G-Store: High Performance Graph Store for Trillion-edge Processing

Pradeep Kumar and H. Howie Huang

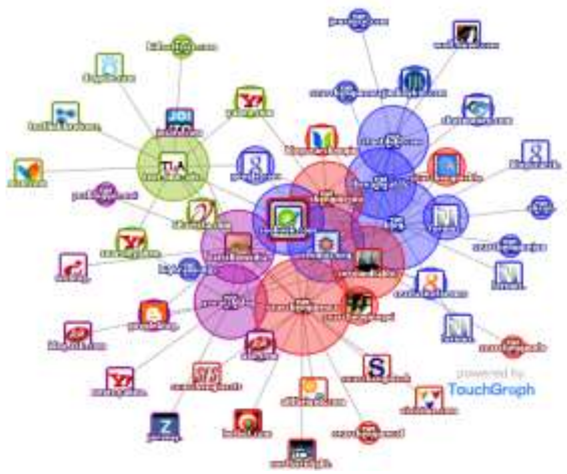
The George Washington University

Introduction

- Graph is everywhere

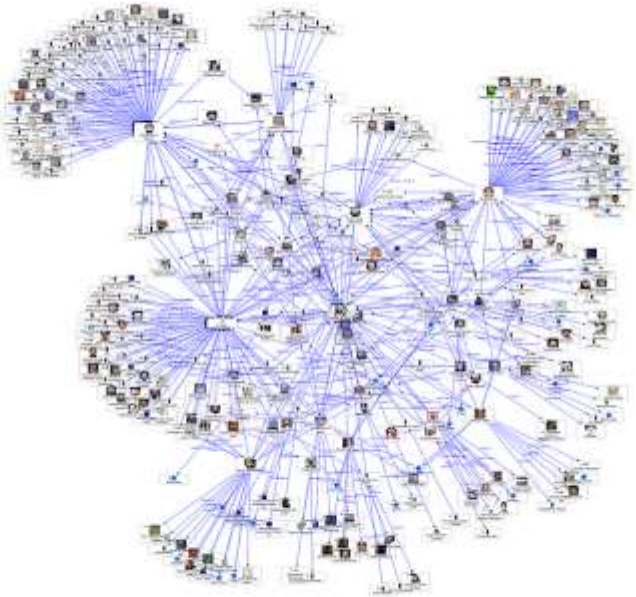
- Applications include
  - Social Networks
  - Computer Networks
  - Protein Structures
  - Recommendation Systems
  - And many more …

- Graph size is reaching to Trillion edges



GW

Motivation



Trillion-edge graph

=

***Lower* capacity but *Fast* access**

+

***Higher* capacity but *Slow* access**

Outline

01 - Observations

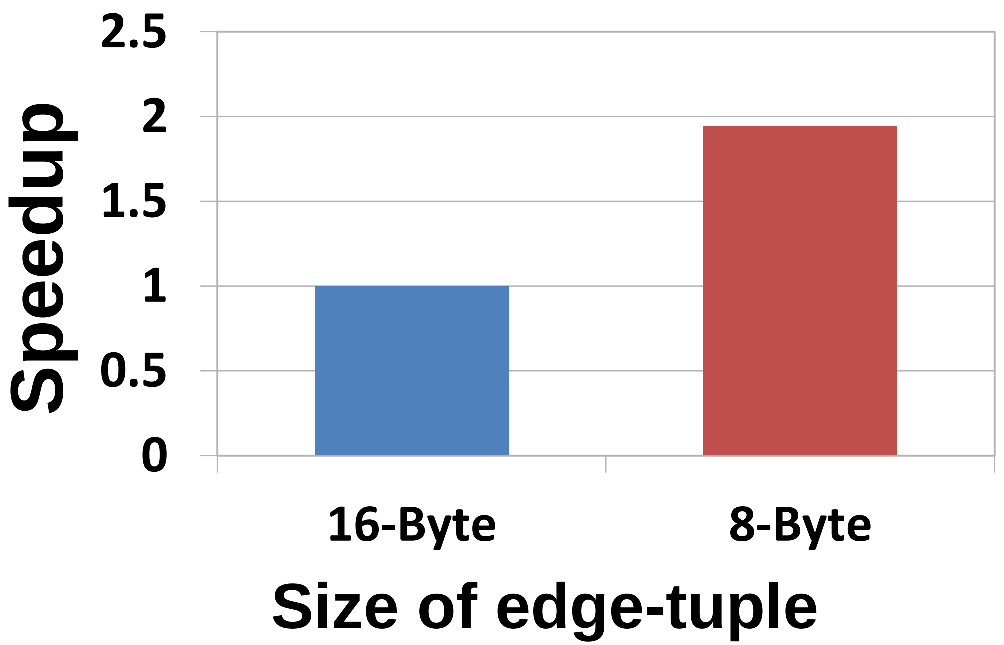
02 – G-Store Architecture

03 - Experiment

04 – Conclusion and QA

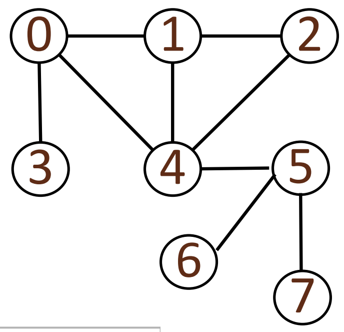
Observation 1: Graph Size

- PageRank algorithm on Kron-28-16 graph
  - 2^{28} vertices
  - 2^{33} edges
- Graph sizes vary
  - 128GB if using 16-byte for an edge-tuple
  - 64GB for 8-byte edge-tuple
- Smaller graph size leads to better performance



