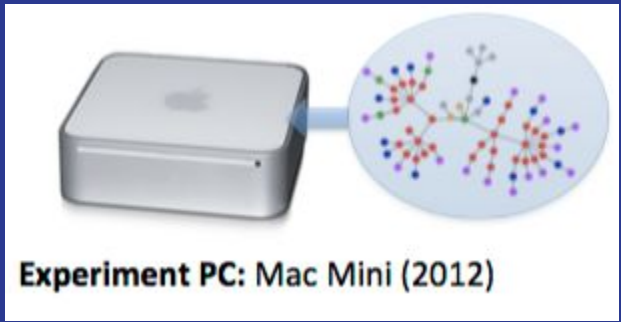
GraphChi: Large-Scale Graph Computation on Just a PC

Kyrola Et al.

James Trever

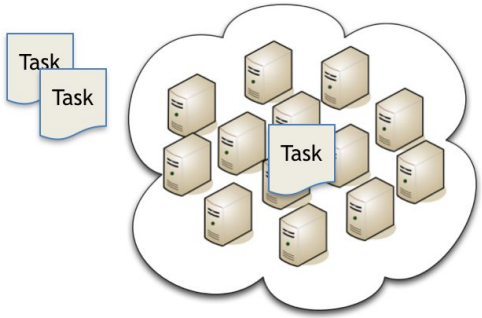
Could we compute Big Graphs on a single machine?

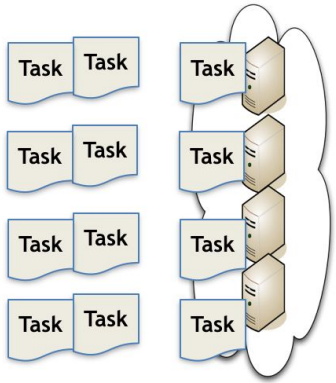
Disk Based Computation



Why would you want to?

- Distributed State is hard to program
  - Cluster crashes can occur
  - Cumbersome
- Efficient Scaling
  - Parallelise each task vs Parallelise across tasks
- Cost
  - Easier management and simpler hardware
- Energy Consumption
  - Full utilisation of a single computer
- Easier Debugging

