Kineograph: Taking the Pulse of a Fast-Changing and Connected World

EuroSys '12

Zuhair Khayyat

What is Kineograph

- A distributed system for processing a continuously changing graph.
- Input: stream of incoming data
- Main contribution: Distributed in-memory storage that supports:
  - Incremental graph algorithms that extracts important information on the fast changing graph structure.
  - Maintain consistency for static structures graph algorithms.

Applications on top of Kinograph

- Time sensitive applications: Trends & breaking news
- Small data rich connectivity
- Input data: Twitter feed including user interactions and hashtags
- Algorithms:
  - User ranking
  - Approximate shortest paths
  - Controversial topic detection
- Claim that Kinograph can process 100K tweets per second on 40 machine cluster

Application Challenges

- Handling fast updates and produce timely results.
- Distributed flexible graph structure to capture relations.
- Handle updates and static algorithms consistency.
- Appropriate fault tolerance.

Kineograph design

- Separate graph processing and graph updates
- Provide consistent periodic graph snapshots
- Distributed updates: multiple workers (Ingest nodes) receives twitter feed.
- Updates to the graph are transformed into sequenced transactions.

System Overview

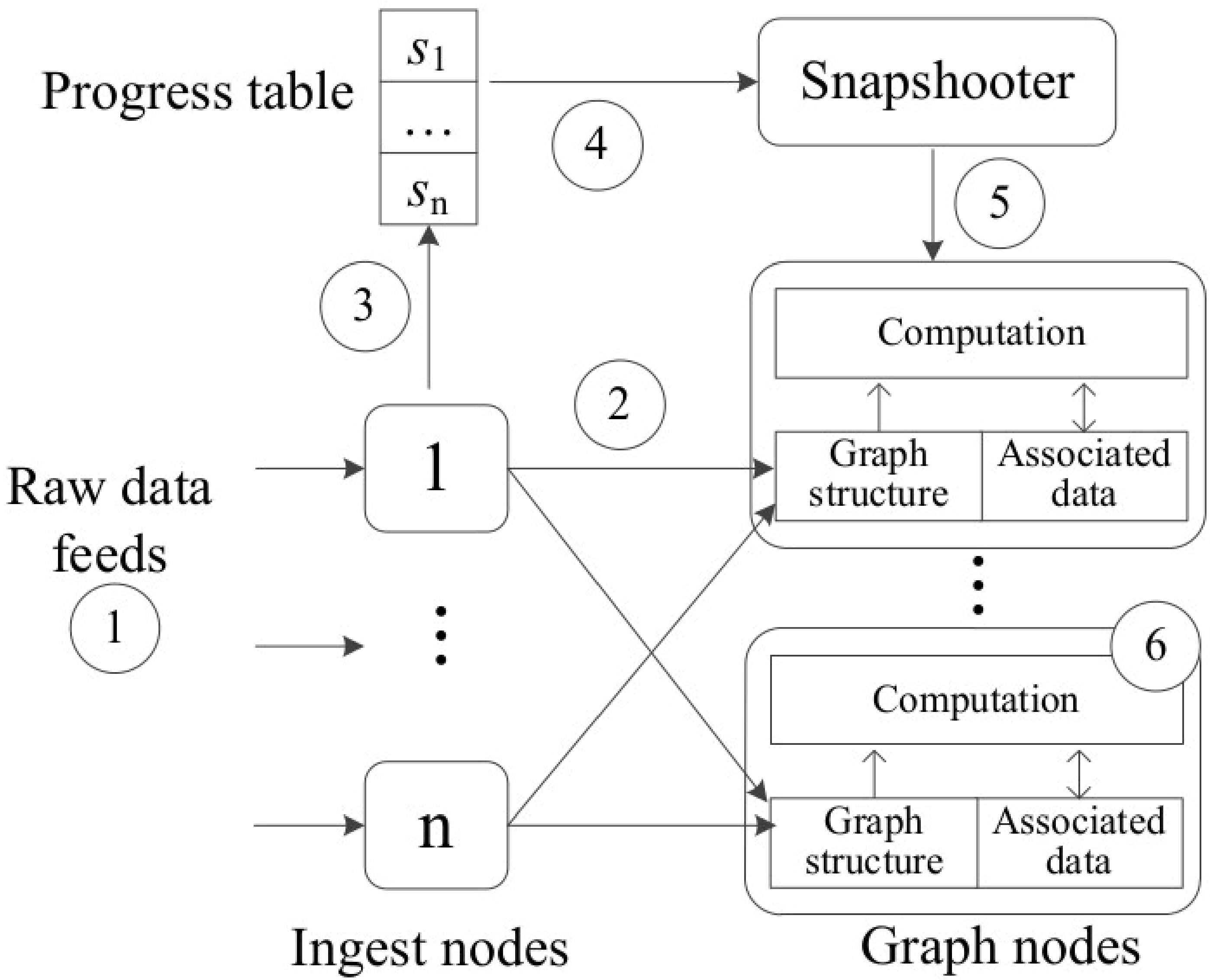


Figure 1. System overview.

Consistent Distributed snapshots

- Key/Value based storage system
- Graph partition based on hashing, multiple partitions on a physical machine
- An incoming record is converted to an update transaction which can span to multiple partitions.
- Each transaction has a unique sequence number, which are used for system synchronization (global logical timestamp)
- The global progress table keep track of updates
- Snapshot and applying update transactions are overlapped.