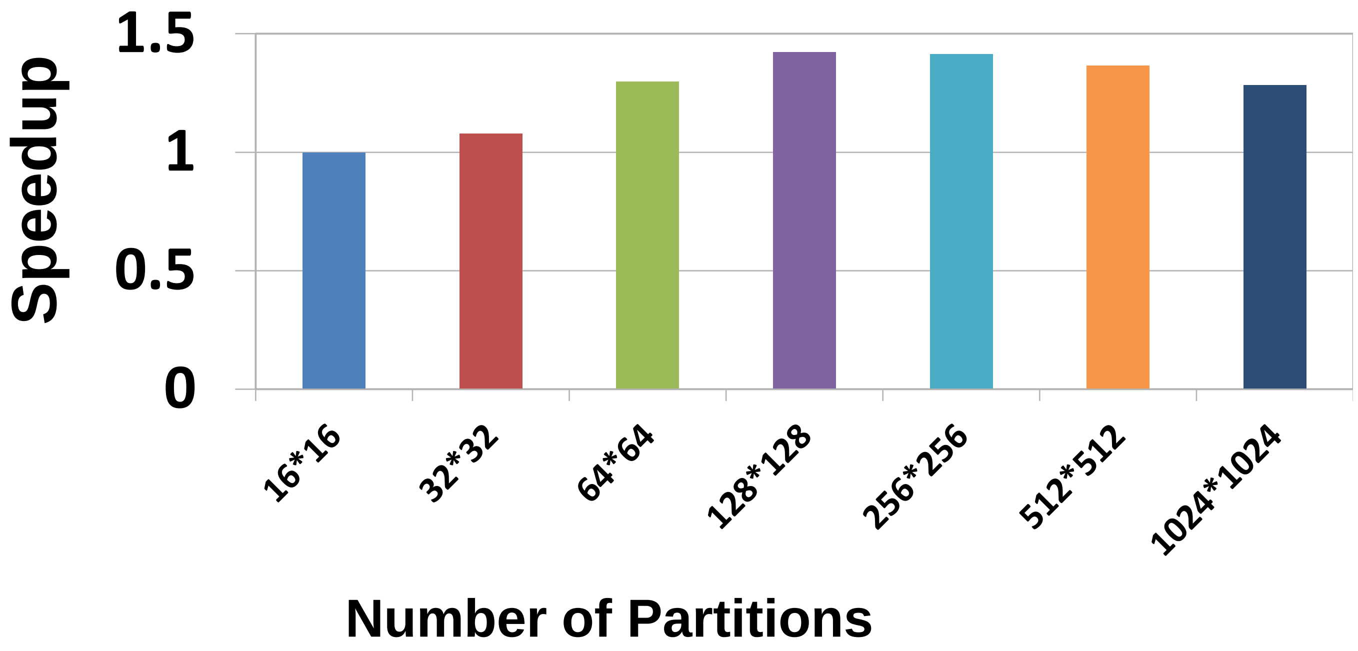
Observation 2: Metadata Access Localization

- PageRank on 2-D partitioned graph



(0,1)(0,3), (1,0),(1,2), (2,1),(3,0)	(0,4),(1,4), (2,4)
(4,0),(4,1), (4,2)	(4,5),(5,4), (5,6),(5,7), (6,5),(7,5)