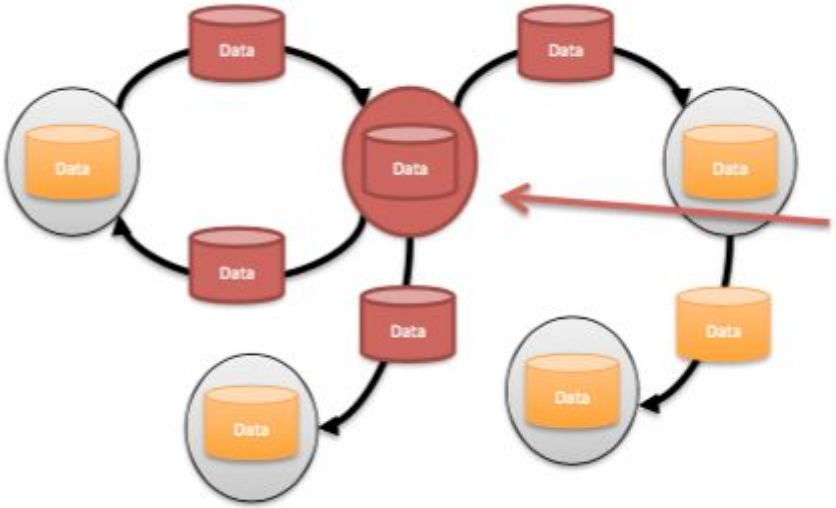




Contents

Computational Model

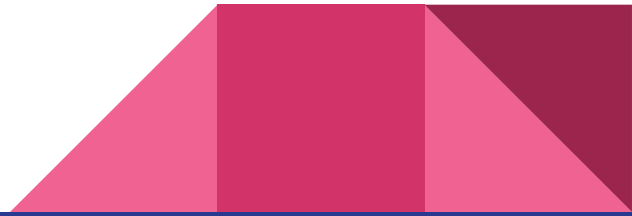
Computational Model



MyFunc(vertex)
{ // modify neighborhood }

Storage Model

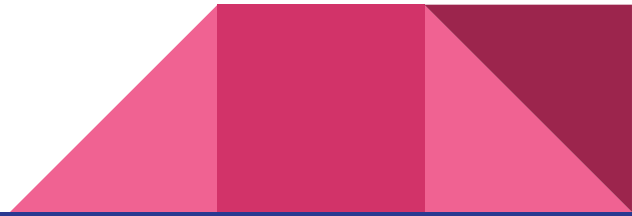
- Compressed Sparse Row (CSR) - allows for fast loading of out-edges
- Compressed Sparse Column (CSC) - allows for fast loading of in-edges



Storage Model

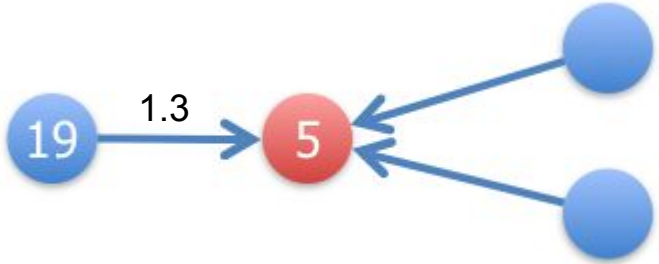
- Compressed Sparse Row (CSR) - allows for fast loading of out-edges
- Compressed Sparse Column (CSC) - allows for fast loading of in-edges

Why not both?



Challenges

Random Access Problem



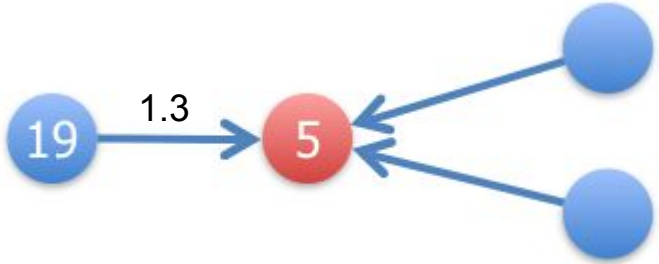
- Symmetrised adjacency file with values

vertex	in-neighbors	out-neighbors
5	**3**:2.3, 19: 1.3, **49**: 0.65,...	**781**: 2.3, **881**: 4.2..
....		
19	**3**: 1.4, **9**: 12.1, ...	5: 1.3, 28: 2.2, ...

synchronise

Random write

Random Access Problem



- File Index Pointers

vertex	in-neighbor-ptr	out-neighbors
5	**3**: 881, **19**: 10092, **49**: 20763,...	**781**: 2.3, **881**: 4.2..
....		
19	**3**: 882, **9**: 2872, ...	5: 1.3, 28: 2.2, ...

read

Random read

Possible Solutions

1. Use SSD as memory extension
    - Too many small objects, need millions of reads and writes a second
2. Compress the graph structure to fit in RAM
    - Associated values do not compress well
3. Cachine the hot vertices
    - Unpredictable Performance

Parallel Sliding Windows (PSW)

PSW: Phases

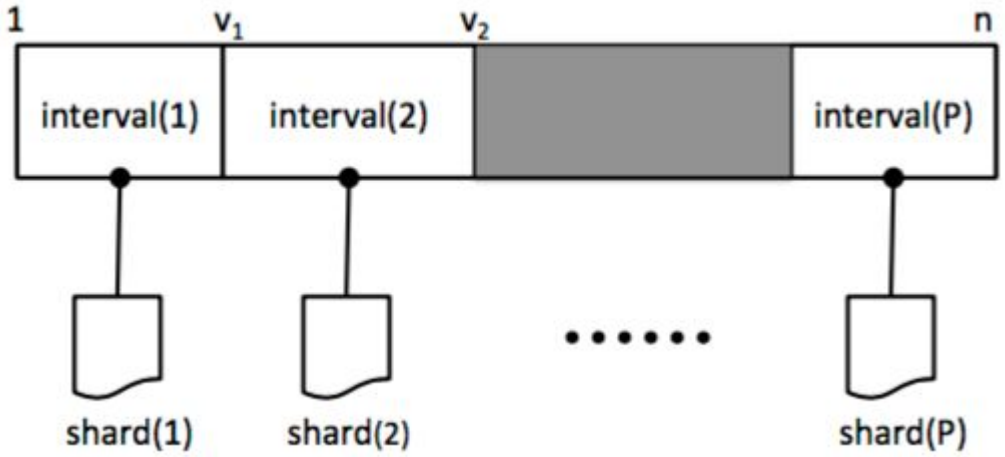
PSW processes the graph one sub-graph at a time

