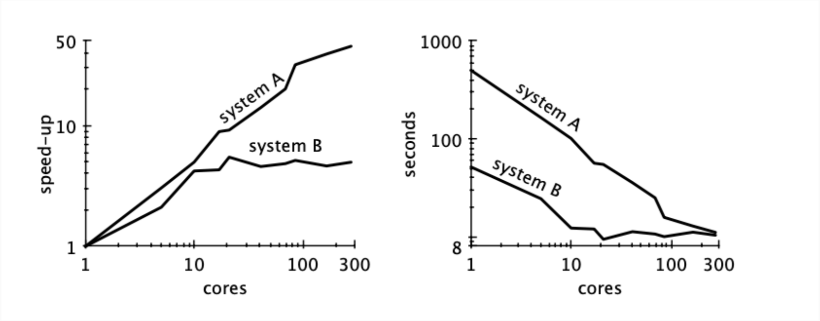


Figure 1: Scaling and performance measurements for a data-parallel algorithm, before (system A) and after (system B) a simple performance optimization.

A New Metric

Motivated by this, the authors present a new metric for big data platforms called COST (Configuration that Outperforms a Single Thread).

Specifically, the COST of a given platform is the hardware configuration (Number of cores) required to outperforms a competent single-threaded implementation. Example:

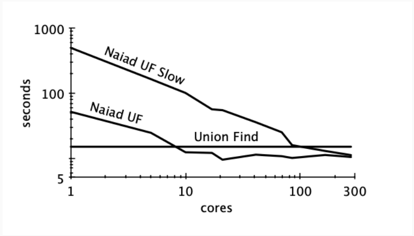


Figure 2: Naiad UF has a COST of 10 cores, while the slow implementation has a COST of roughly 100 cores.

Comparison With Single Threaded Programs

Single Threaded programs

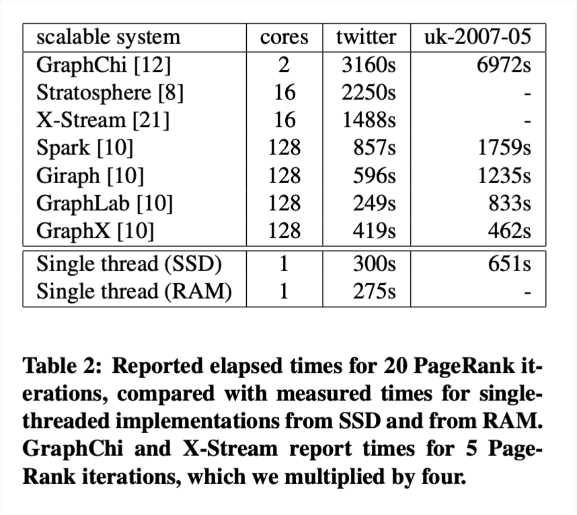
To get their point across, the author’s compare a single-threaded program with several scalable graph processing platforms.

Specifically, they carry out two tasks:

1. PageRank
2. Connected Components

on two large graphs: a Twitter graph and the web graph “uk-2007-05”.

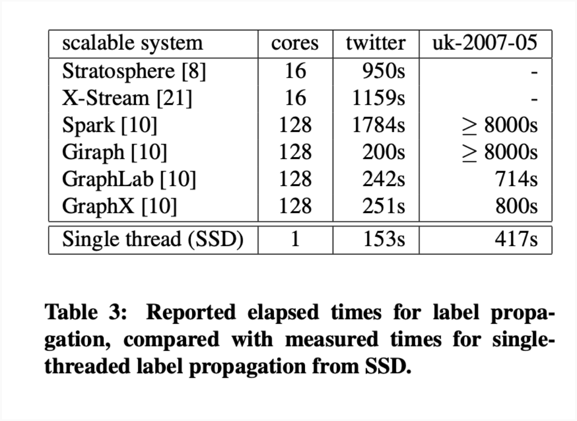
Page Rank Comparison



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	-

Table 2: Reported elapsed times for 20 PageRank iterations, compared with measured times for single-threaded implementations from SSD and from RAM. GraphChi and X-Stream report times for 5 PageRank iterations, which we multiplied by four.