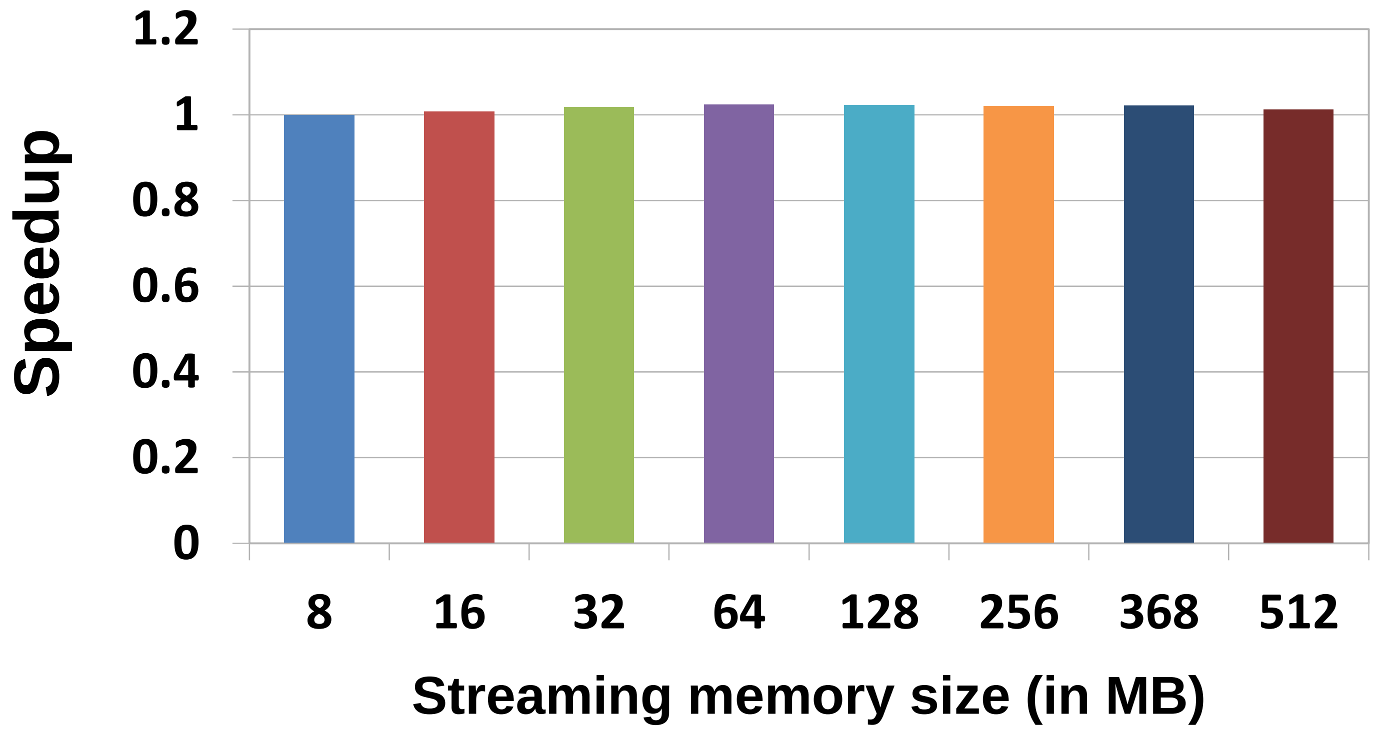
- In-memory processing performance



- Partition count affects metadata locality

Observation 3: Streaming Memory Size

- Small streaming memory does not significantly affect IO performance



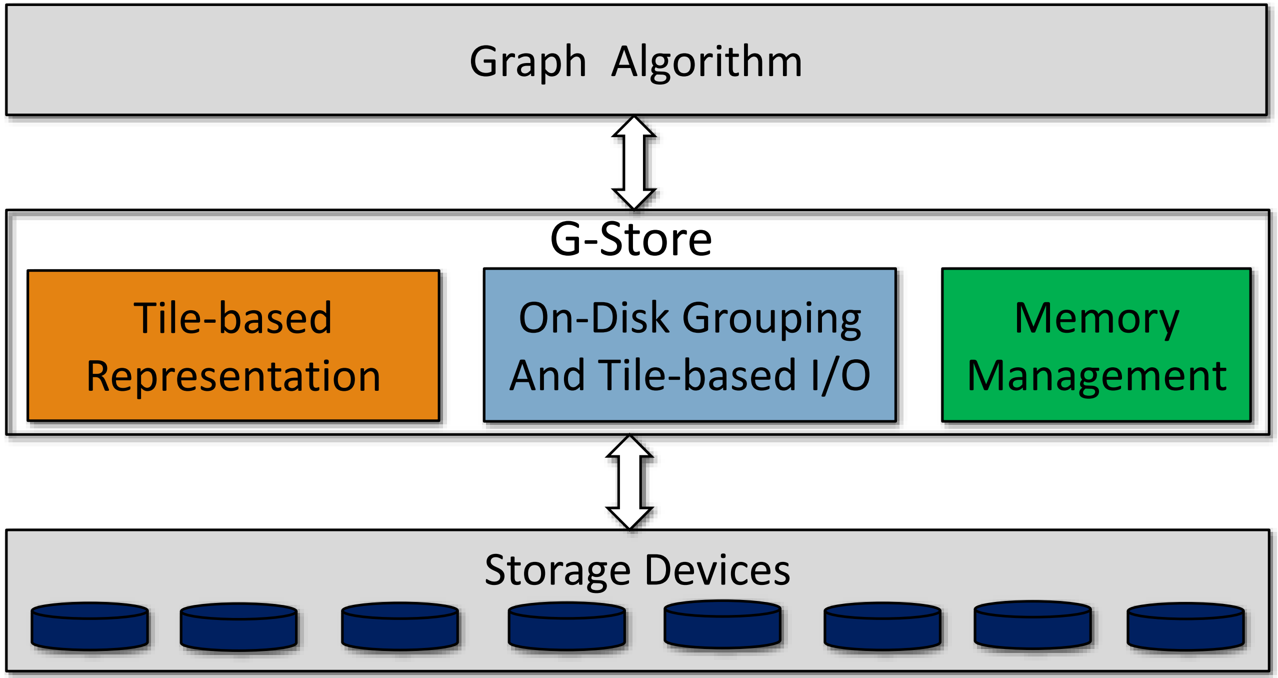
- Use rest of the memory for caching to improve algorithm performance

G-Store

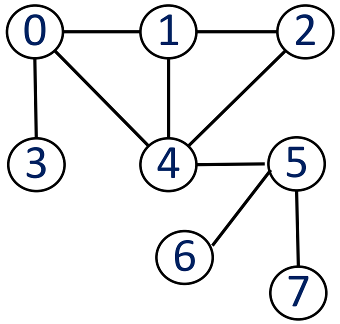
01 Graph Size
- Symmetry
- SNB Format

02 Metadata
- Grouping
- On-Disk Layout

03 Memory
- Proactive Caching
- Slide-Cache-Rewind



Tile-based Representation



(a) Example graph

0	0	0	1	1	1	2	2	3	4	4	4	4	5	5	5	6	7
1	3	4	0	2	4	1	4	0	0	1	2	5	4	6	7	5	5

(b) Edge-list

(0,1),(0,3),(1,0),
(1,2),(2,1),(3,0)

partition[0,0]

(0,4),(1,4),(2,4)

partition[0,1]

(4,0),(4,1),(4,2)

partition[1,0]

(4,5),(5,4),(5,6),
(5,7),(6,5),(7,5)

partition[1,1]

(c) 2-D Partitioned Graph

(0, 1), (0, 3),
(1, 2)

Tile[0, 0]

(0, 4), (1, 4),
(2, 4)

Tile[0, 1]

(4, 5), (5, 6),
(5, 7)

Tile[1, 1]

(d) Symmetry advantage

(0x00, 0x01), (0x00, 0x11),
(0x01, 0x10)

Tile[0x0, 0x0]

(0x00, 0x00), (0x01, 0x00),
(0x10, 0x00)

Tile[0x0, 0x1]

(0x00, 0x01), (0x01, 0x10),
(0x01, 0x11)

Tile[0x1, 0x1]

(e) Smallest Number of Bits (SNB) Format

Tile Advantages

- G-Store achieves upto 8x space saving
  - 2x due to Symmetry
  - Upto 4x due to Smallest Number of Bit representation (SNB)

- Processing can run directly on top of the SNB format with no need for conversion
  - Use a new relative pointer per tile for each algorithmic metadata
  - Please refer to the paper for details

