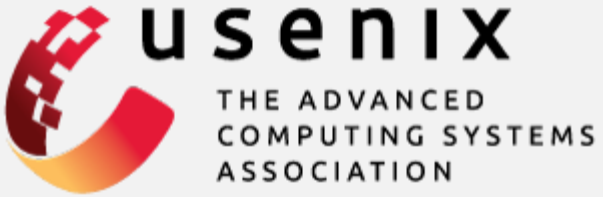
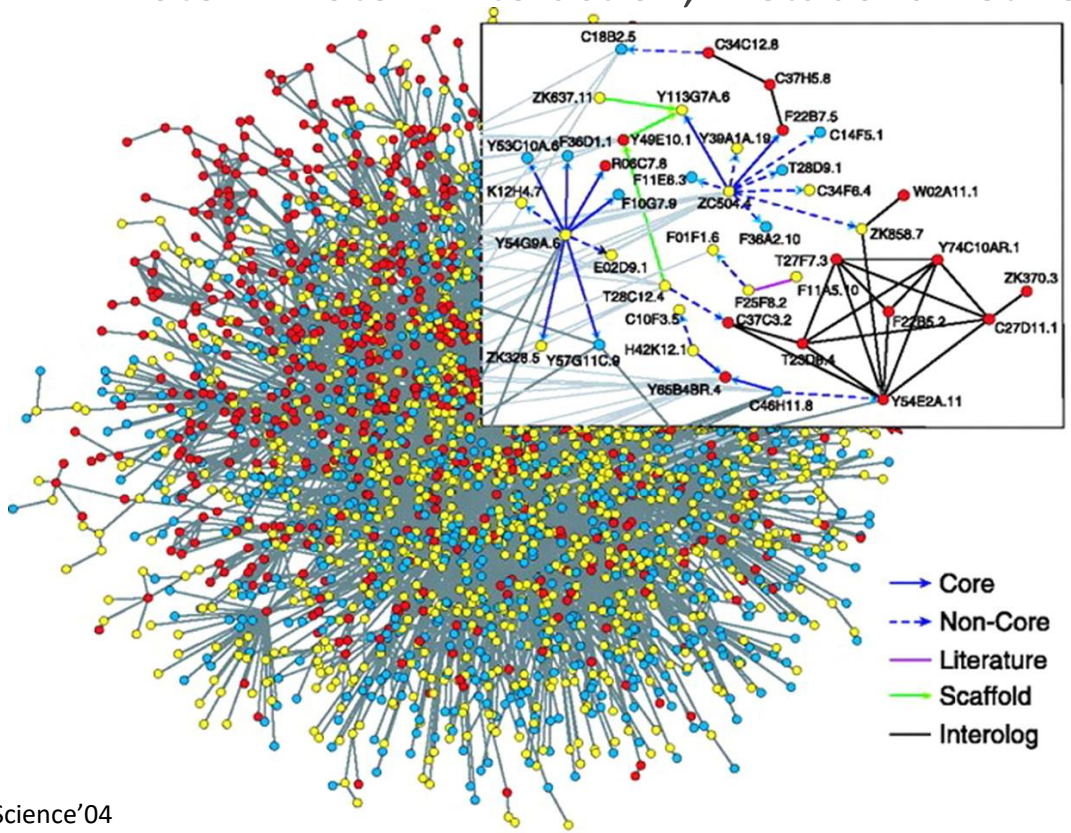
GraphOne: A Data Store for Real-time Analytics on Evolving Graphs

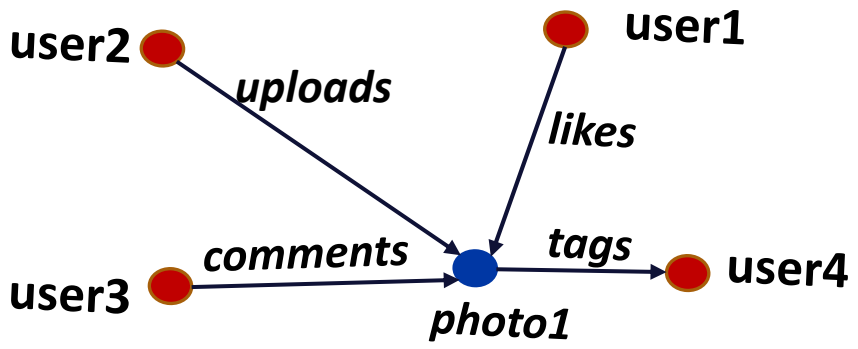
Pradeep Kumar, H. Howie Huang
The George Washington University



Graph Applications: Big Data and Scientific Applications

- **Basic Building Block** for Modern Trade
  - Social Network, Navigation, E-Commerce, etc.
  - **Metadata Store** for Big Data
- **Scientific Applications**
  - Protein-Protein Interaction, Metabolic Network, etc.