1. Load
2. Compute
3. Write

In one iteration the whole graph is processed

PSW: Intervals and Shards - Load

- Subgraph = Interval



PSW: Example - Load

Shard 1

src	dst	value
1		
	2	0.3
3		
	2	0.2
4		
	1	1.4
5		
	1	0.5
	2	0.6
6		
	2	0.8

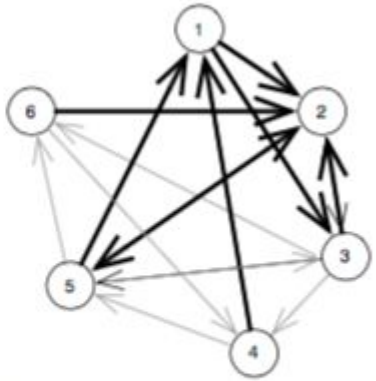
Shard 2

src	dst	value
1		
	3	0.4
2		
	3	0.3
3		
	4	0.8
5		
	3	0.2
6		
	4	1.9

Shard 3

src	dst	value
2		
	5	0.6
3		
	5	0.9
	6	1.2
4		
	5	0.3
5		
	6	1.1

(a) Execution interval (vertices 1-2)



(b) Execution interval (vertices 1-2)

PSW: Example - Load

Shard 1

src	dst	value
1		
	2	0.273
3		
	2	0.22
4		
	1	1.54
5		
	1	0.55
	2	0.66
6		
	2	0.88