Small Algorithm Change Needed for Tile

Algorithm 1 BFS on the partition[i,j] of undirected graph

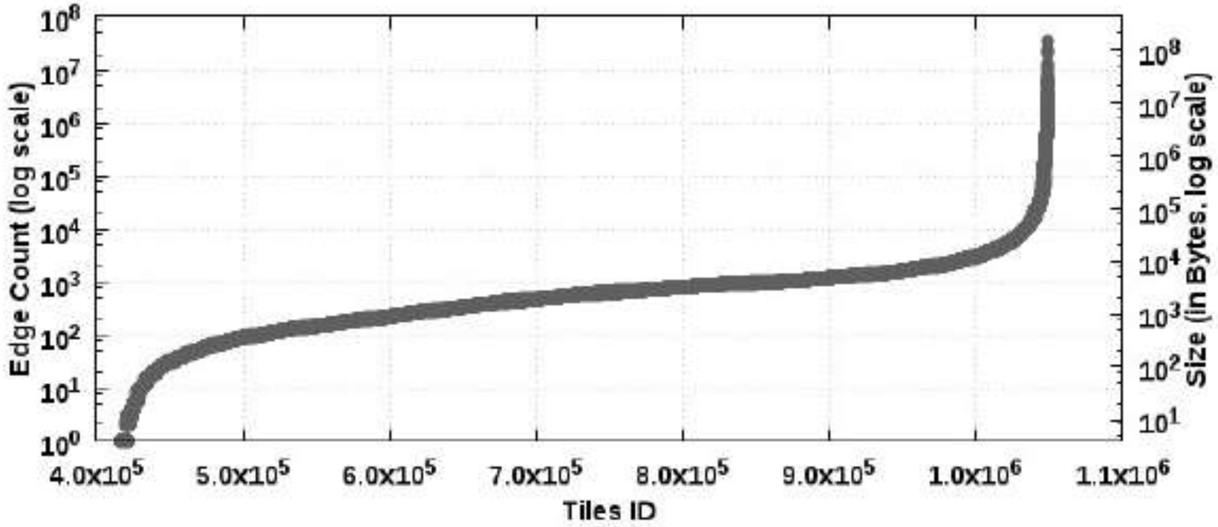
```
edge ← get_edge_ptr(i, j);
for k ← 1, edge_count(i, j) do
    src ← edge[k].src;
    dst ← edge[k].dst;
    if depth[src] == level & depth[dst] == INF then
        depth[dst] ← level + 1;
    end if
    // Added code for new storage format
    if depth[dst] == level & depth[src] == INF then
        depth[src] ← level + 1;
    end if
end for
```

Forward Direction (lines 5–7: the first if block)

Backward Direction (lines 9–11: the second if block)

Tile Properties

- Tile size of Twitter graph
  - Power law distribution



- Tile metadata size
  - All smaller than LLC

Algorithm	Metadata Size
Page-Rank	256KB
Connected Components	256KB
BFS	64KB

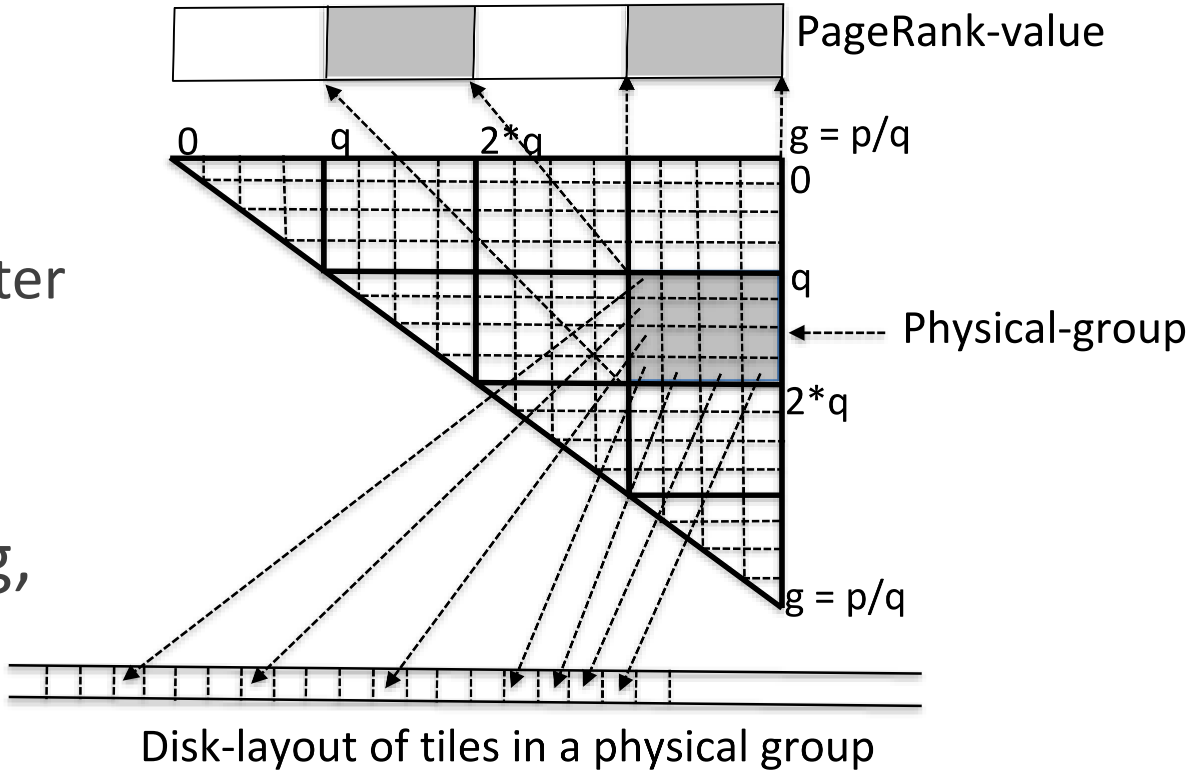
Grouping and On-disk Layout

- Combine *q*q* tiles into a physical group

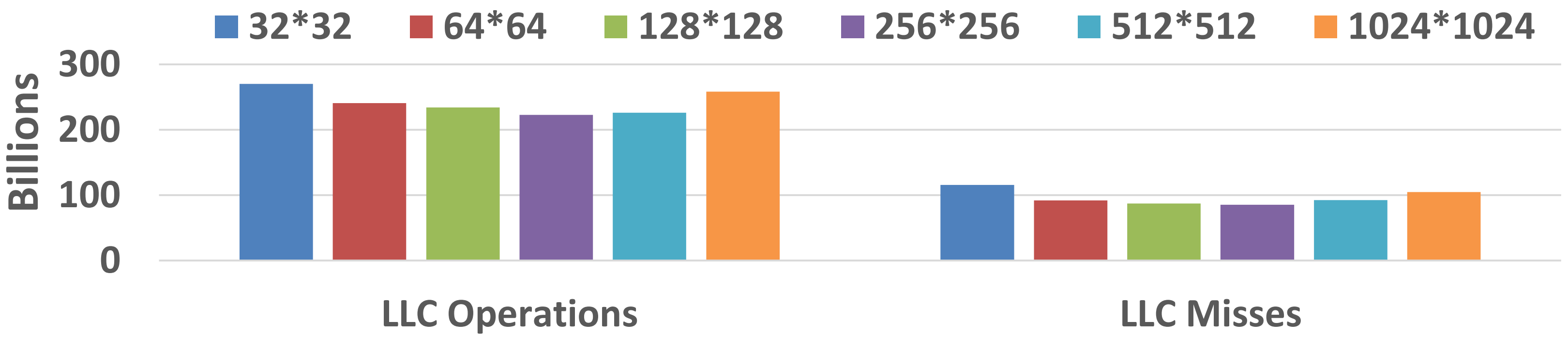
- Layout of tiles in disk
  - Reading tiles sequentially provides better hit ratios on LLC

