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

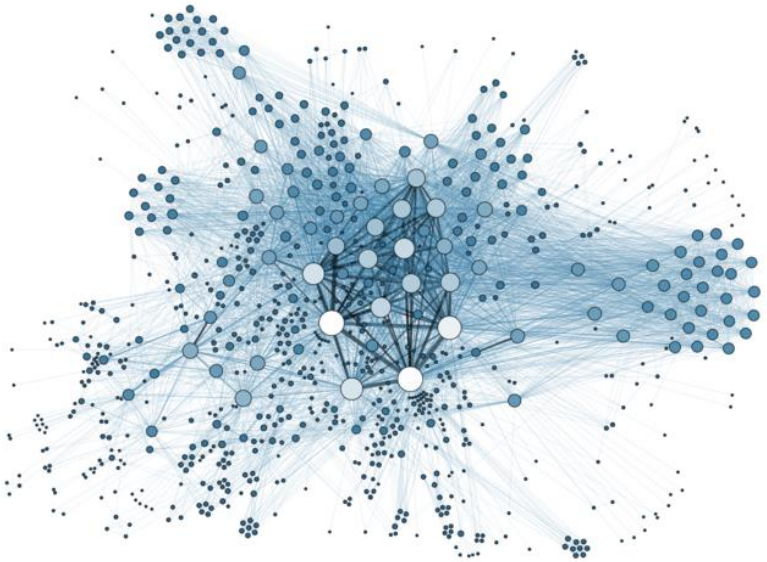
Authors: Amitabha Roy, Ivo Mihailovic, Willy Zwaenepoel

R244 Session 3

30 October 2024

Background

- Large-scale graphs
- Standard Scale-Out approach
  - Google Pregel
  - PowerGraph
- Single Machine?
  - Ligra
  - X-Stream



Existing Approach

- Scatter-Gather programming model
  - **Scatter**: Propagate updates across edges
  - **Gather:** Apply incoming updates

- Many graph problems can be expressed in this model
  - PageRank

```
vertex_scatter(vertex v)
  send updates over outgoing edges of v

vertex_gather(vertex v)
  apply updates from inbound edges of v

while not done
  for all vertices v that need to scatter updates
    vertex_scatter(v)
  for all vertices v that have updates
    vertex_gather(v)
```

Figure 1: Vertex-centric Scatter-Gather

Existing Approach – Vertex-Centric

Way fewer vertices than edges

Existing systems:

1. Sort edges of the graph based on the source vertex

2. Build an index for the edges

3. For each vertex:
   - Perform a lookup through the index to obtain outgoing edges

Main Bottleneck

- Access Locality

- Sequential Access >>> Random Access
  - 500x faster for HDDs
  - 30x faster for SSDs
  - Even main memory (2x faster)

- Great opportunity for performance gains if sequential access is leveraged

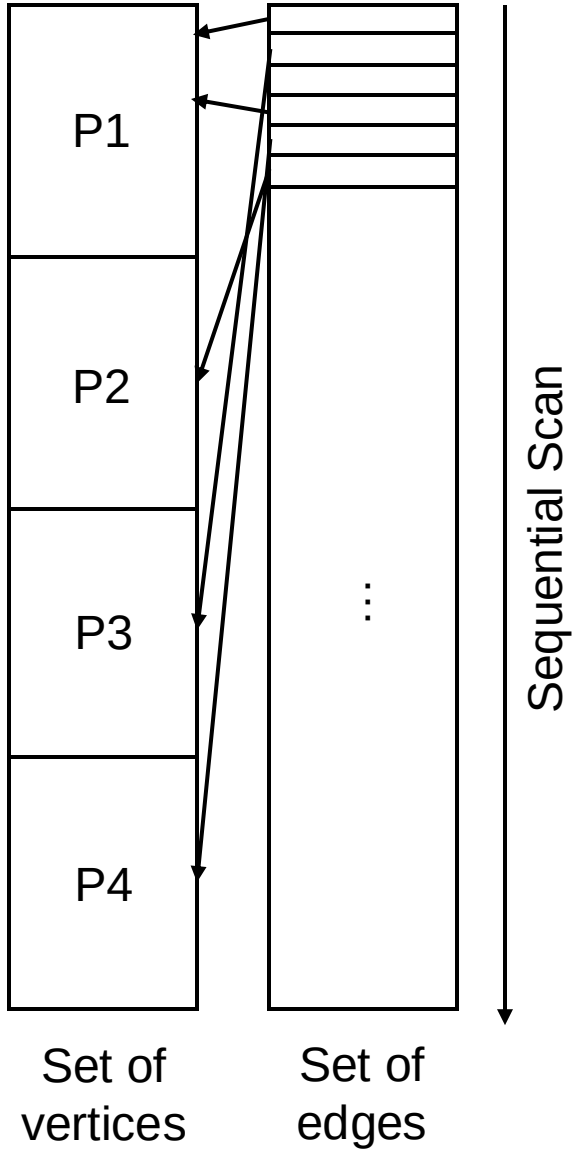
Two-tier Storage Hierarchy

- We can afford random lookups on **Fast Storage** mediums
- But **Slow Storage** mediums **should** only be accessed sequentially
- In-memory graphs:
  - **Fast Storage**: CPU Cache
  - **Slow Storage**: Main Memory
- Out-of-core graphs:
  - **Fast Storage**: Main Memory
  - **Slow Storage**: HDDs/SSDs

