15-min Review of:

GraphChi

Large-Scale Graph Computation on Just a PC

Presented by Niko Stahl for R212

Context

- **Familiar problem**:

How to process a graph that does not fit into main memory?

- **Previous solutions:**

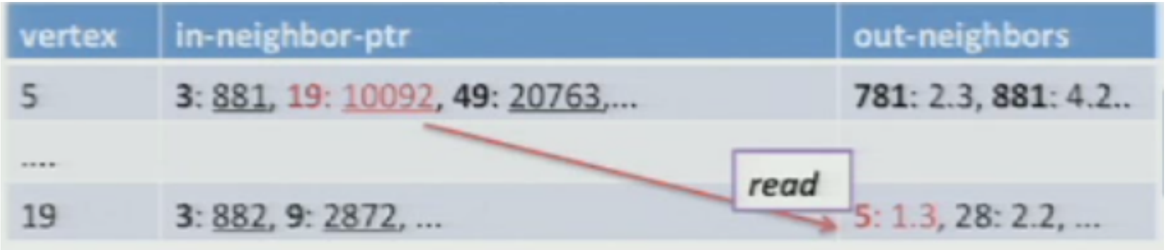
Distribute: Split data across machines. These approaches have to deal with fault tolerance and can be difficult to debug.

GraphChi’s solution

Efficiently use disks (HDD or SDD) of a single machine.

The key problem with using disks:

Random Access is **slow** for disk-based computation. And naive implementations of graph algorithms require frequent random access.



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, ...

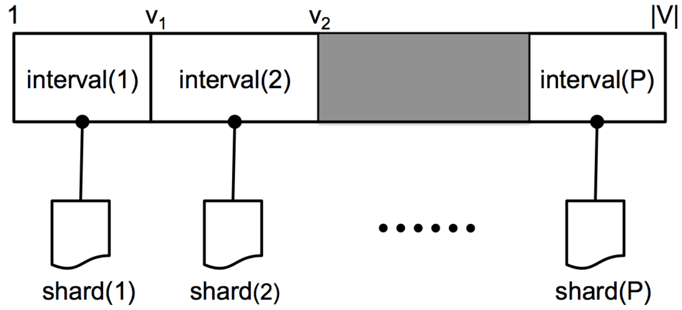
The Model - Overview

Parallel Sliding Window (PSW) algorithm:

For each vertex v, optimize processing of the subgraph containing v and its neighbours (predecessor and successors) by **maximizing sequential memory access**.

The Model - Primitives

- Each vertex is mapped to an **interval**.
- Each interval stored on a **shard,** which can be loaded completely into memory (~ order of 10^2 MB).
- Edges mapped to intervals based on their target vertex.



The Model - An Example

Memory shards — Sliding shards

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

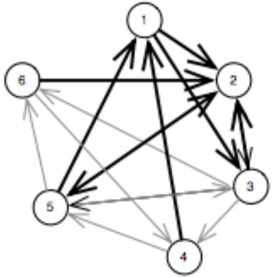
Shard 2

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)

- Vertices are accessed one shard at a time.
- For each shard s, the edges containing vertices on s must be accessed to construct the subgraph.
  - in-edges: already in interval i
  - out-edges: **ordered by source** on sliding shards. Therefore, they can be accessed **sequentially**.

The Model - An Example

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

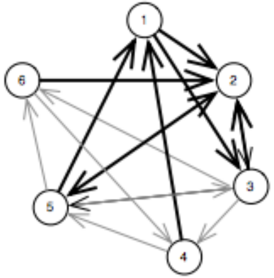
Shard 2

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)

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

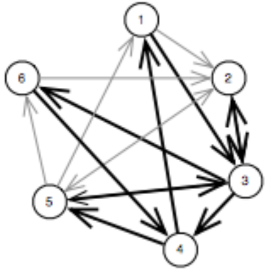
Shard 2

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)



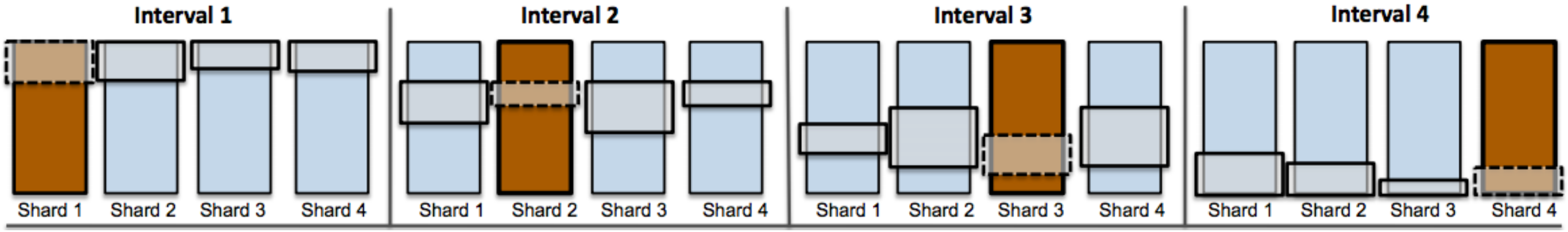
(d) Execution interval (vertices 3-4)

Crux: Within each sliding shard access is sequential

The Model - Complexity

Let P be the number of intervals. PSW requires P sequential disk accesses to process each interval.