- Tile is a basic unit for Data Fetching, Processing and Caching

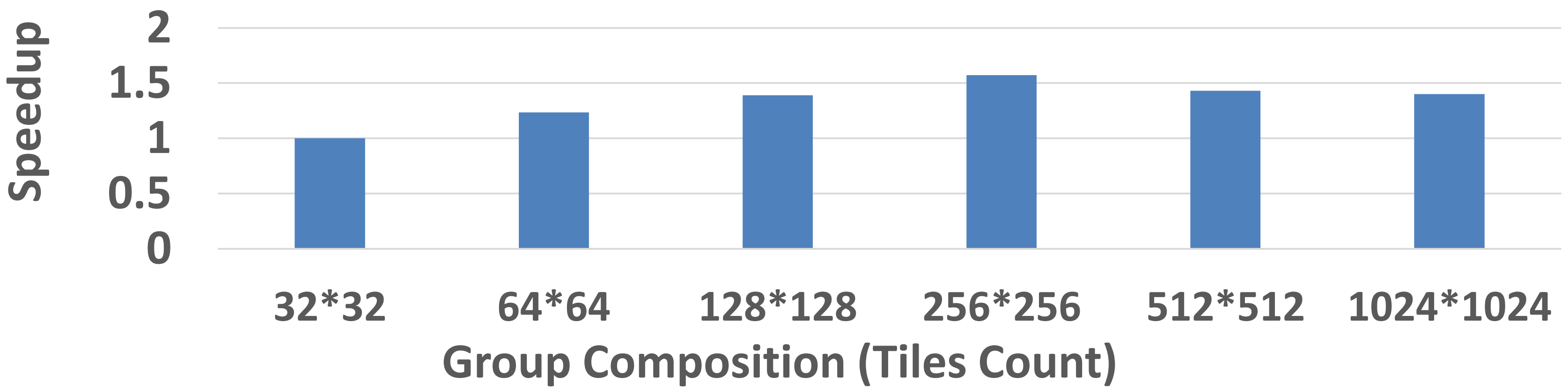
- Use Linux AIO



Grouping and On-disk Layout: Advantage



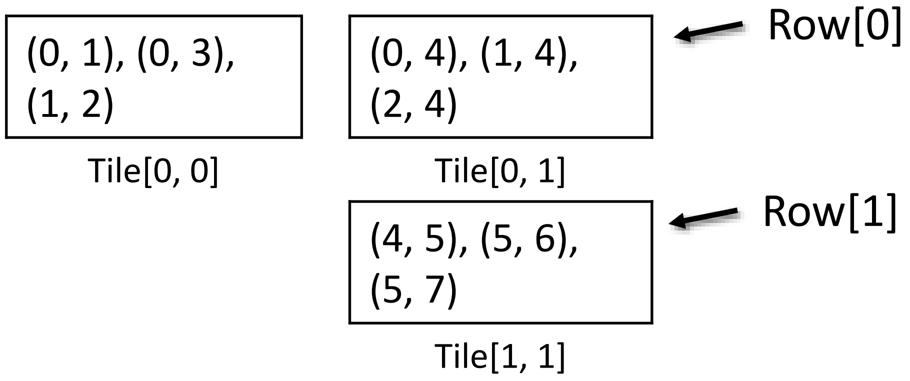
- 256*256 grouping has fewest LLC operations (loads, store) and misses