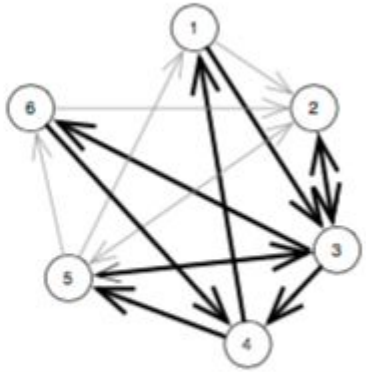
Shard 2

src	dst	value
1		
	3	0.364
2		
	3	0.273
3		
	4	0.8
5		
	3	0.2
6		
	4	1.9

Shard 3

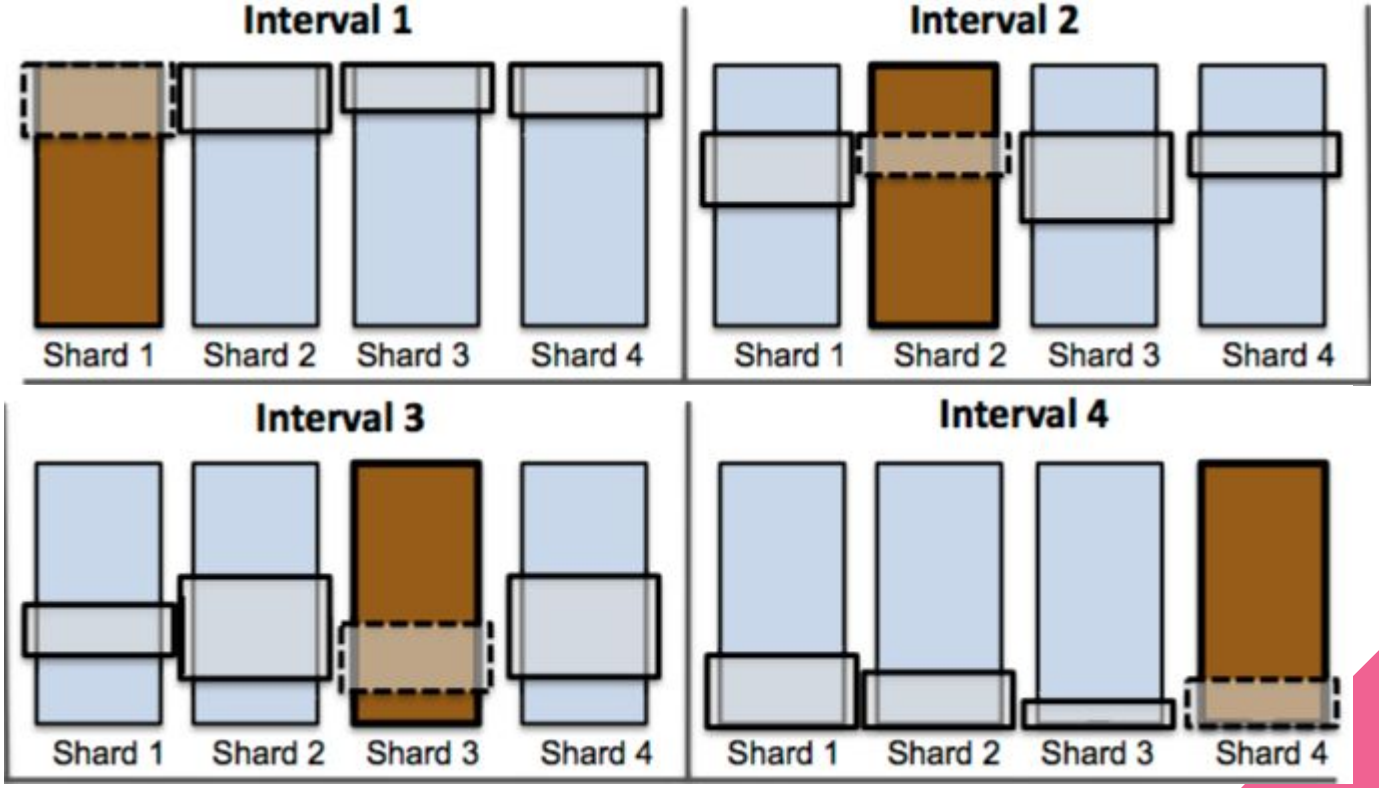
src	dst	value
2		
	5	0.545
3		
	5	0.9
	6	1.2
4		
	5	0.3
5		
	6	1.1

(c) Execution interval (vertices 3-4)



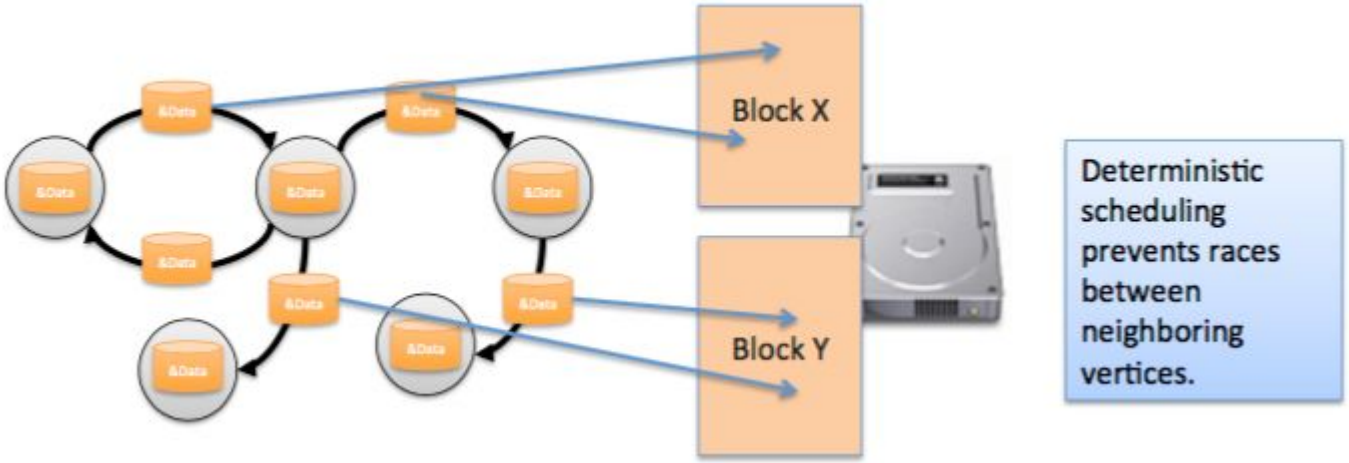
(d) Execution interval (vertices 3-4)

