X-Stream: Edge-centric Graph Processing using Streaming Partitions

Amitabha Roy, Ivo Mihailovic, Willy Zwaenepoel (SOSP'13)

Presented by: Stella Lau

24 October 2017

Motivation: scalable graph processing

Problem

Performance of large scale graph processing
$\Rightarrow$ Lack of access locality

Motivation: scalable graph processing

Problem

Performance of large scale graph processing
$\Rightarrow$ Lack of access locality

Solution?

Large clusters (e.g. Pregel, Giraph, GraphLab)
$\Rightarrow$ Increased complexity and power consumption

X-Stream: contributions

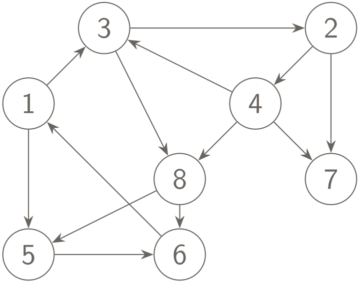
A system for scale-up graph processing for both in-memory and out-of-core graphs on a *single, shared-memory machine*, using

1. an *edge-centric* scatter gather model
2. streaming partitions

Context: scatter-gather model (Pregel, PowerGraph, etc.)

- Store state in vertices
- Vertex operations:
  - *Scatter* updates over outgoing edges of vertex
  - *Gather* updates from inbound edges of vertex

Vertex-centric scatter gather: BFS

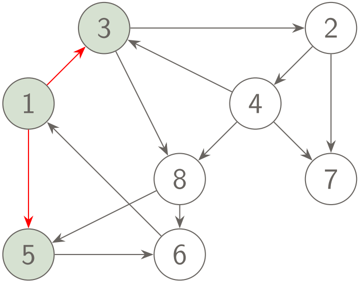


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Vertex-centric scatter gather: BFS

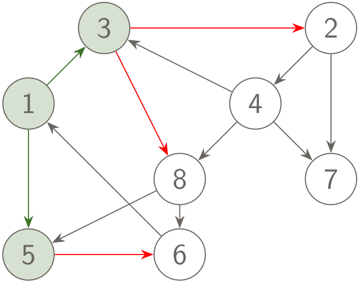


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Vertex-centric scatter gather: BFS

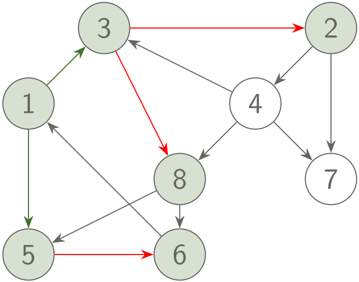


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Vertex-centric scatter gather: BFS

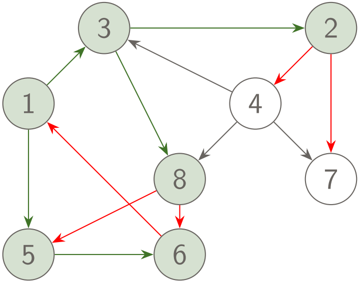


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Vertex-centric scatter gather: BFS

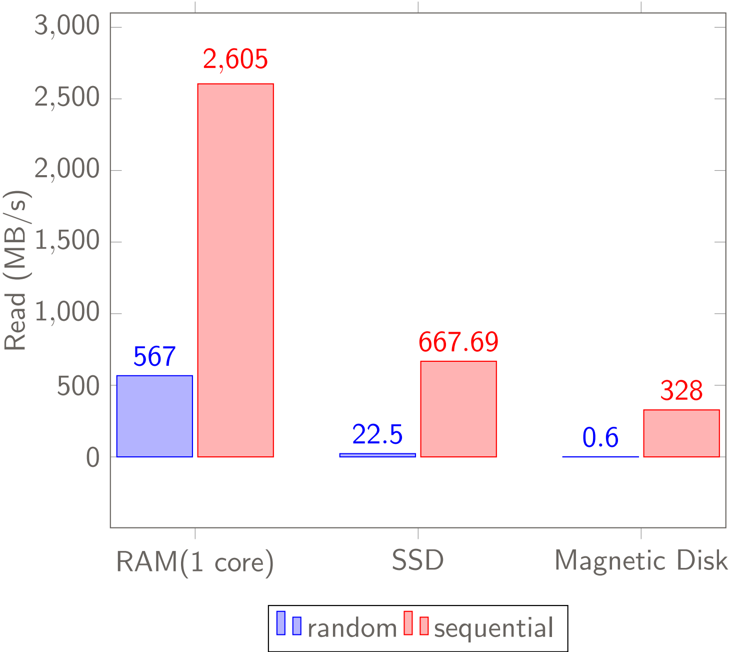


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Problem: random access vs sequential access