Example

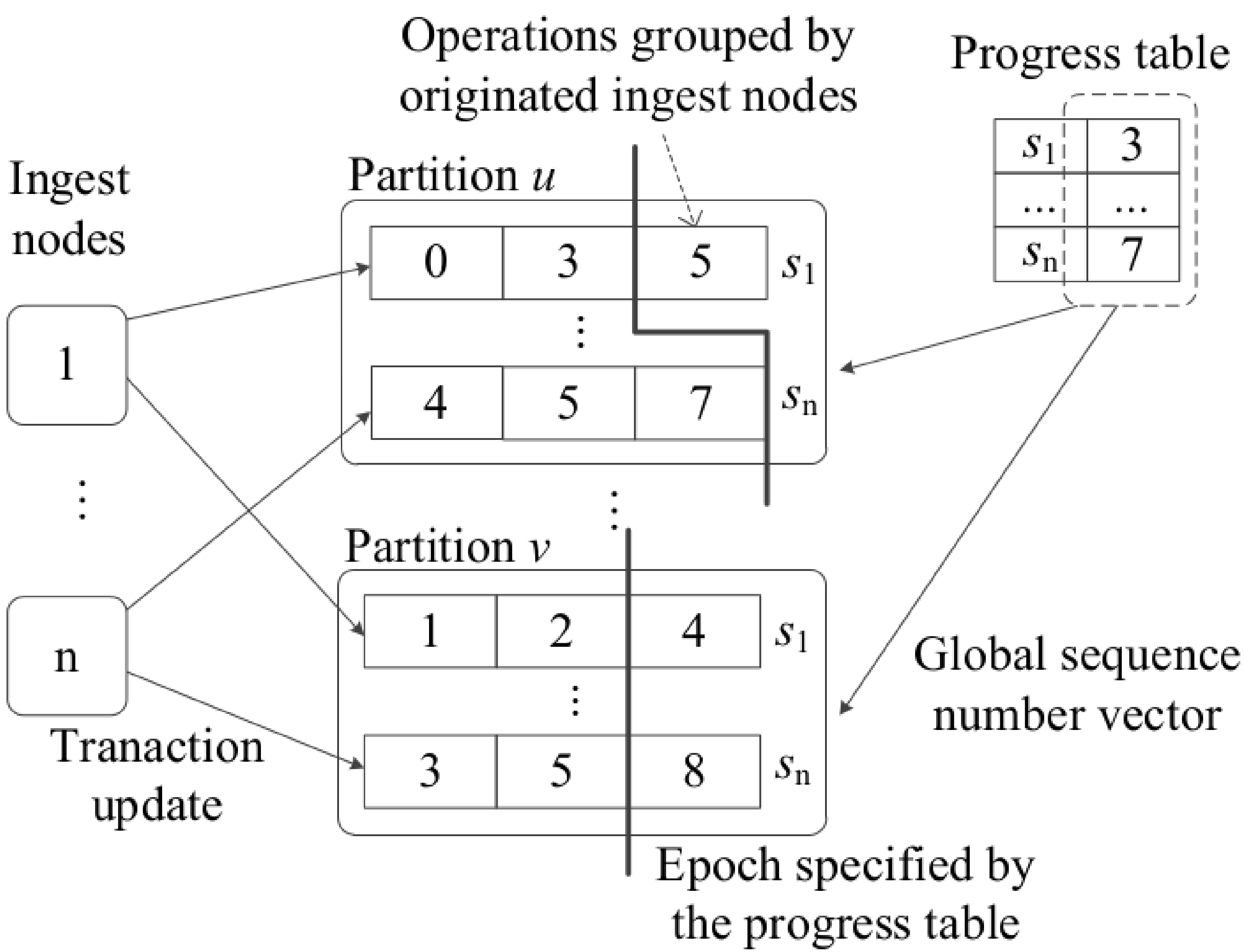
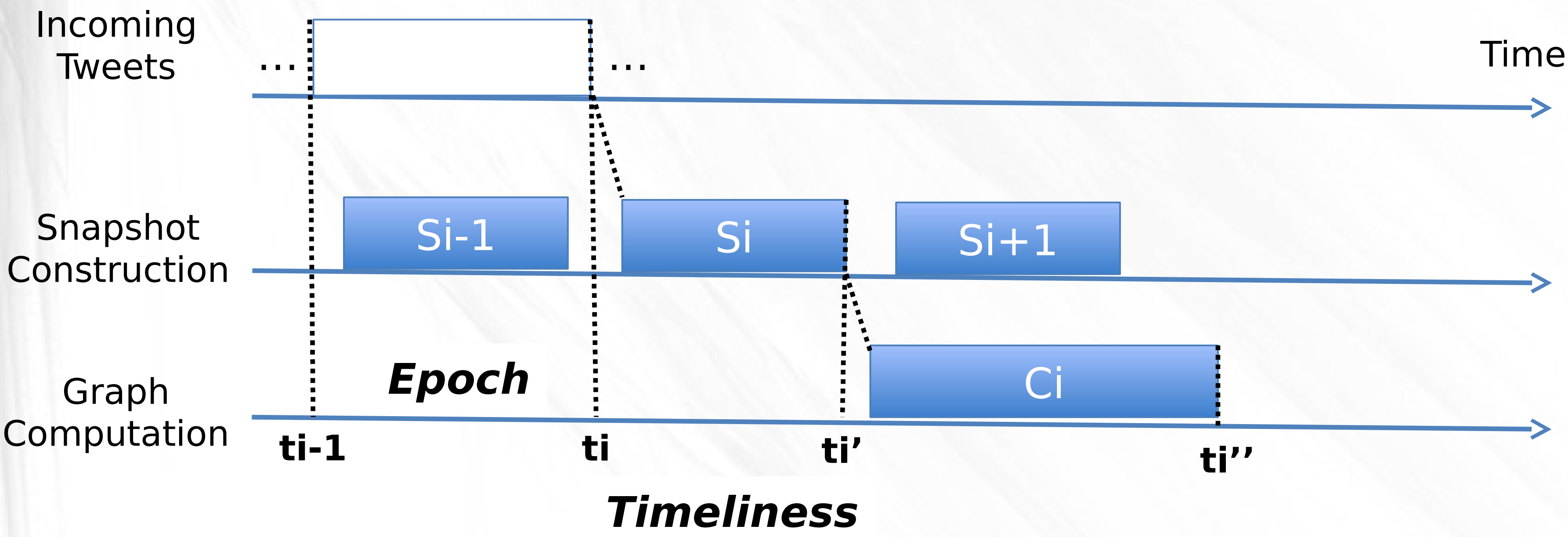


Figure 2. An example of creating a consistent snapshot across partition u and v.