PSW: General Example - Load



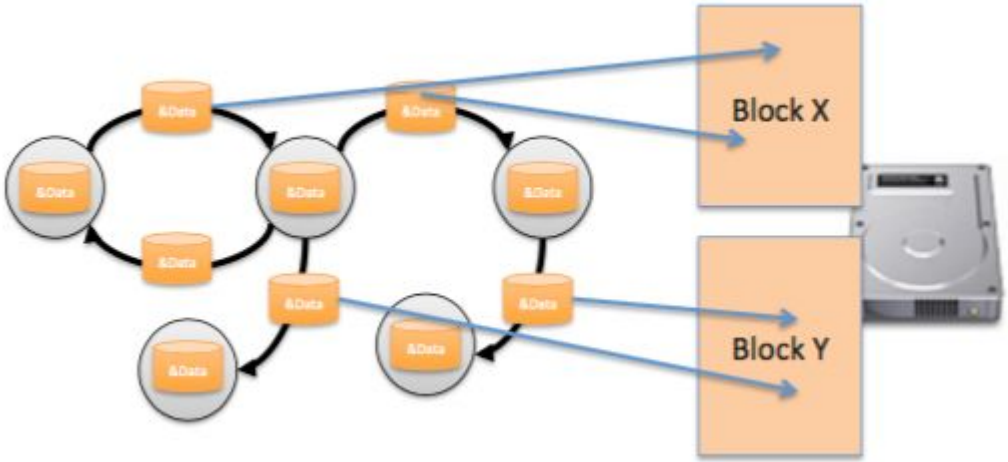
PSW: Compute Phase

- UpdateFunction executes on intervals vertices in parallel
- Edges have pointers to the loaded data blocks



PSW: Write Phase

- Blocks are written back to disk asynchronously

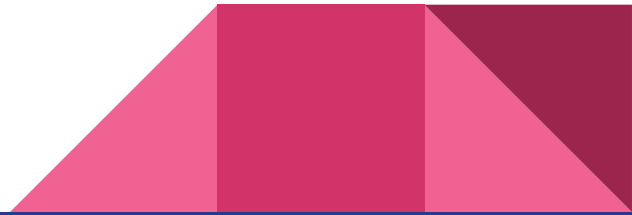


Implementation and Experiments

Preprocessing Step

- Sharder program included with GraphChi

1. Counts the in-degree of each vertex and computes the prefix sum over the degree array so that each interval contains same number of in edges
2. Sharder writes each edge to temporary scratch file belonging to the shard
3. Sharder Processes each scratch file
4. Sharder computes binary degree file containing in and out degree for each vertex (used to calculate memory requirements)



Preprocessing Experiment

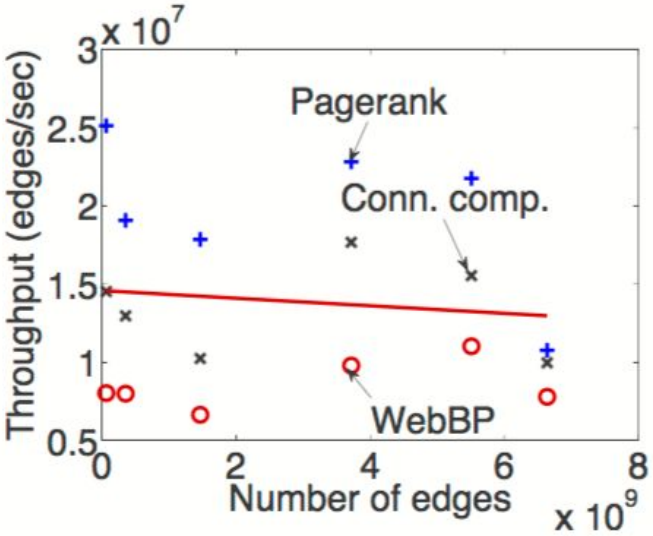
Graph name	Vertices	Edges	P	Preproc.
live-journal [3]	4.8M	69M	3	0.5 min
netflix [6]	0.5M	99M	20	1 min
domain [44]	26M	0.37B	20	2 min
twitter-2010 [26]	42M	1.5B	20	10 min
uk-2007-05 [11]	106M	3.7B	40	31 min
uk-union [11]	133M	5.4B	50	33 min
yahoo-web [44]	1.4B	6.6B	50	37 min