Solution: edge-centric scatter-gather

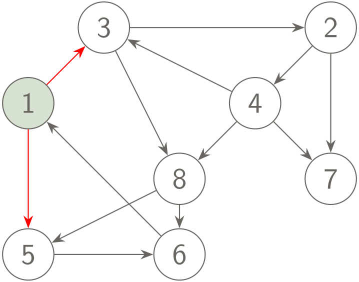
Vertex-centric

```
for each vertex v
  if v has update
    for each edge e from v
      scatter update along e
```

Edge-centric

```
for each edge e
  if e.src has update
    scatter update along e
```

Edge-centric scatter gather: BFS

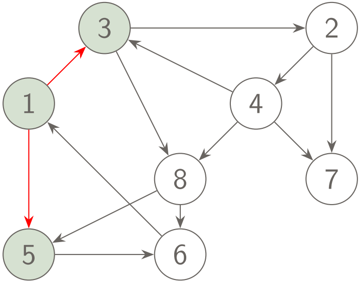


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Edge-centric scatter gather: BFS

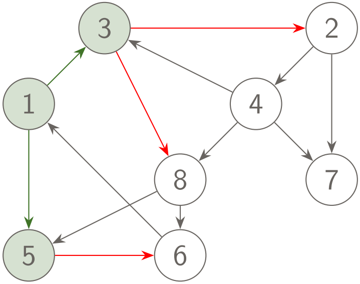


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Edge-centric scatter gather: BFS

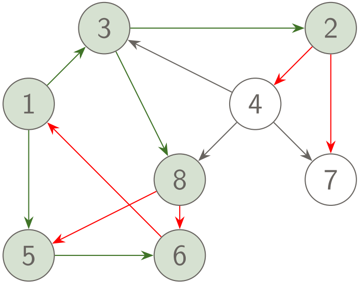


v
1
2
3
4
5
6
7
8

src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Edge-centric scatter gather: BFS



v
1
2
3
4
5
6
7
8

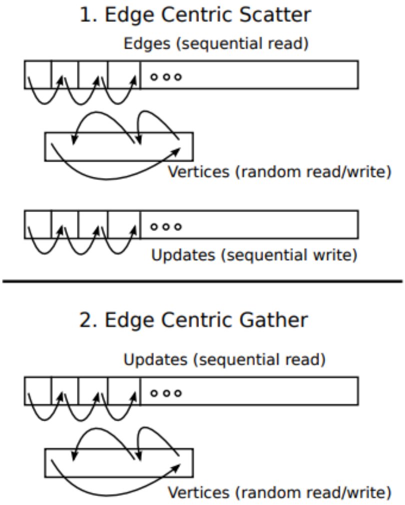
src	dest
1	3
1	5
2	7
2	4
3	2
3	8
4	3
4	7
4	8
5	6
6	1
8	5
8	6

Example from SOSP'13 talk by Amitabha Roy

Gains from edge-centric model

- Edge table does not need to be sorted
- No index table
- Vertex-centric scatter-gather: $\frac{EdgeData}{RandomAccessBandwidth}$
- Edge-centric scatter-gather: $\frac{Scatters \times EdgeData}{SequentialAccessBandwidth}$
- Sequential access bandwidth $\gg$ random access bandwidth

Problem: random access to vertices



Solution

- Store vertices in fast storage
  - In-memory: caches vs main-memory
  - Out-of-core: main-memory vs SSD/Disk
- What if they don't fit?
  - Streaming partitions

Streaming partitions

1. Vertex set V: subset of vertices that fits in fast storage
2. Edge set: source $\in V$
3. Update list: dest $\in V$

Example partition

v1
1
2
3
4

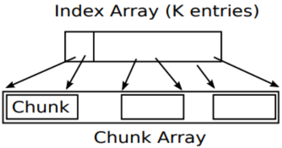
src	dest
2	4
1	3
4	8
4	3
3	2
2	7
3	8
4	7
1	5