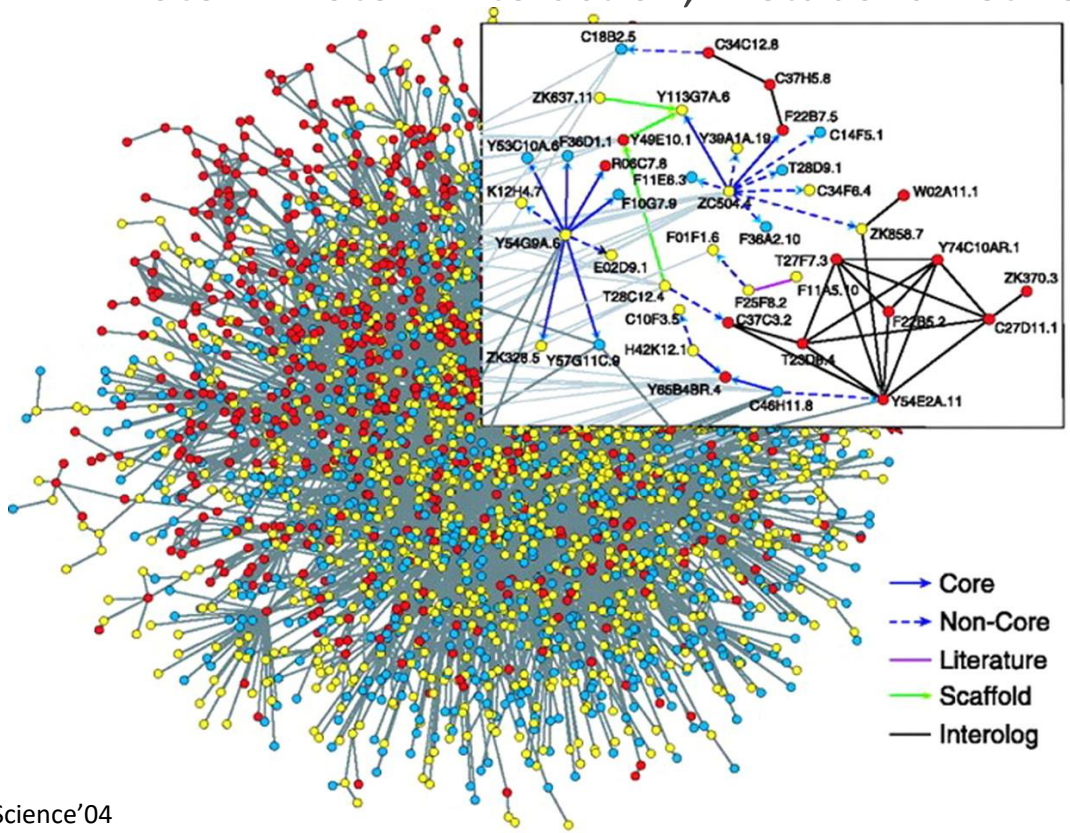


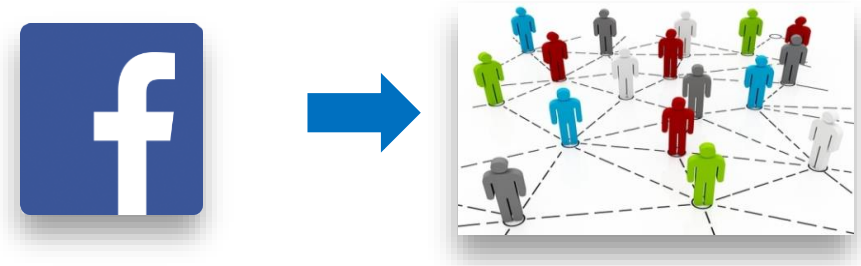


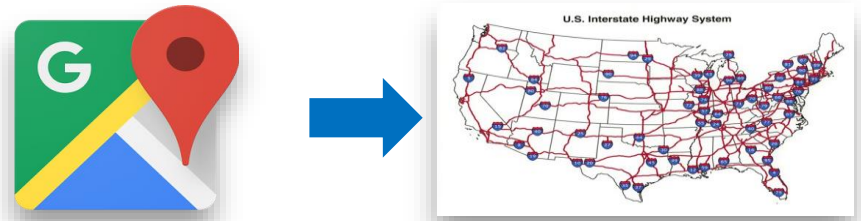
Li et al. Science'04

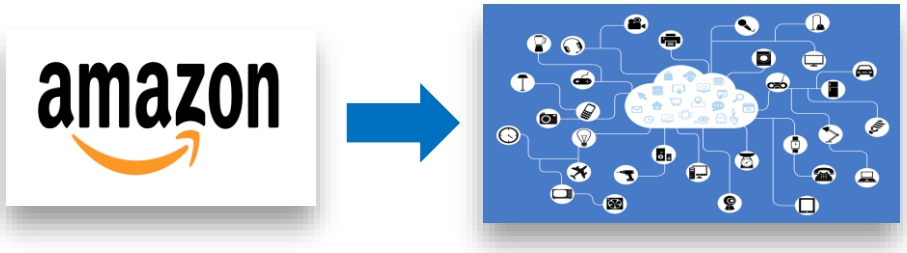
Graph Applications: Big Data and Scientific Applications

- **Basic Building Block** for Modern Trade
  - Social Network, Navigation, E-Commerce, etc.
  - **Metadata Store** for Big Data
- **Scientific Applications**
  - Protein-Protein Interaction, Metabolic Network, etc.







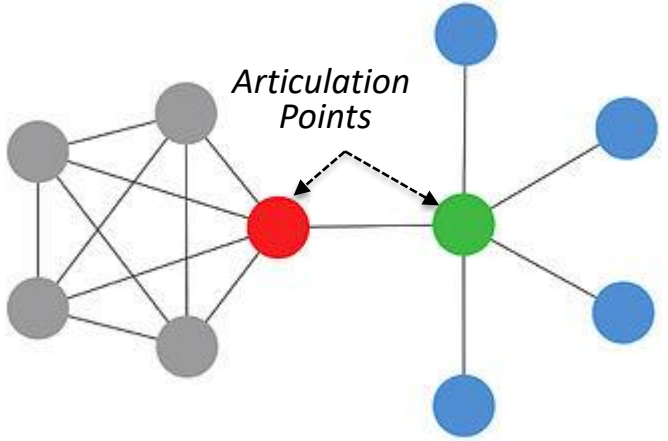


- **And Many More**
  - Identification of Vulnerable code, Cyber Attack
  - Knowledge Graph
  - Key for Data Governance, e.g. GDPR

Li et al. Science'04

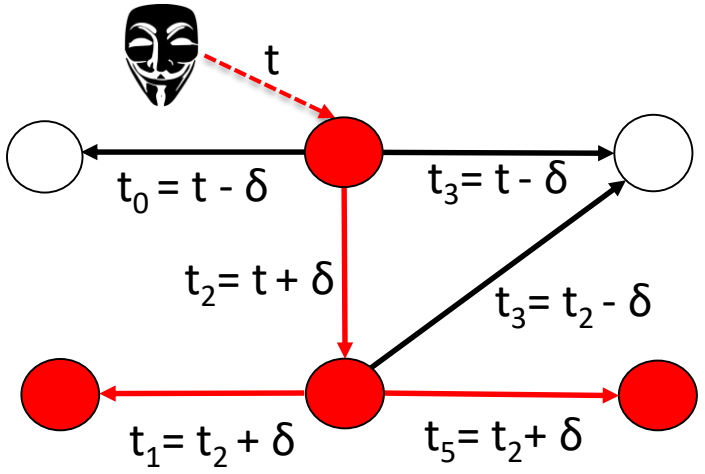
Definitions

- **Topological Graph:** Static (or slowly changing) Relationships
  - **Vertices:** Switches and Computers
  - **Edges:** Network Bandwidth between Vertices
- **Batch Analytics** on Topological Graph
  - Network Infrastructure Management
  - PageRank, Breadth First Traversal, etc.



(a) A Computer Network Showing Articulation Points

- **Streaming Graphs**: Faster Arrival Rate
  - **Vertices:** Users, Processes, Computers + Ports, etc.
  - **Edges:** Network Data Flow, Authentication Logs, Process Events
- **Stream Analytics** on Streaming Graph
  - Identification of Security Risks*



(b) A Stream Graph after Time t: Useful for finding all the infected machines

* Grades'15 Paper from Pacific Northwest National Laboratory; LANL Net-flow dataset, 2015 and 2017; computers and Security'15, …

Observation: Prior Works have Specialized/Private Data Stores

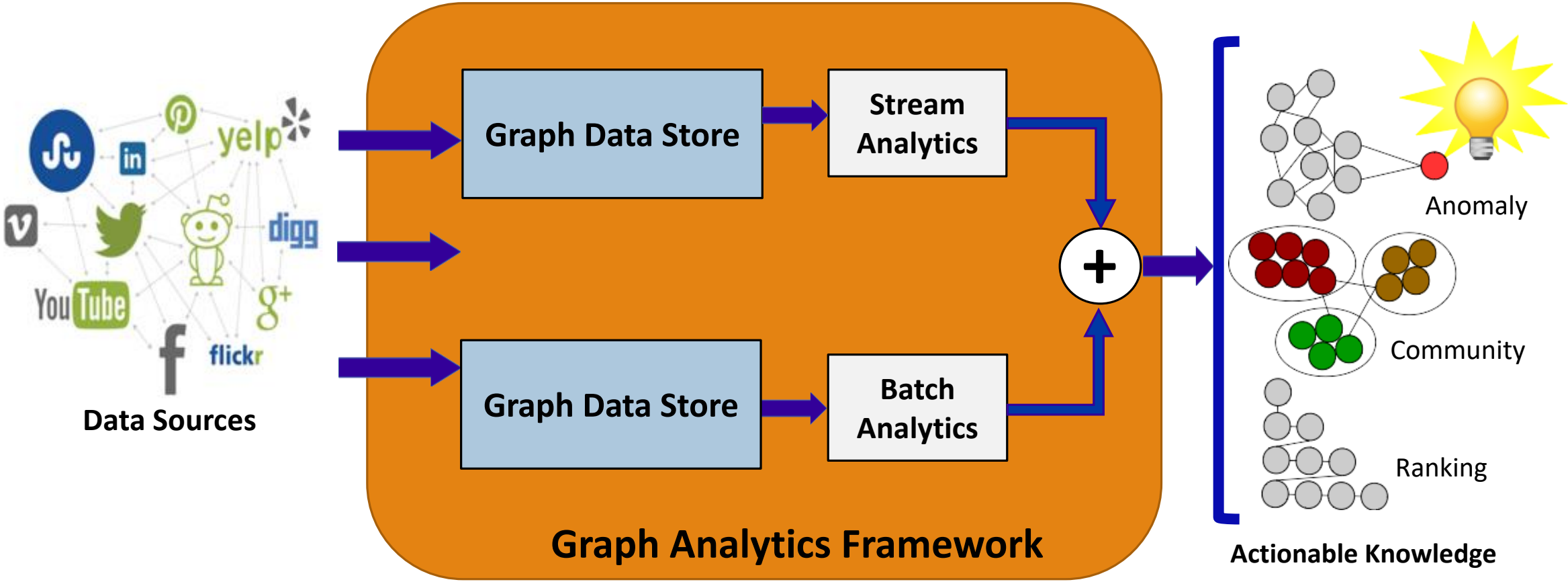
Research Question: How to perform *diverse set of real-time analytics* in presence of high arrival rate of graph data?