Snapshot consistency

- Ensure atomic transactions



Incremental computations

- The user provides:
  - Rule to check if vertex changed
  - User function to compute new values
  - Support aggregations, push and pull
- No need for consistency model.
- Vertex-centric computations.
- Supports supersteps like GraphLab and Pregel
- Functions:
  - Initialize, updateFunction, trigger, accululator, vertex-request

Example

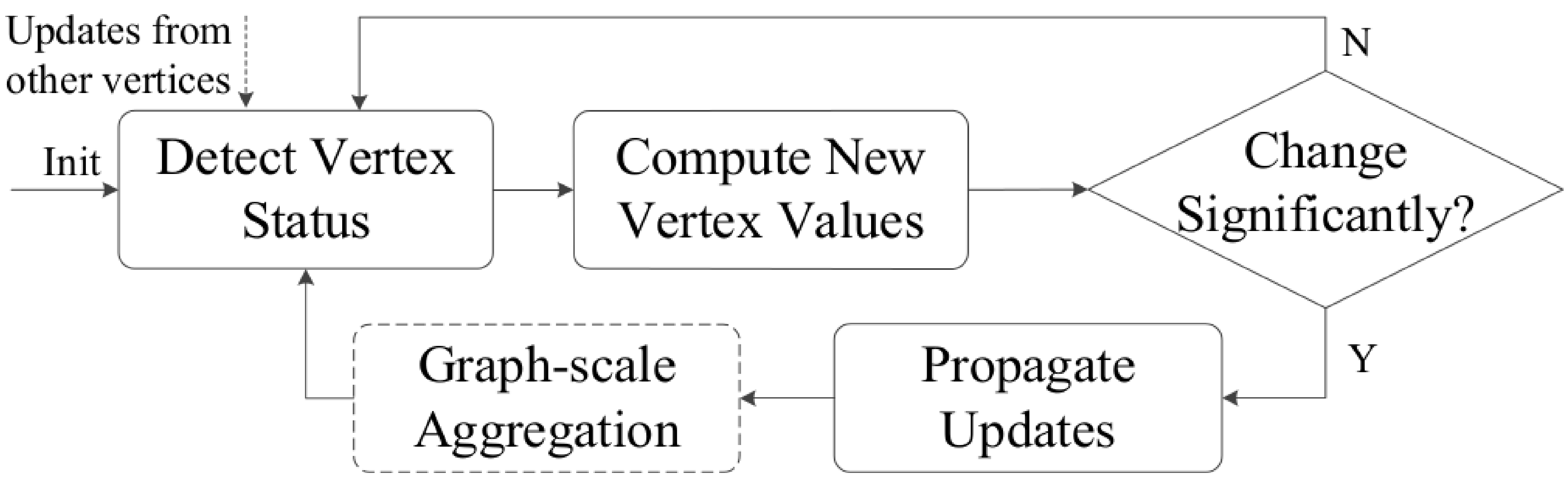


Figure 3. Computation overview.

Fault Tolerance

- Future: Chubby or ZooKeeper for global progress table and monitoring
- Feeds are stored reliably.
- Each graph partition has k replicas in different physical machines, graph updates are sent to all replicas.
- The result of the computation is stored in a master/slave model.

Evaluation

- Code: C# on windows server 2008
- Cluster size: 51 machines
- Combination of machines:
  - Intel Xeon X3360 quad CPU, 8GB memory
  - Intel Xeon X5550 quad CPU, 12 GB memory
- Gigabit network
- Graph Size: 8M nodes & 29M edges
- Simulated a live Twitter stream of 100 million archived tweets.

Graph update throughput

- The fastest peak amount of Twitter traffic was 8900 tweets/s on October 2011.
- Kineograph: Up to 180000 tweet/s

Snapshot Interval(s)	SCT Max/Avg	Avg SCT(s)	Throughput(t/s)
10	3.1	1.9	137.6k
30	2.2	4.4	143.0k
60	1.9	8.4	150.8k

Table 2. The impact of transient imbalance on throughput under K-Exposure, with 8 ingest nodes, 32 graph nodes. SCT Max/Avg: The ratio of maximum over average Snapshot Construction Time.