$O(P^2)$ reads/writes at each iteration ($P<1,000$).



PSW Properties

Pros

+ Asynchronous model (not BSP). More efficient because vertices can be processed in any order (this can be useful for some algorithms, e.g. Dijkstra’s shortest path).

+ Extends well to evolving graphs

+ Easier to debug because computation runs on a single machine.

Cons

- Standard vertex queries are not efficient (building neighborhood of a vertex require scan of a memory shard)

Evaluation

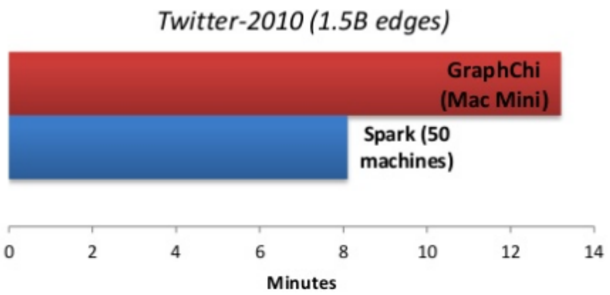
Mac Mini

- 8 GB RAM
- 256 GB SSD
- 1 TB HDD
- 2.5 GHz

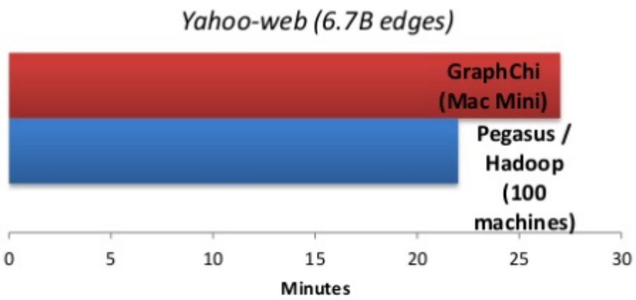
Graph name	Vertices	Edges
live-journal [3]	4.8M	69M
netflix [6]	0.5M	99M
domain [47]	26M	0.37B
twitter-2010 [28]	42M	1.5B
uk-2007-05 [12]	106M	3.7B
uk-union [12]	133M	5.4B
yahoo-web [47]	1.4B	6.6B

Evaluation

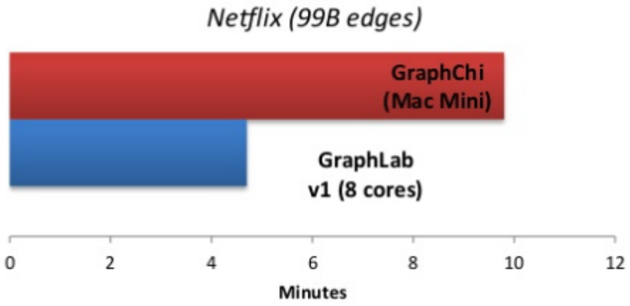
PageRank



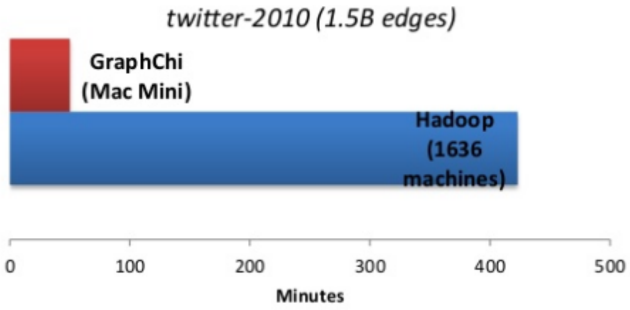
WebGraph Belief Propagation (U Kang et al.)



Matrix Factorization (Alt. Least Sqr.)



Triangle Counting



Comparing with GraphLab 2

GraphLab with 512 CPUs (64 machines).

Computation on Twitter Graph:

- **PageRank**: 40x faster than GraphChi
- **Triangle Counting**: 30x faster than GraphChi
- Note: In terms of CPU, GraphLab requires 256x the resources

Conclusion & Personal View

- Good for prototyping. Is support for evolving graphs necessary?
- The evaluation does not include setup time.
- The model is intuitive. But it is not clear how cumbersome the programming interface is.
- GraphChi introduces an alternative perspective on large scale graph computation. It relies on algorithms and data structures instead of greater resources.