Findings:

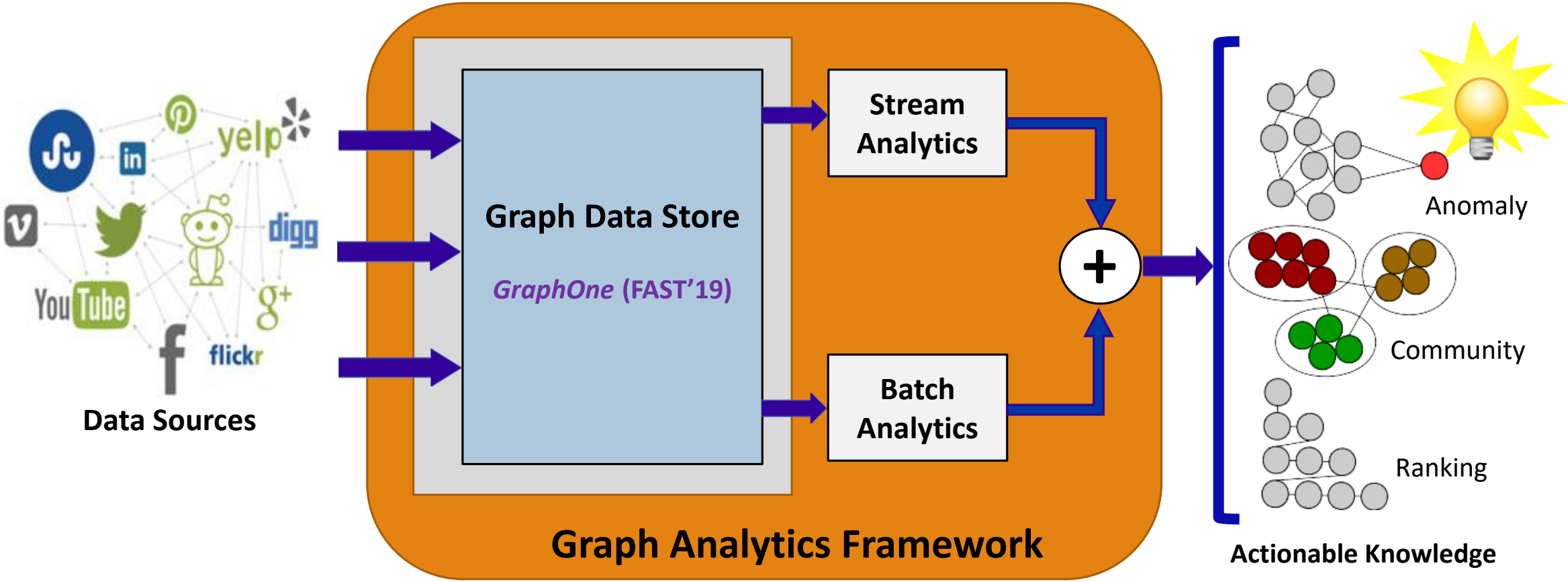
- Specialized/Private Data Stores
- Integration of Many Systems
  - Excessive Data Duplication
  - Weakest Link Problem

Solutions	Data Ingestion	Data Access Type
Graph Databases (For Short Queries)	Fine-grained (e.g**., Neo4j**)	Whole Data
Dynamic Graph Engines (For Batch Analytics)	Fine-grained (e.g., **Stinger**)	Whole Data
	Coarse-grained (e.g., **Graphchi**)	Whole Data
Stream Graph Engines (For Stream Analytics)	Coarse-grained (e.g. **Naiad**)	Streaming Access Of Whole Data
	Coarse-grained (e.g. **GraphTau**)	Streaming Access from a Time Window
GraphOne	**Both**	**All**

Abstracting Data Store away from Graph Analytics Engine



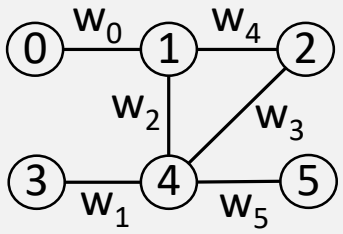
Idea: Abstracting Data Store away from Graph Analytics Engine



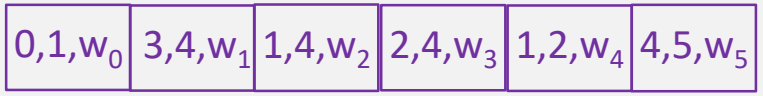
Results:

- Over **Neo4j and SQLite**: 2 to 3 orders of higher ingestion rate
- Over **Stinger**: 5.36x more ingestion rate, over 3x speedup for analytics
- Over **Static Graph Engine**: No Pre-processing Cost