Throughput vs ingest nodes

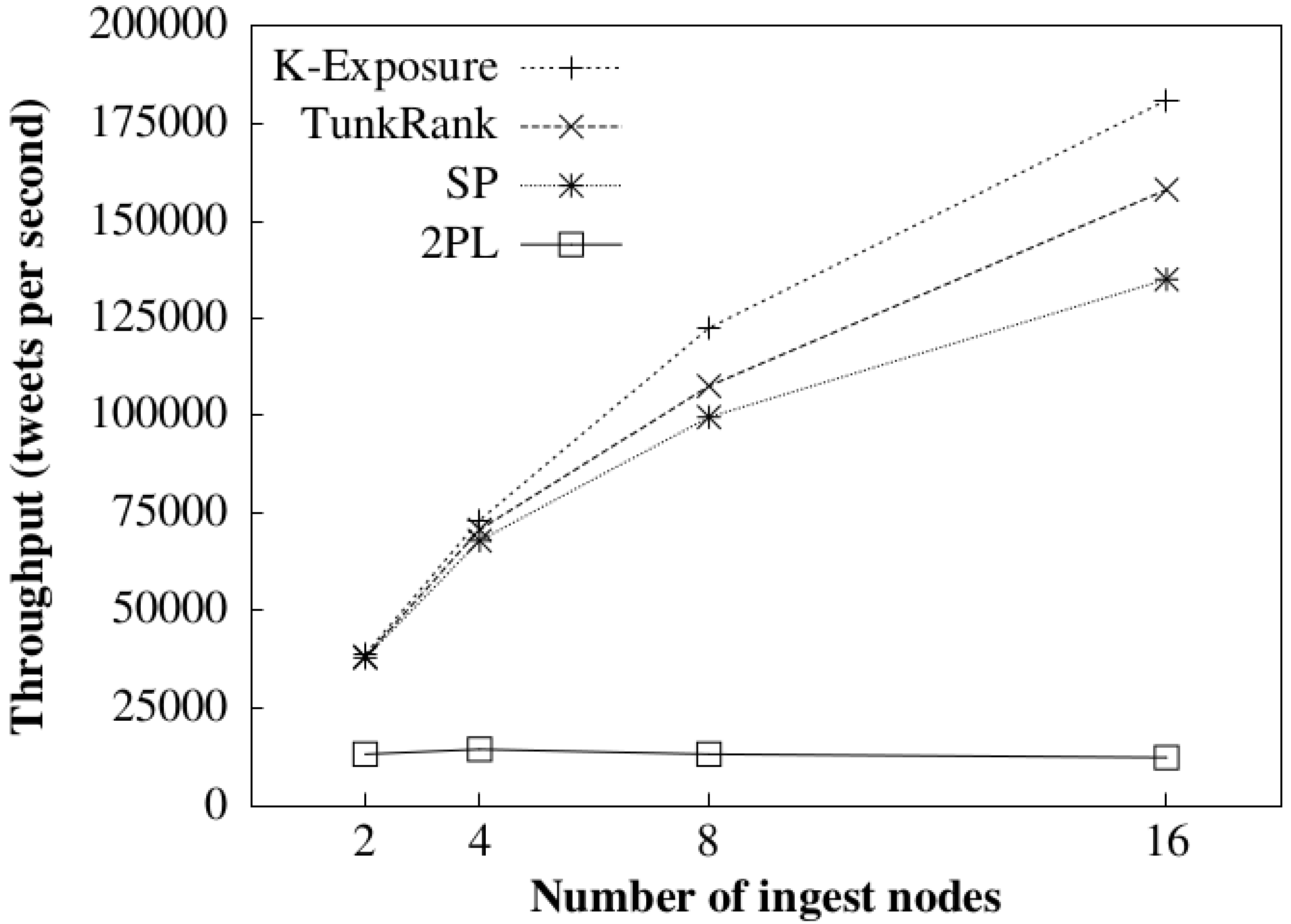


Figure 9. Graph-update throughput on 32 graph nodes with varying numbers of ingest nodes and with different applications. Snapshot interval is set to 10 seconds.