Comparison Experiment

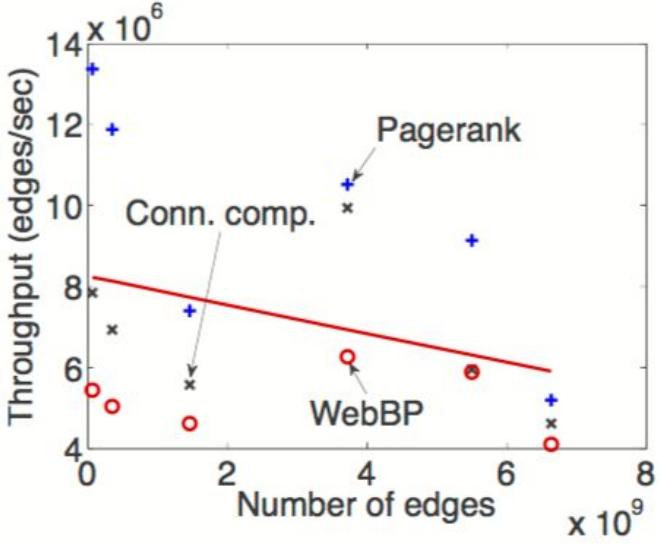
Application & Graph	Iter.	Comparative result	GraphChi (Mac Mini)	Ref
Pagerank & domain	3	GraphLab[30] on AMD server (8 CPUs) **87 s**	**132 s**	-
Pagerank & twitter-2010	5	Spark [45] with 50 nodes (100 CPUs): **486.6 s**	**790 s**	[38]
Pagerank & V=105M, E=3.7B	100	Stanford GPS, 30 EC2 nodes (60 virt. cores), **144 min**	approx. **581 min**	[37]
Pagerank & V=1.0B, E=18.5B	1	Piccolo, 100 EC2 instances (200 cores) **70 s**	approx. **26 min**	[36]
Webgraph-BP & yahoo-web	1	Pegasus (Hadoop) on 100 machines: **22 min**	**27 min**	[22]
ALS & netflix-mm, D=20	10	GraphLab on AMD server: **4.7 min**	**9.8 min** (in-mem) **40 min** (edge-repl.)	[30]
Triangle-count & twitter-2010	-	Hadoop, 1636 nodes: **423 min**	**60 min**	[39]
Pagerank & twitter-2010	1	PowerGraph, 64 x 8 cores: **3.6 s**	**158 s**	[20]
Triange-count & twitter- 2010	-	PowerGraph, 64 x 8 cores: **1.5 min**	**60 min**	[20]

Mac Mini Dual Core 2.5 GHz, 8GB Ram
AMD Server 8 core server with 4 dual core CPU’s