GraphOne: Background and Architecture

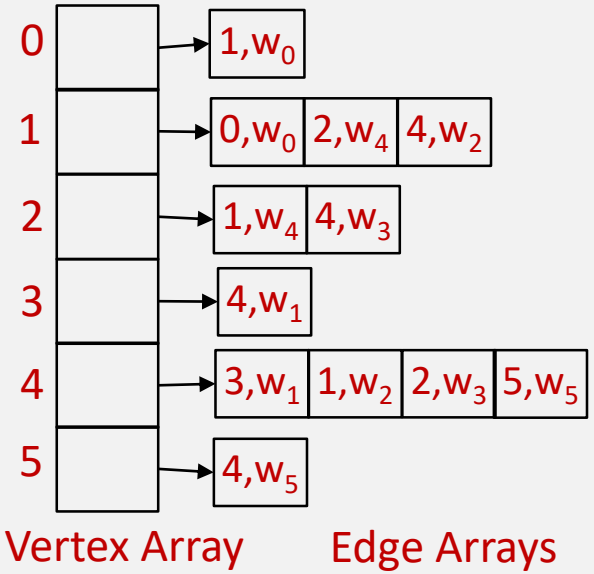


(a) Example graph



(b) Edge List

- **Log:** Captures Temporal Ordering
- Fine-grained Versioning is Free



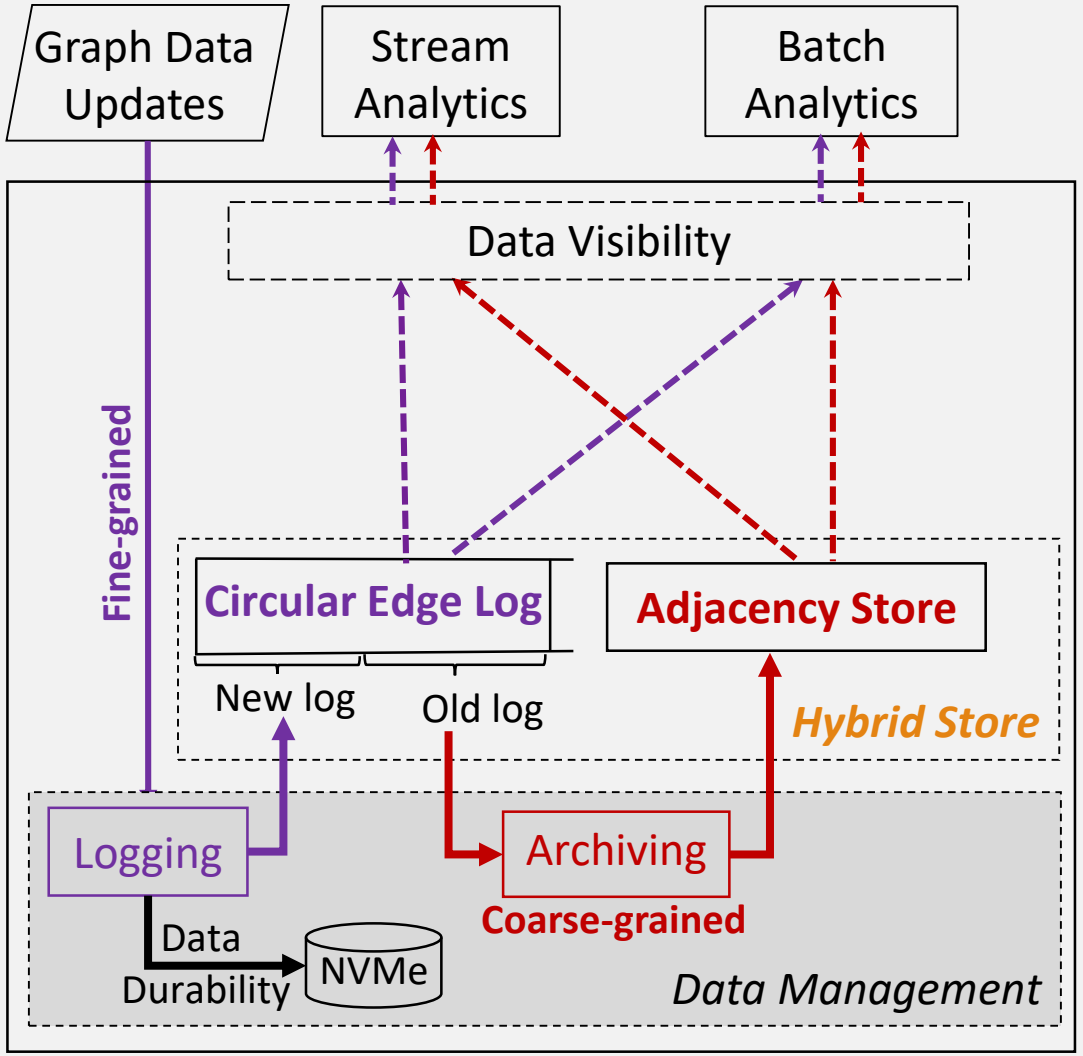
(c) Adjacency List

- **Indexed Data Structure**
- Coarse-grained Versioning

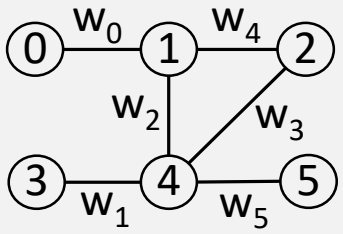
Opportunity: Hybrid Store

Two New Abstractions:

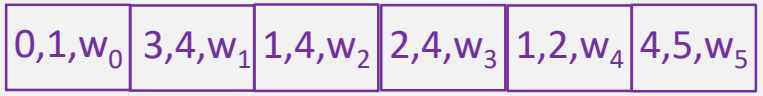
Data Visibility: *Analytics* choose their Ingestion type



GraphOne: Background and Architecture

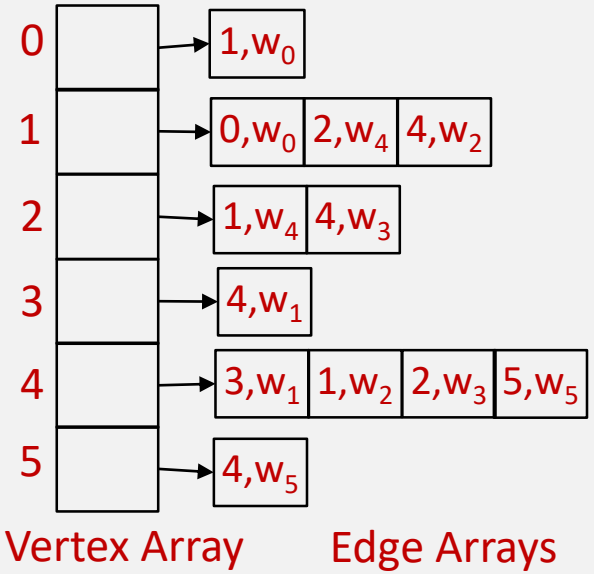


(a) Example graph



(b) Edge List

- **Log:** Captures Temporal Ordering
- Fine-grained Versioning is Free



(c) Adjacency List

- **Indexed Data Structure**
- Coarse-grained Versioning

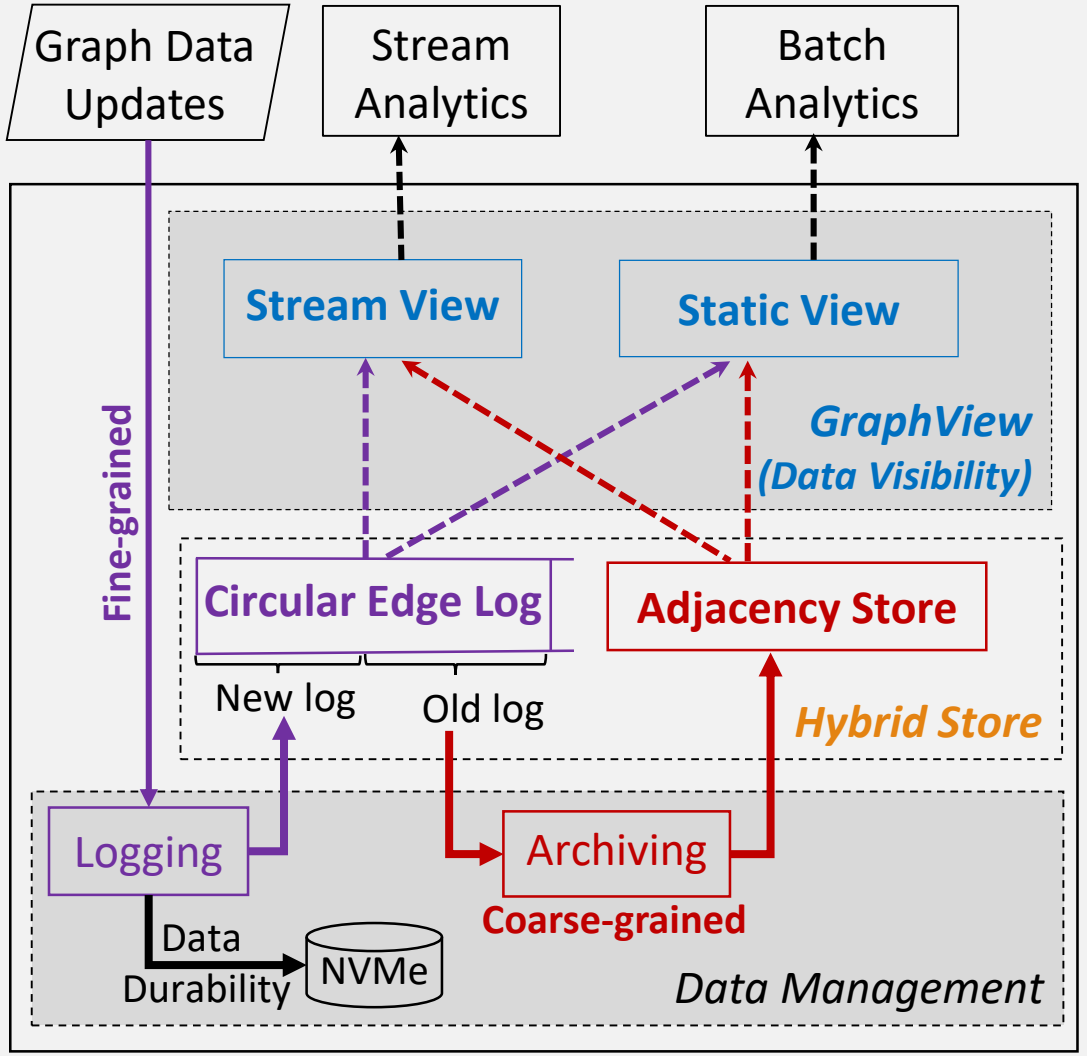
Opportunity: Hybrid Store

Two New Abstractions:

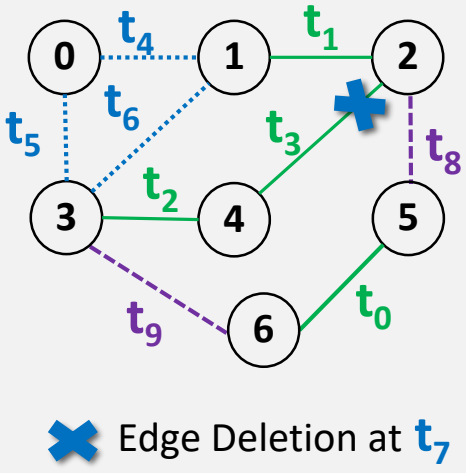
Data Visibility: *Analytics* choose their Ingestion type

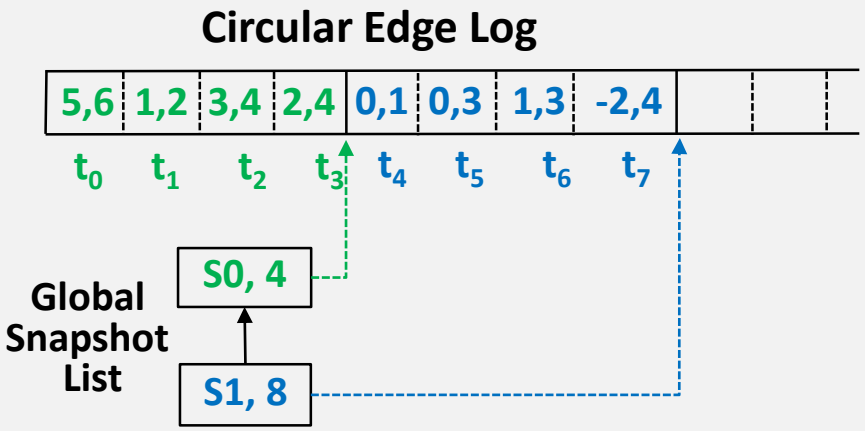
Graph View:

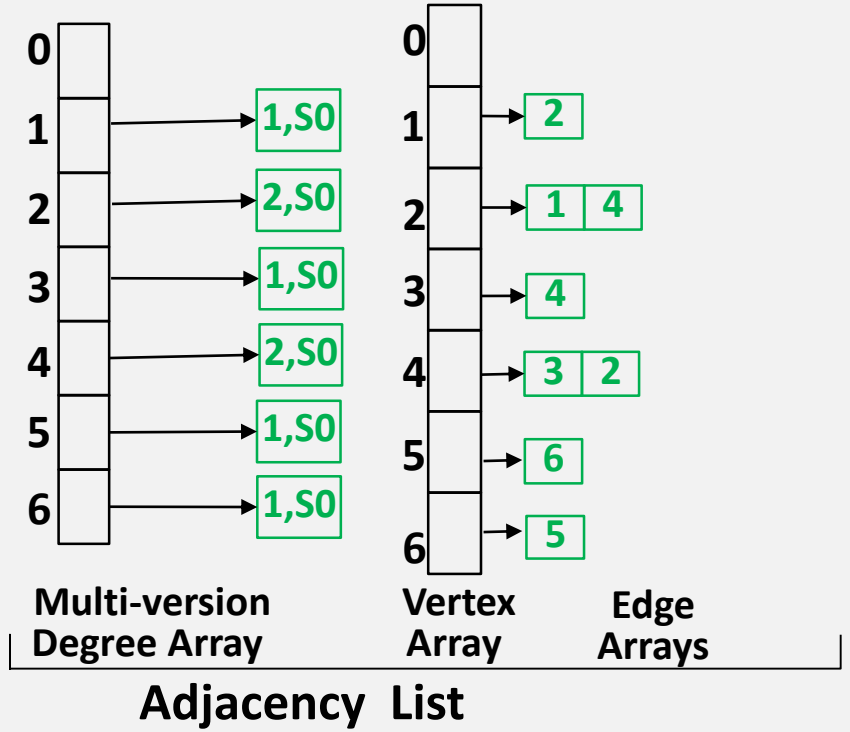
- *Static View:* *Batch Analytics*
- *Stream View:* *Stream Analytics*