---

v2
5
6
7
8

src	dest
8	5
6	1
8	6
5	6

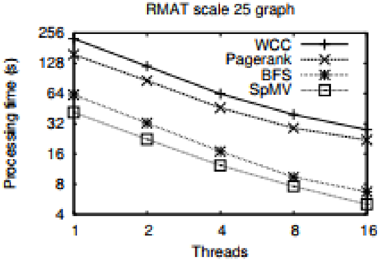
Implementation

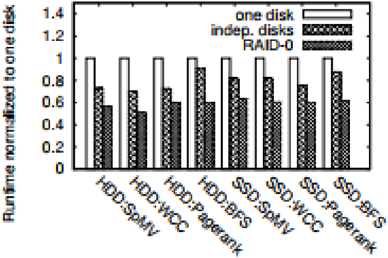


- Scatter/gather over streaming partitions
- In-memory data structures: disk input, shuffling, disk output
- In-memory shuffle of updates: two buffers
  1. Store updates from scatter phase
  2. Store result of in-memory shuffle
- Parallelism: process partitions in parallel

Performance

- Evaluation: test 10 algorithms on real and synthetic graphs
- Performs well, except for traversals on large diameter graphs
  - "... the diameter of real-world graphs only grows sub-logarithmically with the number of vertices"
- Scalable with increasing number of I/O devices and cores



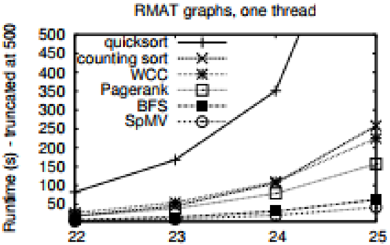


Comparison with Ligra

Ligra

- In-memory graph processing system designed for traversals
- Requires sorting and index list

Threads	Ligra (s)	X-Stream (s)	Ligra-pre (s)
BFS			
1	11.10	168.50	1250.00
2	5.59	86.97	647.00
4	2.83	45.12	352.00
8	1.48	26.68	209.40
16	0.85	18.48	157.20
Pagerank			
1	990.20	455.06	1264.00
2	510.60	241.56	654.00
4	269.60	129.72	355.00
8	145.40	83.42	211.40
16	79.24	50.06	160.20



Comparison with GraphChi

GraphChi

- Graph processing on a single machine
- Targets larger sequential bandwidth of SSD and disk
- *Sorted* shards, all vertices and edges must fit in memory

	Pre-Sort (s)	Runtime (s)	Re-sort (s)
Twitter pagerank			
X-Stream (1)	*none*	397.57 ± 1.83	–
Graphchi (32)	752.32 ± 9.07	1175.12 ± 25.62	969.99
Netflix ALS			
X-Stream (1)	*none*	76.74 ± 0.16	–
Graphchi (14)	123.73 ± 4.06	138.68 ± 26.13	45.02
RMAT27 WCC			
X-Stream (1)	*none*	867.59 ± 2.35	–
Graphchi (24)	2149.38 ± 41.35	2823.99 ± 704.99	1727.01
Twitter belief prop.			
X-Stream (1)	*none*	2665.64 ± 6.90	–
Graphchi (17)	742.42 ± 13.50	4589.52 ± 322.28	1717.50

Future work: Chaos

- Builds on streaming partitions of X-Stream
- X-Stream: limited by bandwidth and capacity of single machine
- Scale to cluster: process partitions in parallel

Summary

A system for processing large graphs on a *single shared-memory machine* using

1. edge-centric scatter gather
2. sequential streaming partitions

Summary

A system for processing large graphs on a *single shared-memory machine* using

1. edge-centric scatter gather
2. sequential streaming partitions

Questions?

References

- Joseph E Gonzalez et al. "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs." In: *OSDI*. Vol. 12. 1. 2012, p. 2.
- Aapo Kyrola, Guy E Blelloch, and Carlos Guestrin. "Graphchi: Large-scale graph computation on just a pc". In: USENIX. 2012.
- Yucheng Low et al. "Graphlab: A new framework for parallel machine learning". In: *arXiv preprint arXiv:1408.2041* (2014).
- Grzegorz Malewicz et al. "Pregel: a system for large-scale graph processing". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM. 2010, pp. 135–146.
- Amitabha Roy, Ivo Mihailovic, and Willy Zwaenepoel. "X-stream: Edge-centric graph processing using streaming partitions". In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM. 2013, pp. 472–488.
- Amitabha Roy et al. "Chaos: Scale-out graph processing from secondary storage". In: *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM. 2015, pp. 410–424.
- Julian Shun and Guy E Blelloch. "Ligra: a lightweight graph processing framework for shared memory". In: *ACM Sigplan Notices*. Vol. 48. 8. ACM. 2013, pp. 135–146.