- Improves performance by 57%

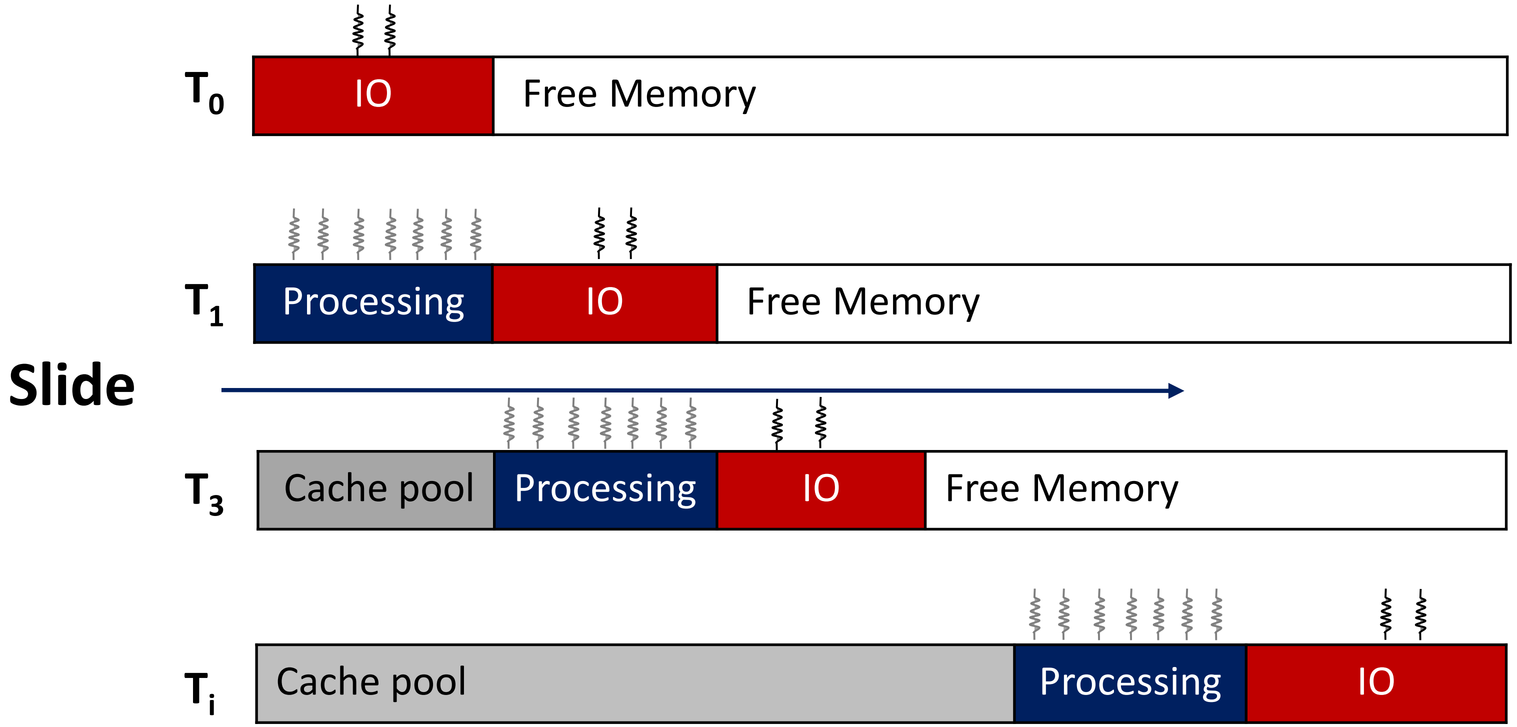
Proactive Caching

- Divide memory into streaming and caching
- Rule 1: at the end of the processing of *row[i]*
  - We know whether *row[i]* would be processed in the next iteration
  - Hints can be used later

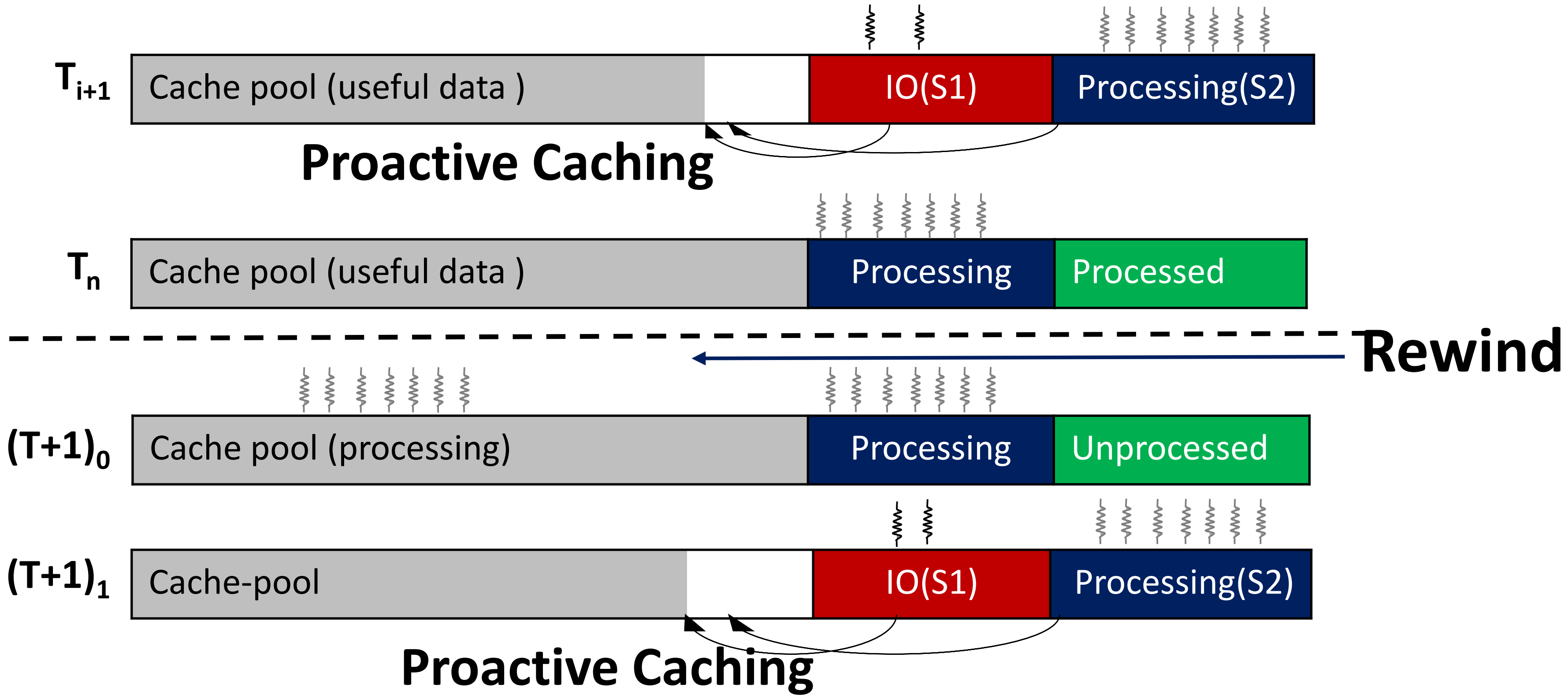


- Rule 2: if *row[i]* is not needed for next iteration
  - Then we know that *Tile[i,i]* will not be needed
  - We only have partial information about other individual tiles

Slide-cache-rewind



Slide-cache-rewind



- Less IO due to reuse
- Provide hints for proactive caching policy
- Evict unwanted data from cache pool

Experimental Setup

- Dual Intel Xeon CPU E5-2683, 14 core each, hyper-threading enabled = 56 threads
- 8GB streaming and caching memory
- Cache
  - 32KB data and instruction L1 cache
  - 256KB L2 cache
  - 16MB L3 cache
- LSI SAS9300-8i HBA, 8 Samsung EVO 850 512GB SSD
- Linux Software RAID0 with 64KB stripe size