Scalability

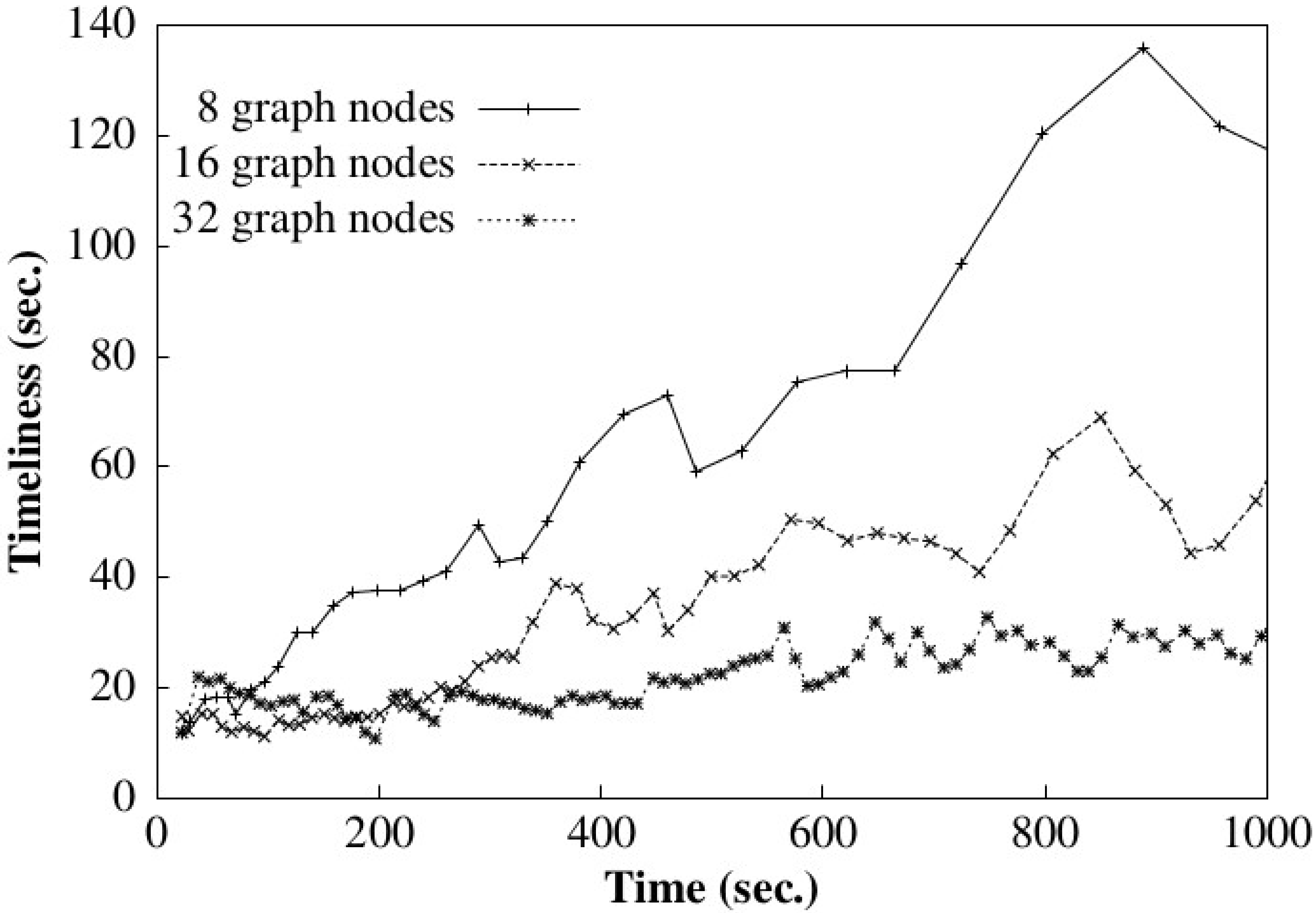


Figure 13. Scalability of TunkRank with different numbers of graph nodes and 2 ingest nodes.