Streaming Partitions

1. Subset of **vertices** (fits entirely into fast storage)
2. Subset of **edges** (streamed from slow storage)
   - All edges whose source vertex is in the vertex set
3. Set of **updates** (streamed from slow storage)
   - All updates whose destination vertex is in the vertex set

- Streaming Buffers (in fast storage)



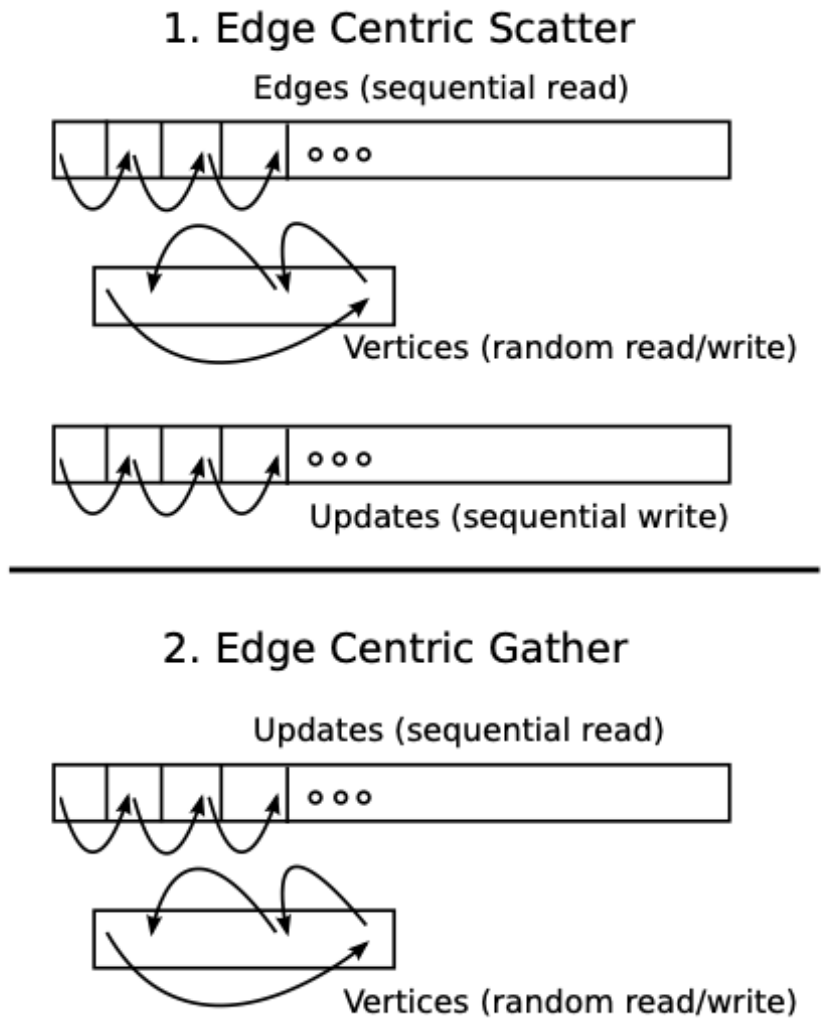


Figure 3: Streaming Memory Access

```
merged scatter/shuffle phase:
  for each streaming partition s
    while edges left in s
      load next chunk of edges into input buffer
      for each edge e in memory
        edge_scatter(e) appending to output buffer
        if output buffer is full or no more edges
          in-memory shuffle output buffer
          for each streaming partition p
            append chunk p to update file for p

gather phase:
  for each streaming_partition p
    read in vertex set of p
    while updates left in p
      load next chunk of updates into input buffer
      for each update u in input buffer
        edge_gather(u)
    write vertex set of p
```

Figure 6: Disk Streaming Loop

In-memory

- To fully utilise the high streaming bandwidths of main memory, parallelism is required

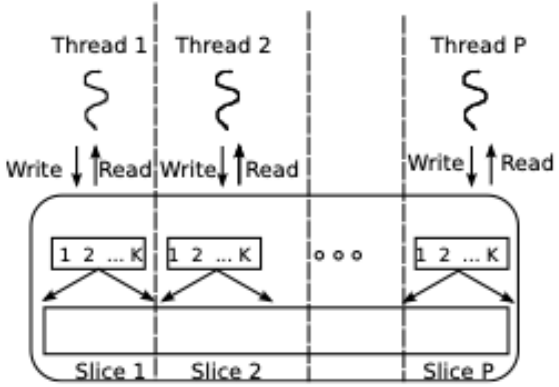


Figure 7: Slicing a Streaming Buffer

- Since the CPU Cache has limited storage, way more streaming partitions are required

Evaluation

	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

Figure 22: Comparison with Graphchi on SSD with 99% Confidence Intervals. Numbers in brackets indicate X-Stream streaming partitions/Graphchi shards (Note: re-sorting is included in Graphchi runtime.)