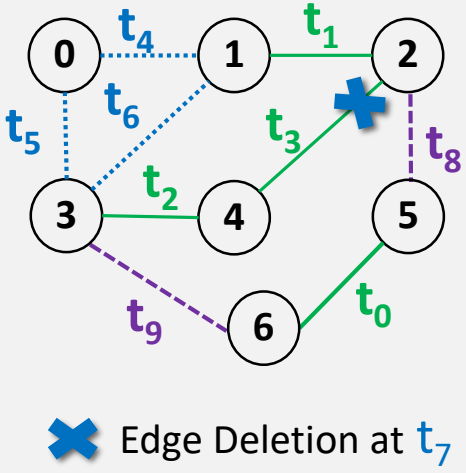
Hybrid Store Details

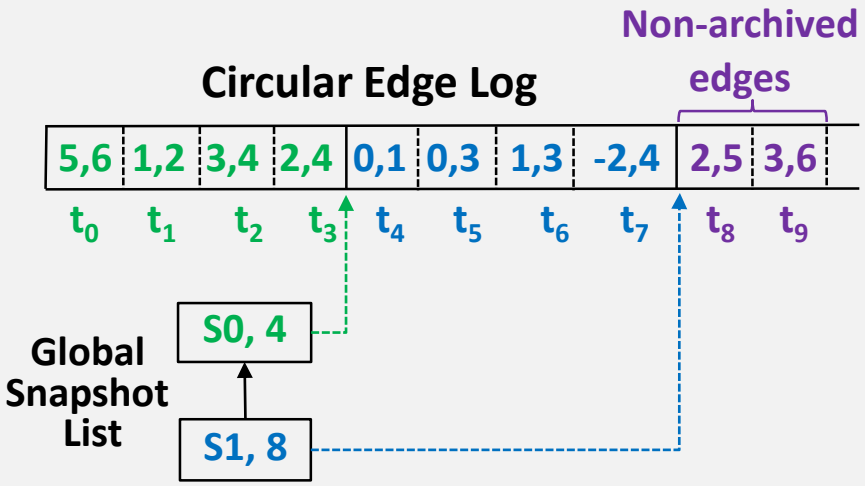


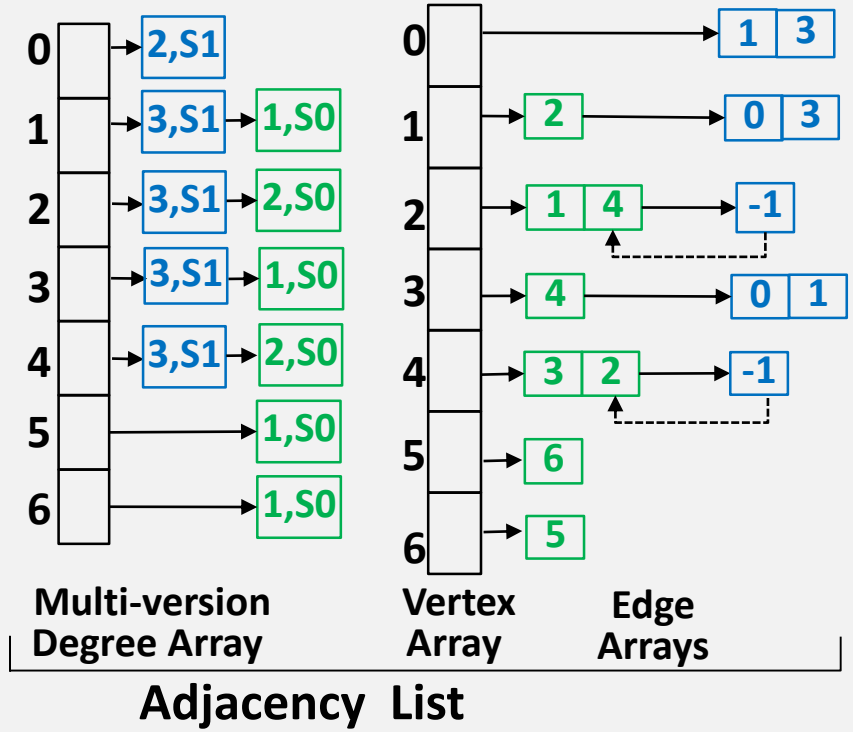


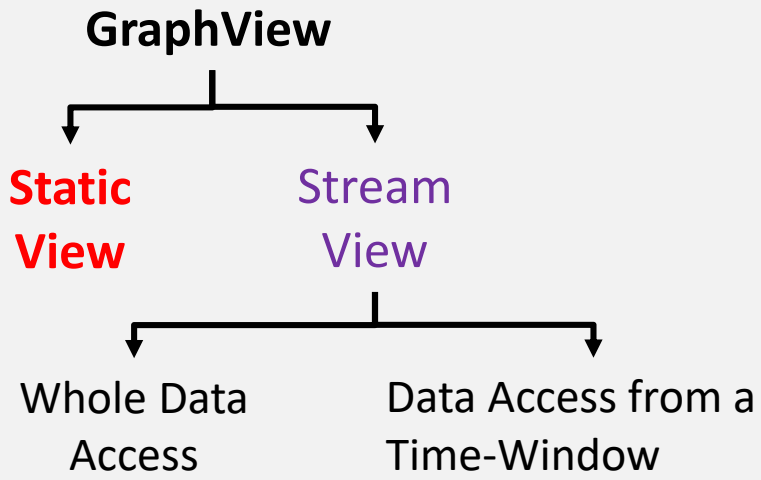


Hybrid Store Details





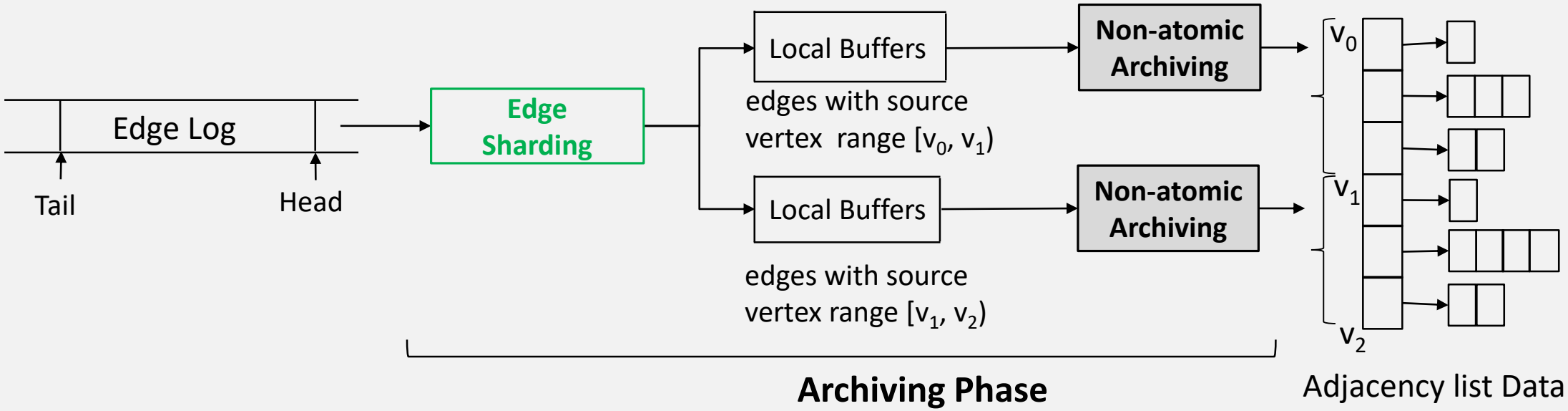




handle **create-static-view**(flags)
status **delete-static-view** (handle)
count **get-nebr-degree**(handle, vertex-id)
count **get-nebrs**(handle, vertex-id, ptr)

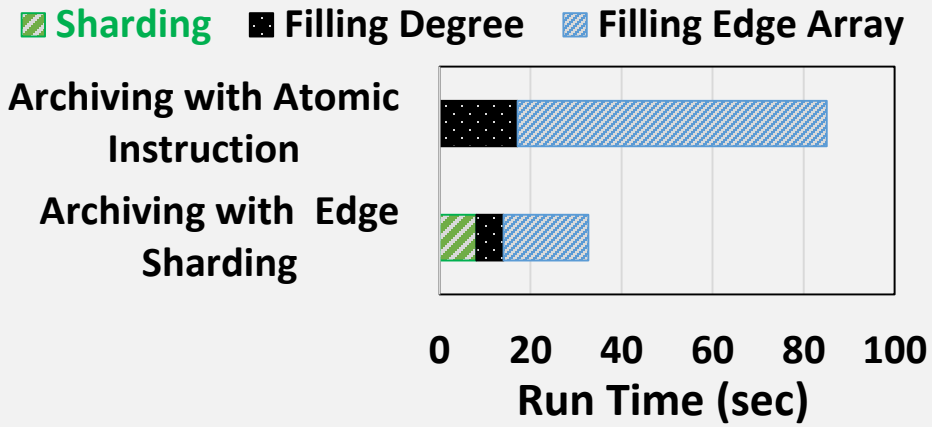
status **update-view**(handle)
bool **has-vertex-changed**(handle, vertex-id)

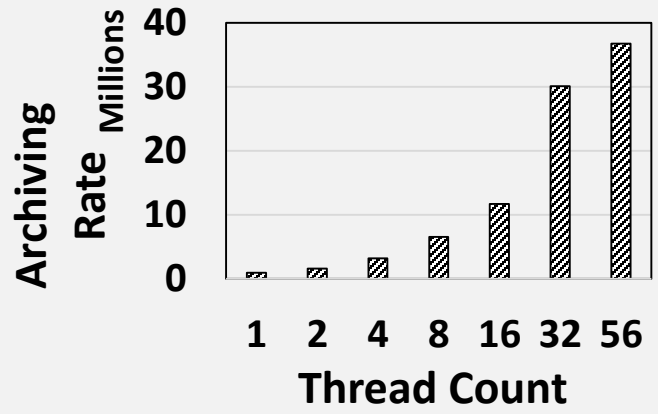
Hybrid Store: Optimizing the Archiving Phase