Throughput Experiment



(a) Performance: SSD

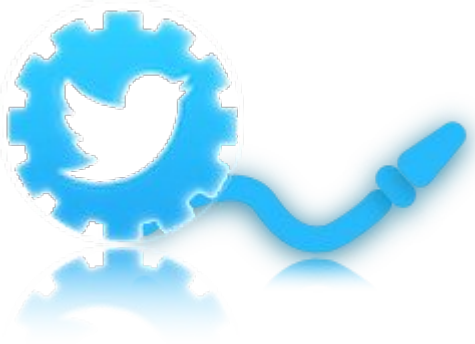


(b) Performance : Hard drive

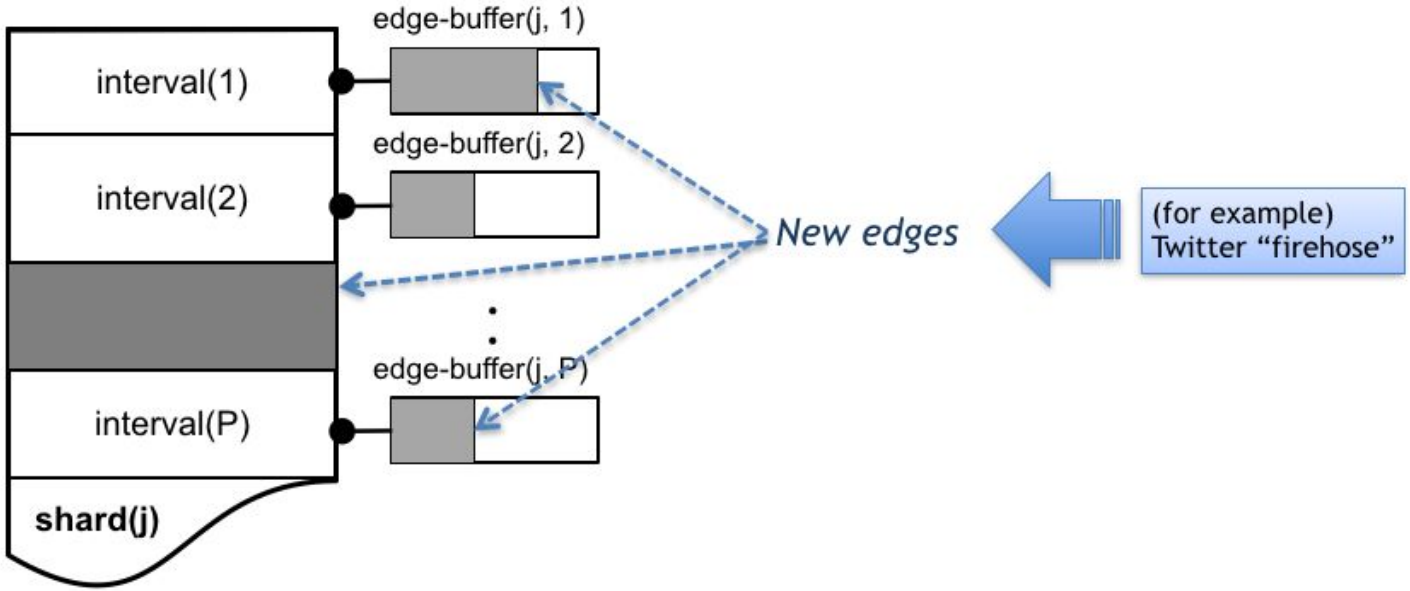
Evolving Graphs

Evolving Graphs

- Add and remove edges in streaming fashion whilst continuing computation
- Most interesting networks grow continuously