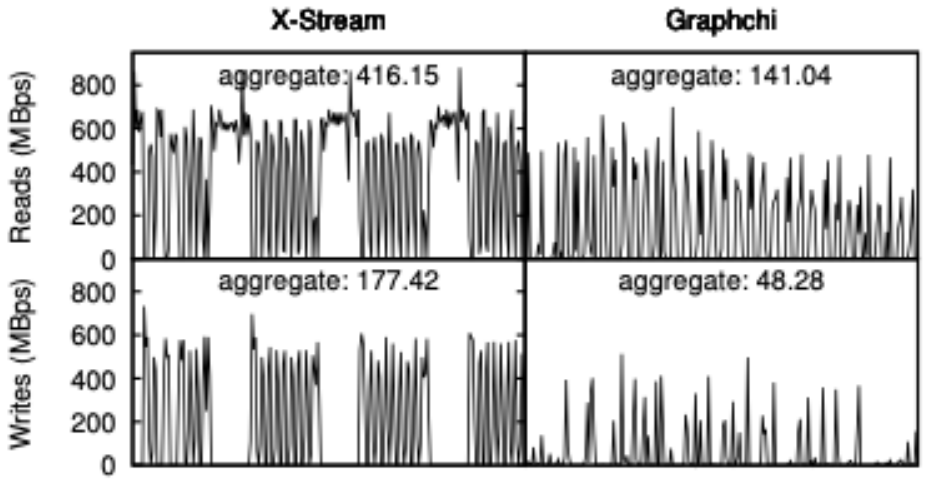


Figure 23: Disk Bandwidth

Evaluation

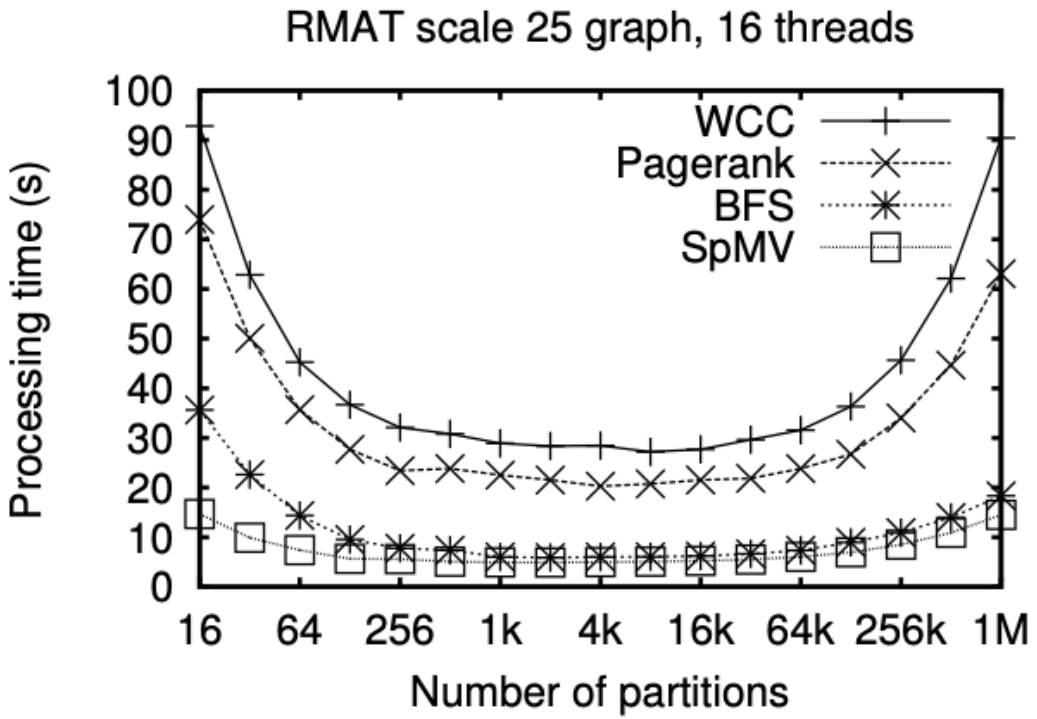


Figure 24: Effect of the Number of Partitions

Weaknesses

- Does not perform well with high-diameter graphs – too many scatter-gather iterations required
- Evaluation for out-of-core graphs is quite limited
- No mention of fault tolerance
- “Wasted” edges problem

	# iters	ratio	wasted %
memory			
amazon0601	19	2.58	63
cit-Patents	21	2.20	50
soc-livejournal	13	2.13	57
dimacs-usa	**6263**	1.94	**98**
ssd			
Friendster	24	1.06	63
sk-2005	25	1.04	67
Twitter	16	1.04	55
disk			
Friendster	24	1.04	63
sk-2005	25	1.04	67
Twitter	16	1.04	55
yahoo-web	—	—	—

Edge-centric weakness

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

Strengths

- Edge list does not need to be sorted
- Adaptable to various setups:
  - In-memory or Out-of-Core
  - CPU Cache, Main Memory, SSD, HDD
- Utilisation of I/O bandwidth