Machine: 2 Intel Xeon CPU socket
- 14 cores each,
- Hyperthreading enabled

Dataset: Kron-28 Graph
- 256 Million Vertex,
- 4 Billion Edges

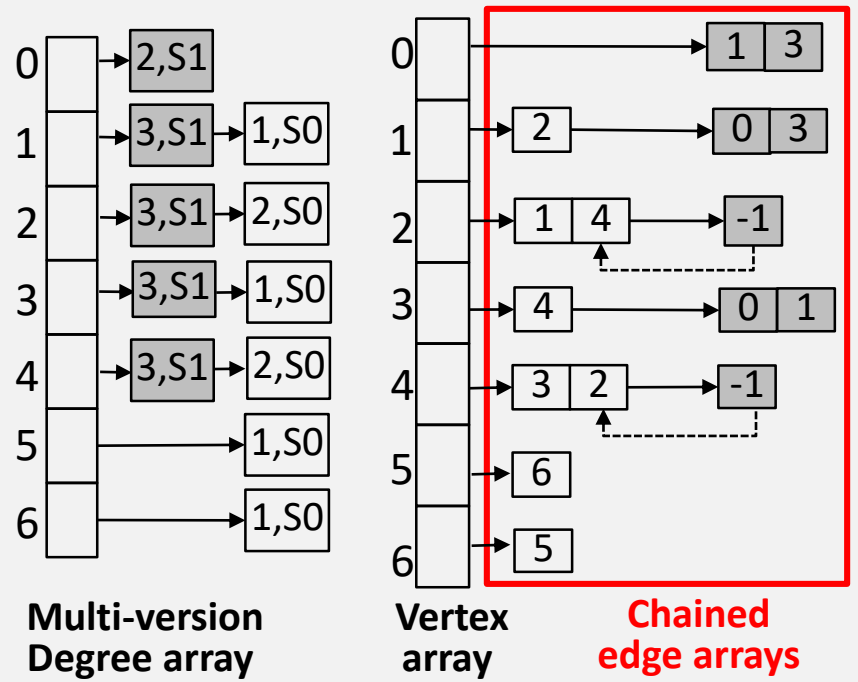




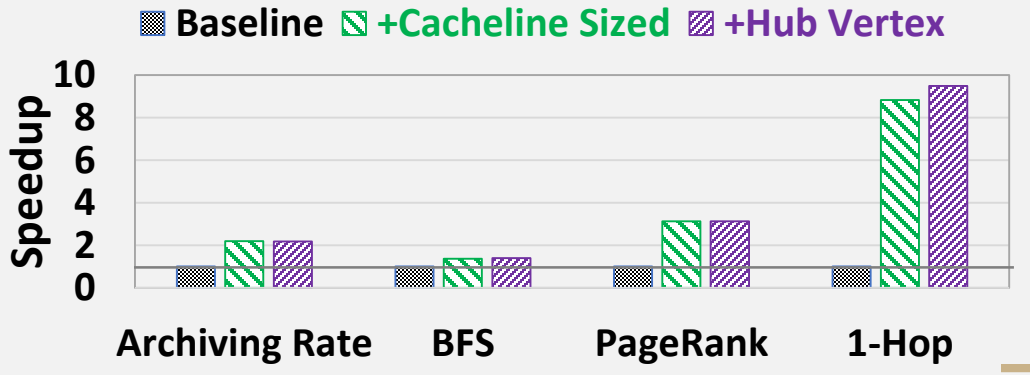
Hybrid Store: Optimizing Memory Usage

- Edge Log : Circular
- Degree Array: Older Versions Reused

- Edge Arrays: Many Memory Links
  - Cacheline Size Memory Allocation
  - Hub Vertex Handling

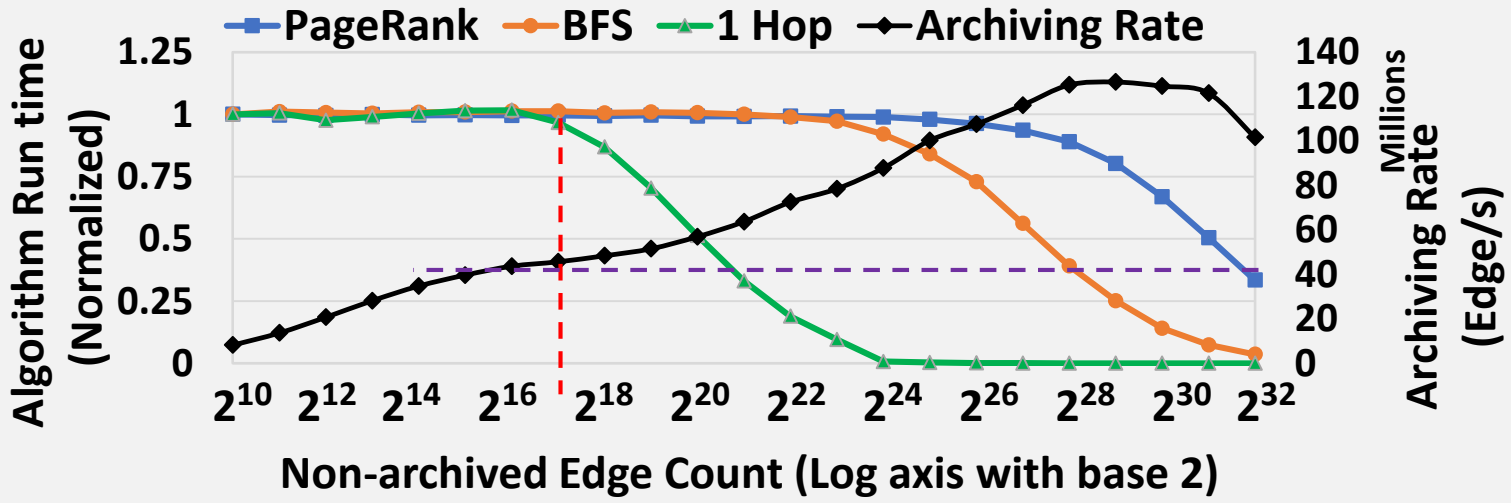


Optimizations	Chain Count		Memory Needed (GB)
	Average	Maximum	
Baseline System	29.18	65536	148.73
Static GraphOne	0.45	1	33.81
+Cacheline Size allocation	2.96	65536	47.42
+Hub vertex Handling	2.47	3998	45.79



Results: Hybrid Store Composition and Ingestion Rate

- Max Non-archived Edge count:
  - 2^{17} edges
- Archiving Threshold:
  - 2^{16} edges
- Archive Rate:
  - 43.68 Million edges/s



- Faster Ingestion?
  - Logging rate:
    - ~80 Million edges/sec
  - Higher archiving threshold

Graph Name	Vertex Count (in Millions)	Edge Count (in Millions)	Rates (Million Edges/s)		
			Logging	Archiving	Ingestion
Twitter (D)	52.58	1963.26	82.62	47.98	66.39
Friendster (D)	68.35	2586.15	82.85	49.32	60.40
Subdomain (D)	101.72	2043.20	82.86	43.43	68.25
Kron-28 (U)	256.00	4096.00	79.23	43.68	52.39
LANL (D)	0.16	1521.19	35.98	28.91	26.99

D = Directed, U = Undirected

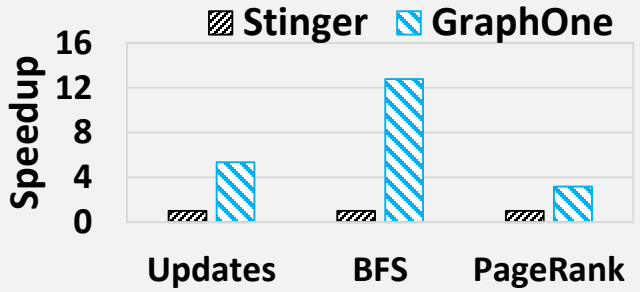
Results: Dynamic Graph Systems

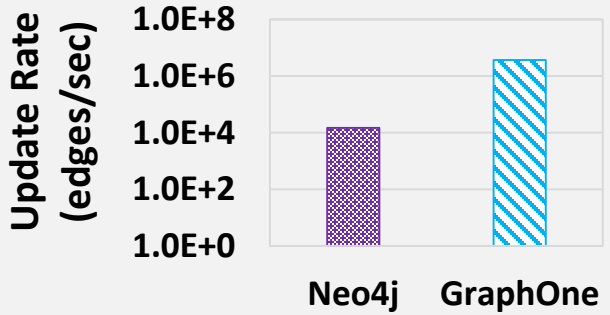
Experimental Setup

- Two Intel Xeon CPU E5-2683 sockets with 14 Core each
- 500GB Memory
- Samsung 950 Pro NVMe SSD, 512GB