PSW and Evolving Graphs



PSW and Evolving Graphs

Shard 1

src	dst	value
1		
	2	0.3
3		
	2	0.2
4		
	1	1.4
5		
	1	0.5
	2	0.6
6		
	2	0.8

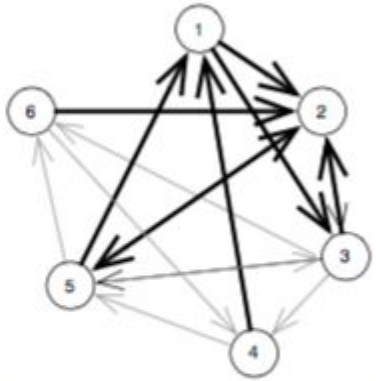
Shard 2

src	dst	value
1		
	3	0.4
2		
	3	0.3
3		
	4	0.8
5		
	3	0.2
6		
	4	1.9

Shard 3

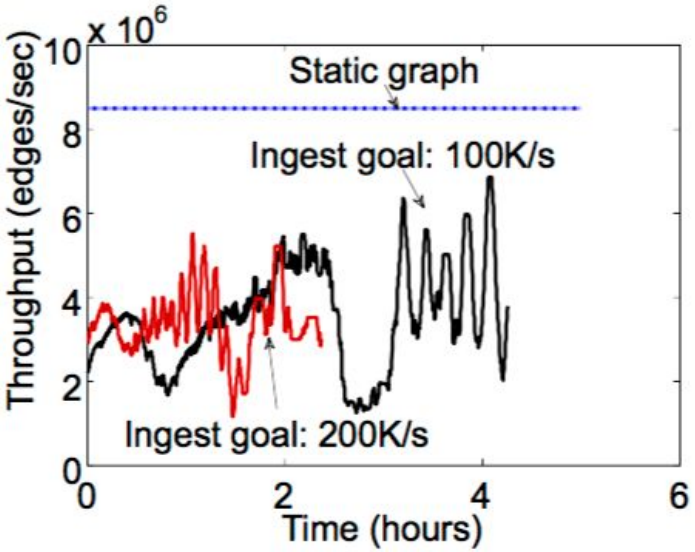
src	dst	value
2		
	5	0.6
3		
	5	0.9
	6	1.2
4		
	5	0.3
5		
	6	1.1

(a) Execution interval (vertices 1-2)

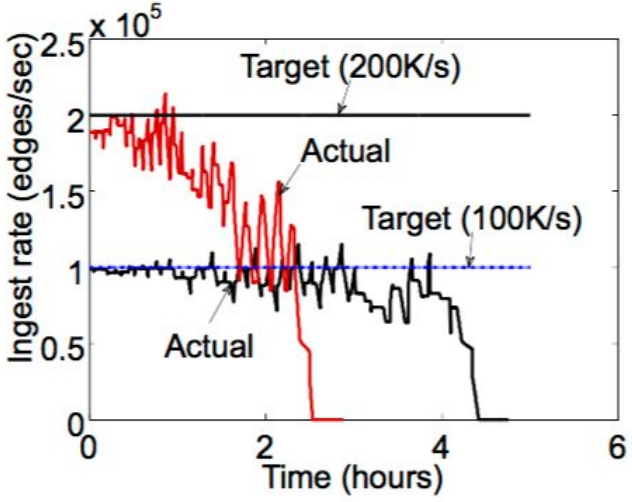


(b) Execution interval (vertices 1-2)

Evolving Graphs - Experiment



(a) Evolving Graph: Throughput



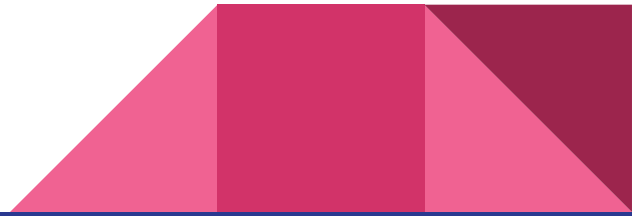
(b) Ingest rate

Graphs Used

Graph name	Vertices	Edges	P	Preproc.
live-journal [3]	4.8M	69M	3	0.5 min
netflix [6]	0.5M	99M	20	1 min
domain [44]	26M	0.37B	20	2 min
twitter-2010 [26]	42M	1.5B	20	10 min
uk-2007-05 [11]	106M	3.7B	40	31 min
uk-union [11]	133M	5.4B	50	33 min
yahoo-web [44]	1.4B	6.6B	50	37 min

Critical Evaluation

- Few mistakes in the paper referencing incorrect tables or quoting wrong figures
- Cannot efficiently support dynamic ordering like priority ordering or efficiently support graph traversals or vertex queries
- Evolving graph experiments not very clear
- No monetary analysis



Bibliography

A. Kyrola, G. Blelloch, and C. Guestrin, “Graphchi: Large-scale graph computation on just a pc,” in Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, OSDI’12, (Berkeley, CA, USA), pp. 31–46, USENIX Association, 2012.

And his original presentation found here:

https://www.usenix.org/sites/default/files/conference/protected-files/kyrola_osdi12_slides.pdf