Kineograph vs Pregel

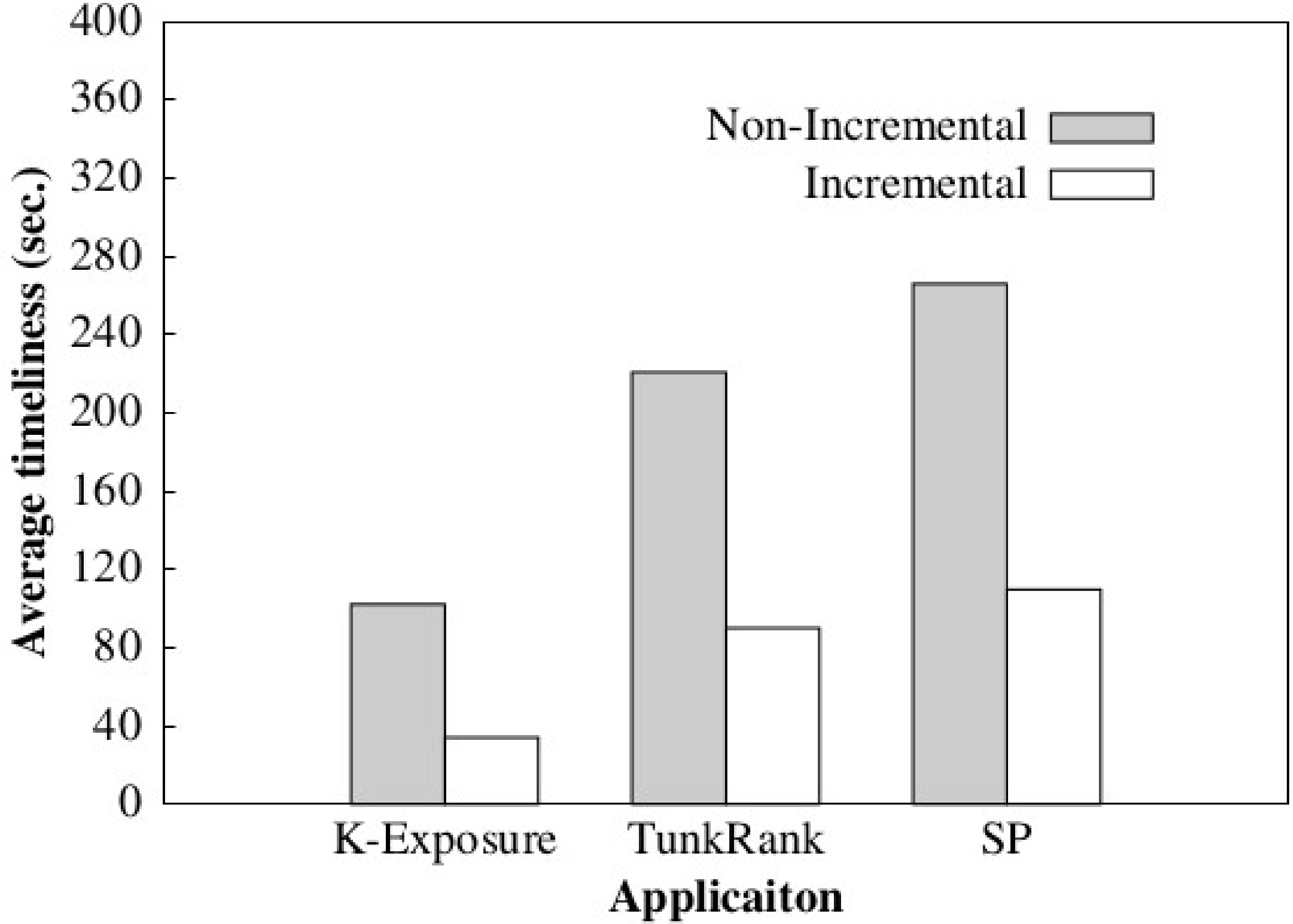


Figure 12. Average timeliness improvement of incremental applications under 4 ingest nodes and 32 graph nodes.

Questions