Connected Components Comparison



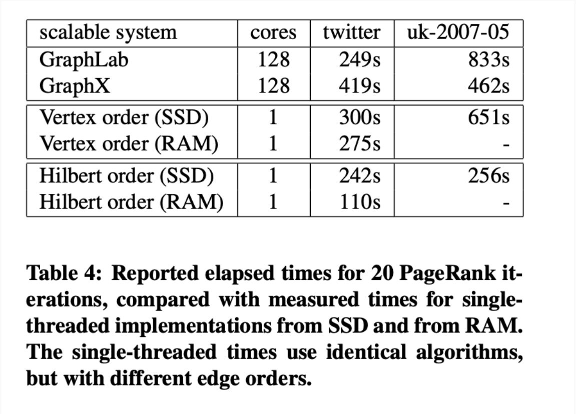
scalable system	cores	twitter	uk-2007-05
Stratosphere [8]	16	950s	-
X-Stream [21]	16	1159s	-
Spark [10]	128	1784s	$\geq$ 8000s
Giraph [10]	128	200s	$\geq$ 8000s
GraphLab [10]	128	242s	714s
GraphX [10]	128	251s	800s
Single thread (SSD)	1	153s	417s

Table 3: Reported elapsed times for label propagation, compared with measured times for single-threaded label propagation from SSD.

Using A better baseline

The authors also compare with a more efficient single-threaded implementations. They find that these simple optimizations yields a performance that surpass the performance of the systems considered by an additional order of magnitude.

1. PageRank: Improving graph layout



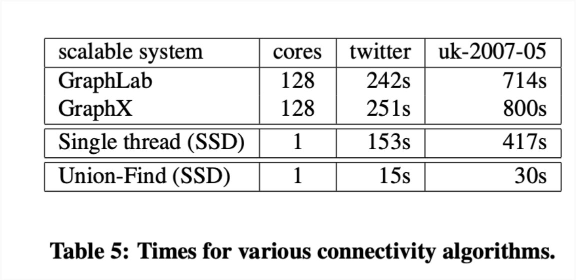
scalable system	cores	twitter	uk-2007-05
GraphLab	128	249s	833s
GraphX	128	419s	462s
Vertex order (SSD)	1	300s	651s
Vertex order (RAM)	1	275s	-
Hilbert order (SSD)	1	242s	256s
Hilbert order (RAM)	1	110s	-

Table 4: Reported elapsed times for 20 PageRank iterations, compared with measured times for single-threaded implementations from SSD and from RAM. The single-threaded times use identical algorithms, but with different edge orders.

Using A better baseline

2. **Connected Components:** Use Union-Find instead of label propagation.

The label propagation algorithm is used for graph connectivity because it fits within the "think like a vertex" computational model. This algorithm scales well due to the algorithm's sub-optimality; as it does more work than better algorithms.



scalable system	cores	twitter	uk-2007-05
GraphLab	128	242s	714s
GraphX	128	251s	800s
Single thread (SSD)	1	153s	417s
Union-Find (SSD)	1	15s	30s

Table 5: Times for various connectivity algorithms.

COST of Prior Systems

COST of several platforms

These curves which shows the run time of PageRank for the twitter graph. From these two curves we can say:

- **Naiad has a COST of 16 cores**.
- **GraphLab has a COST of 512 cores** (Not shown in figure).
- **GraphX does not intersect the corresponding single-threaded measurement, and thus has unbounded COST**.

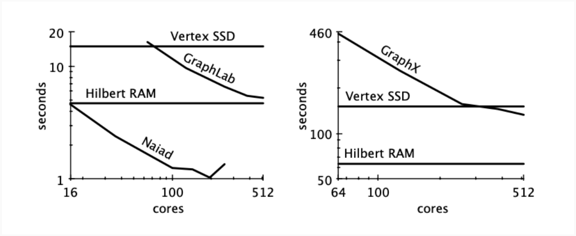


Figure 5: Published scaling measurements for PageRank on twitter graph. The first plot is the time per warm iteration. The second plot is the time for ten iterations from a cold start. Horizontal lines are single threaded measurements.

Conclusions and Discussion

Conclusions and Discussion

- Several aspects of scalable systems design contribute to overheads and increased the absolute running time:
    1. The computational model might restricts the programs one may express (e.g. vertex-centric model only allow for a limited subset of efficient graph algorithms).
    2. The implementation of the system may add overheads a single thread doesn't require.
- Given these reasons, more care should be exercised when evaluating a scalable system. The COST of a system, for example, can yield good insight about the systems performance.

Discussion:

1. Has this paper triggered any changes in how these large-scale systems are evaluated? If not, how do you think this problem should be tackled?
2. How did we get here? Why has the focus shifted to answering the question “Can systems be made to scale well?” (which is as this paper shows is trivially answered by designing less efficient implementations).

Thank You