Graph Datasets & Space Saving

Graph Name	Type	X-Stream Size	FlashGraph Size	G-Store Size	Space Saving Over X-Stream	Space Saving Over FlashGraph
Kron31-256	Undirected	8TB	4TB	2TB	4x	2x
Kron-33-16	Undirected	4TB	2TB	512GB	8x	4x
Kron-30-16	Undirected	256GB	128GB	64GB	4x	2x
Kron-28-16	Undirected	64GB	32GB	16GB	4x	2x
Rmat-28-16	Undirected	64GB	32GB	16GB	4x	2x
Random-27-32	Undirected	64GB	32GB	16GB	4x	2x
Twitter	(Un-)Directed	14.6GB	14.6GB	7.3GB	4x	2x
Friendster	(Un-)Directed	19.26GB	19.26GB	9.63GB	4x	2x
Subdomain	(Un-)Directed	15.22GB	15.22GB	7.6GB	4x	2x

Performance on Trillion-edge Graph

- Kron-31-256: 2^{31} vertices with 2^{40} edges
- Kron-33-16: 2^{33} vertices with 2^{38} edges

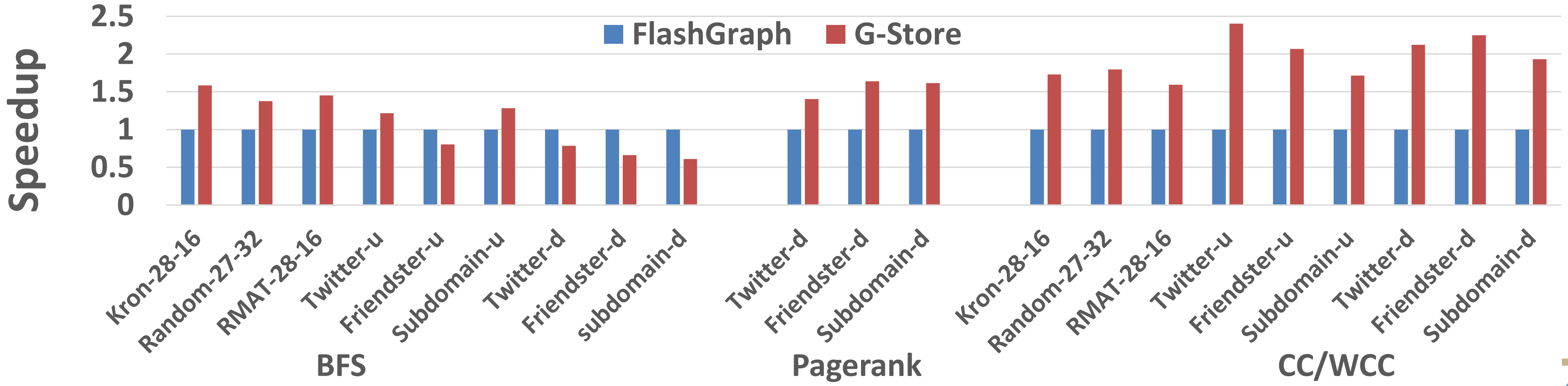
Graph	BFS	Page-rank	WCC
Kron-31-256	2548.54	4214.54	1925.13
Kron-33-16	1509.13	1882.88	849.05

- One iteration of Pagerank:
  - G-Store: **14 min** in a **single machine** for a trillion-edge graph
  - Ching et al*: **3 min** using **200 machines** for similar-sized graph

*Ching et al. One Trillion Edges: Graph Processing at Facebook-scale. Proceedings of the 41st International Conference on Very Large Data Bases(VLDB15)

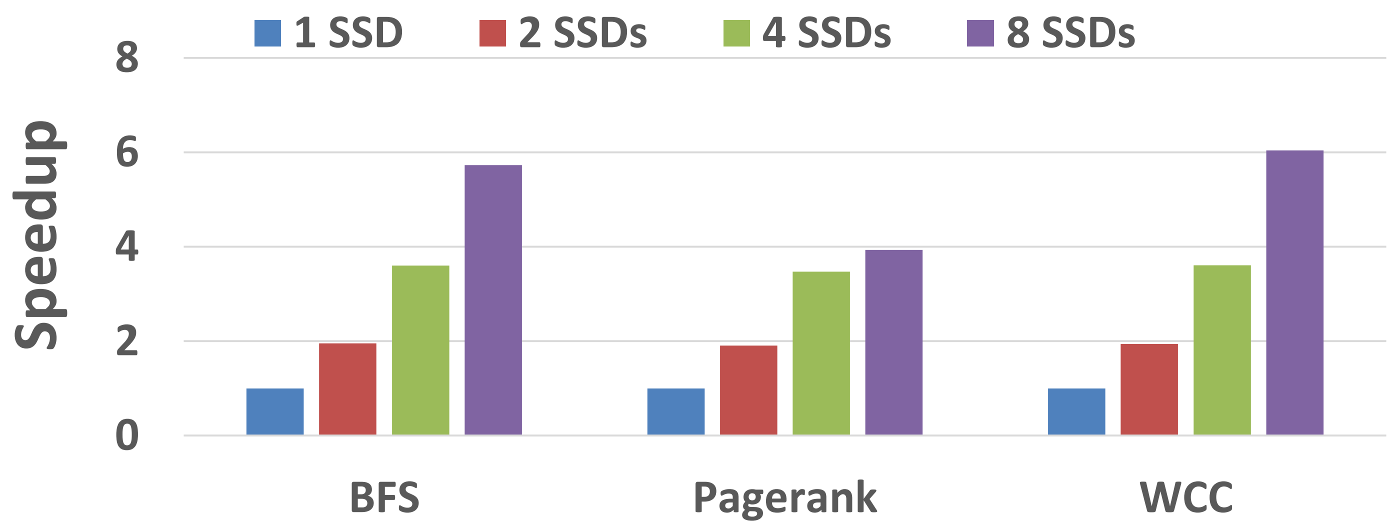
Performance Comparison

- Over X-Stream
  - 17x(BFS), 21x (PageRank), 32x(CC/WCC) speedup for Kron-28-16
  - Others are similar. See paper for details.
- Over FlashGraph
  - 1.5x (CC/WCC), 2x (PageRank), 1.4x (BFS, undirected)
  - Slightly poor for BFS on directed graph due to no space saving in smaller graph