Dataset: RMAT Graph

- 4 Million Vertices, 64 Million Edges
- 8 bytes weight
- 40 Million updates, 2.5 Millions deletions





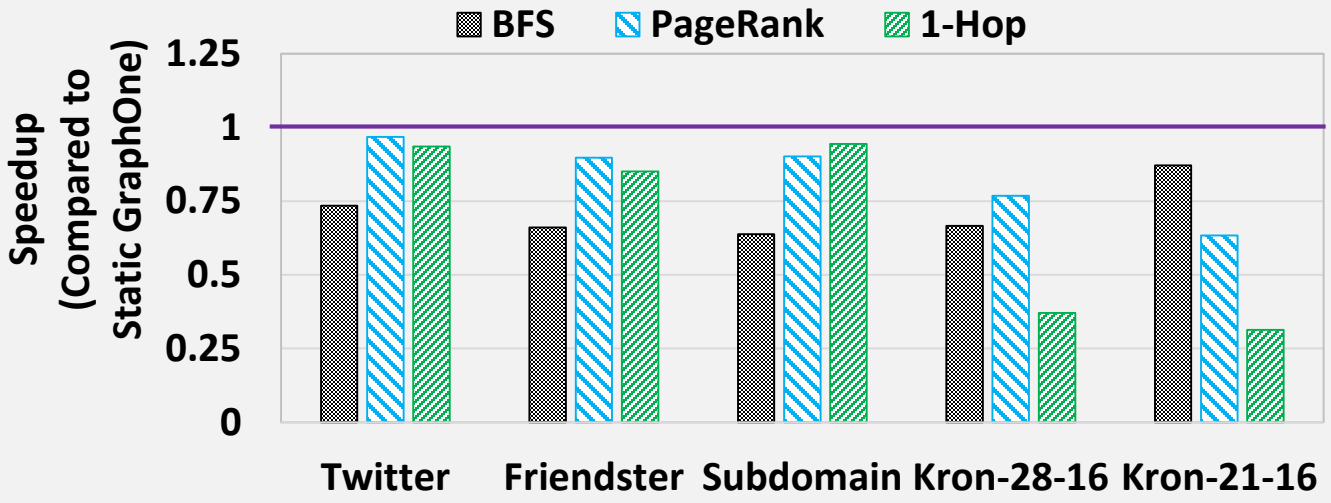
Against Stinger

- 5.36x Higher Ingestion Rate
- 12.76x and 3.18x Speedup for BFS and PageRank

Against SQLite and Neo4j:

- 2 to 3 Orders of Magnitude Higher Ingestion Rate

Results: Cost of Data Management Over Static Graph Engine

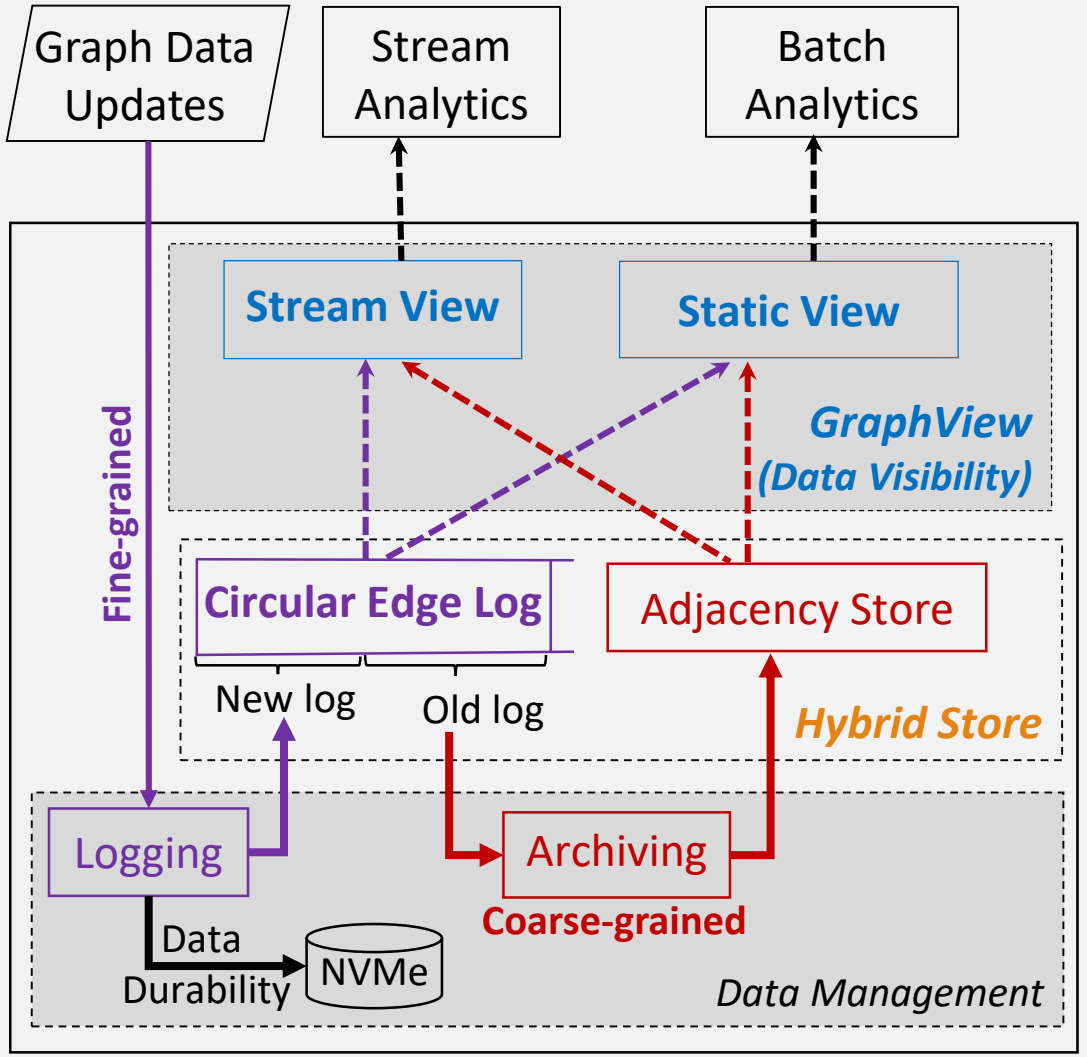


- Static GraphOne: For Static Graphs
  - Builds graph in entirety, Like Galois
- GraphOne: For Dynamic/Streaming Graphs
  - Builds graph one edge at a time

- 17% less performance for real-world graphs
  - Avoids pre-processing cost completely
  - 32.73 second pre-processing cost, 34.12x of BFS

GraphOne: Conclusion

- Batch and Stream Analytics on Evolving Graphs
- Hybrid Store: Two Abstractions
  - Data Visibility and GraphView
- Optimizations:
  - Edge Sharding and Memory Allocation
- **Results:**
  - Over **Neo4j and SQLite**: 2 to 3 orders of higher ingestion rate
  - Over **Stinger**: 5.36x more ingestion rate, over 3x speedup for analytics
  - Over **Static Graph Engine**: No Pre-processing Cost
- **A Real-System** you can use
  - Can convert text/logs/csv/json to graph



usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Thank You

pradeepk@email.gwu.edu