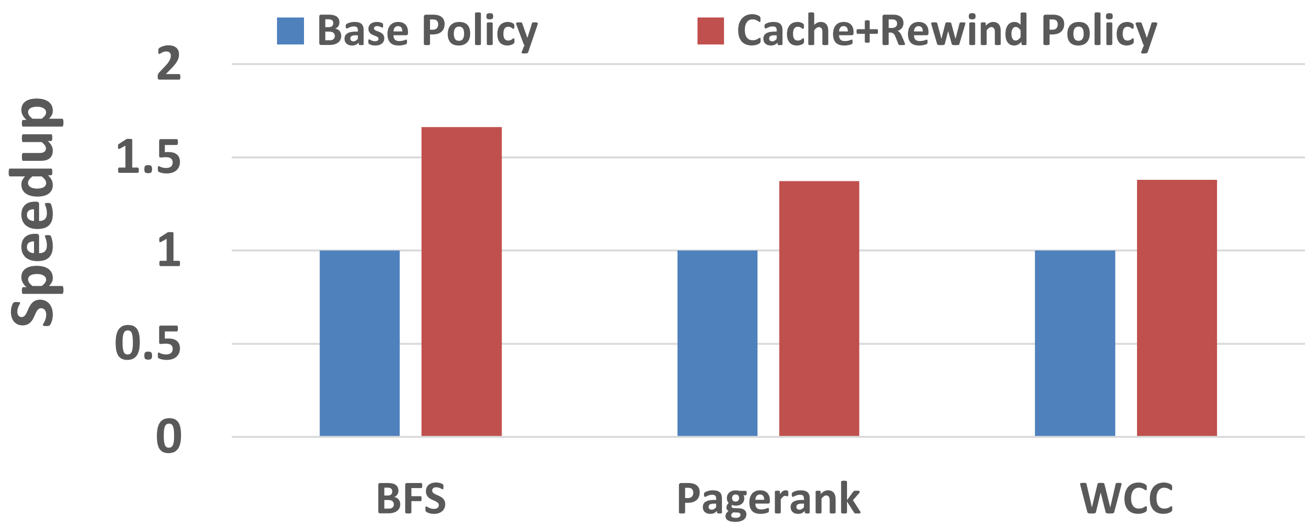
Scalability on SSDs

- RAID0: 64K stripe size
- Close to 4x speedup for 4-SSDs
- Upto 6x speedup for 8-SSDs
  - PageRank becomes compute intensive at 8-SSD configuration



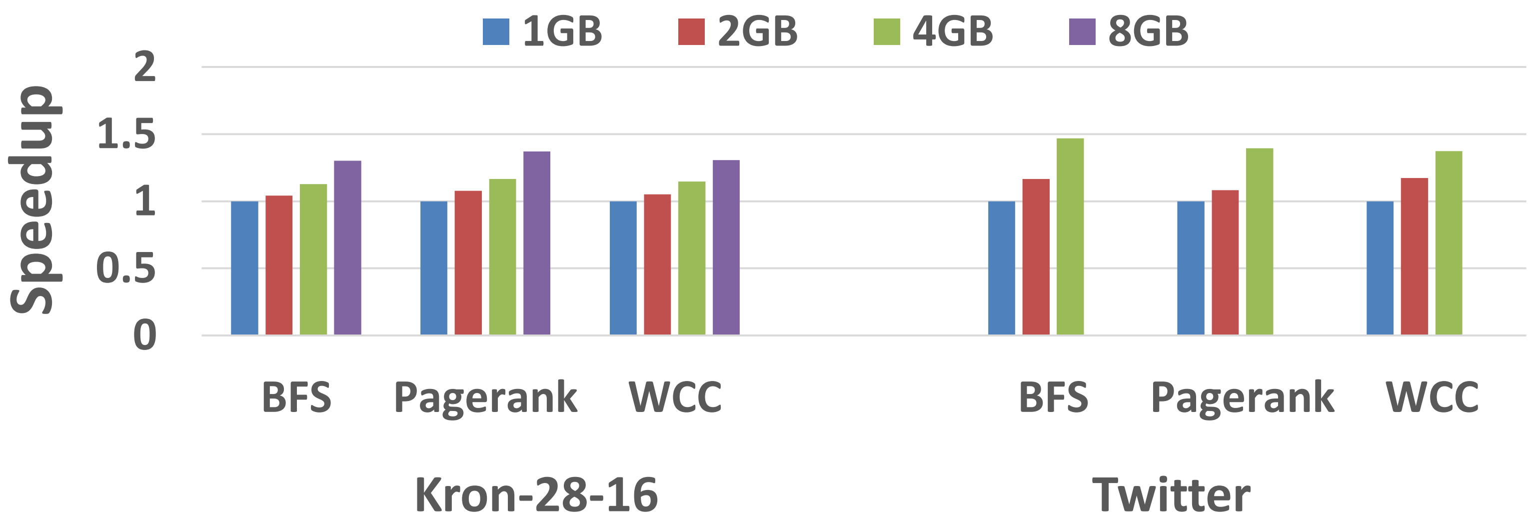
Slide-Cache-Rewind

- Base Policy (No cache): 4GB segment size (two)
- Cache+Rewind Policy: 7.5GB cache, 256MB segments (two)
- 60% (BFS), 35% (PageRank) and 35% (WCC)



Cache Size

- For Kron-28-16 graph from 1GB to 8GB
  - Average 30% speedup
- For Twitter graph from 1GB to 4GB
  - Average 41% speedup



Conclusion

- SNB: Space efficient representation
- Grouping: Optimal utilization of hardware cache
- Slide: Complete overlapping of IO and compute
- Cache: Graph specific proactive caching policy
- Rewind: Using the last drop of memory

Thank You

- Email: pradeepk@gwu.edu and howie@gwu.edu

- Graph software repository
  - https://github.com/iHeartGraph/
  - G-Store: High-Performance Graph Store for Trillion-Edge Processing (SC’16)
  - Enterprise: Breadth-First Graph Traversal on GPUs (SC’15)
  - iBFS: Concurrent Breadth-First Search on GPUs (SIGMOD’16)