Benchmarking High Performance Graph Analysis Systems with Graph Mining and Pattern Matching Workloads

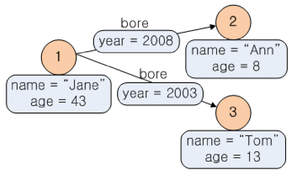
Seokyong Hong[§], Seung-Hwan Lim, Sangkeun Lee, Sreenivas R. Sukumar,, and Ranga R. Vatsavai[§]
North Carolina State University[§]
Oak Ridge National Laboratory

Problem

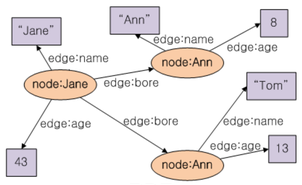
❖Different Data Model

	0	1	...	N
0	0	1	0	0
1	0	1	0	1
...	1	0	0	0
M	0	0	0	0

Matrix



Property Graph



RDF

❖Various Graph Processing Systems

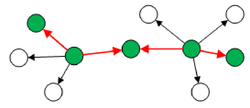
Project Pegasus · Titan:db by Aurelius · Spark · neo4j · uRiKA

❖Different Querying Interfaces

W3C SPARQL · Gremlin $G=(V,E)$ · python · Scala · neo4j Cypher

❖Different Graph Analysis Workloads

Graph Mining



Subgraph Pattern Matching

- **Which systems deal with which workloads better or worse?**
- **What is the advantages/disadvantages of each systems?**
- **What is the best query languages to describe graph analysis tasks?**

Graph Analysis Workloads

❖Datasets

Graph Mining	Graph Pattern Matching
SNAP Datasets ●DBLP (0.3 million vertices, 1 million edges) ●Patents (3.8 million vertices, 16.5 million edges) ●Friendster (65.6 million vertices, 1.81 billion edges)	**Extended LUBM Benchmark Datasets** ●100 Universities (2.1 million vertices, 6.6 million edges) ●1,000 Universities (21.7 million vertices, 66.2 million edges) ●10,000 Universities (0.2 billion vertices, 0.6 billion edges)

❖Benchmark Queries

Graph Mining	Graph Pattern Matching
Graph Mining Operations ●Degree Distribution (non-iterative) ●Connected Components (iterative) ●PageRank (iterative)	**LUBM Benchmark Queries[*]** ●Query6 ●Query7 ●Query9

Benchmarked Systems

Graph Mining				Graph Pattern Matching			
System	**Processing Platform**	**Graph Model**	**Query Interface**	**System**	**Processing Platform**	**Graph Model**	**Query Interface**
Pegasus	MapReduce	Matrix	Java/GIM-V	**Titan**	Titan/HBase	Property Graph	Gremlin
GraphX	Spark	Property Graph	Scala/Pregel	**GraphX**	Spark	Property Graph	Scala/GraphX Library
uRiKA	Cray XMT	RDF	SPARQL	**uRiKA**	Cray XMT	RDF	SPARQL

Evaluation Environment

Graph Mining			Graph Pattern Matching		
System	**Version**	**Environment**	**System**	**Version**	**Environment**
Pegasus	Hadoop 2.4.0	65 AWS m3.2xlarge (64 compute nodes)	**Titan**	Titan 0.5.1 HBase 1.0	9 32-core, 64G-RAM, 500GB-disk nodes (8 compute nodes)
GraphX	Spark 1.0.0	65 AWS m3.2xlarge (64 compute nodes)	**GraphX**	Spark 1.3.1	9 32-core, 64G-RAM, 500GB-disk nodes (8 compute nodes)
uRiKA	-	ORNL 2TB shared memory	**uRiKA**	-	ORNL 2TB shared memory

Comparisons and Conclusion

❖Performance

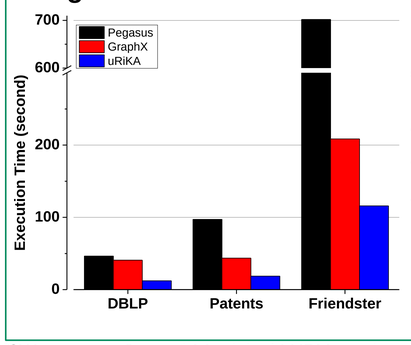
Graph Mining	Graph Pattern Matching
●GraphX showed the best performance for iterative processing. ●uRiKA showed the best performance for non-iterative processing (Degree Distribution) and good performance with small and moderate size graphs. ●Pegasus showed the worst performance in most cases.	●uRiKA showed the best performance in most cases. ●GraphX showed fairly good performance while it could not handle 10K universities on our nine-node cluster. ●Titan/HBase showed poor performance compared to the others.

❖Querying Interfaces and APIs

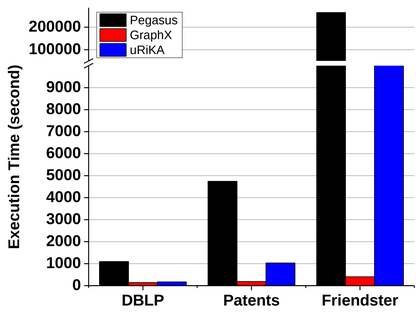
System	Description
Pegasus	●It provides a set of matrix multiplication operations which enable graph mining tasks. ●With those operations, it is not straightforward to describe pattern matching tasks.
GraphX	●It provides the BSP processing model with related primitives along with optimized graph mining operations which allow graph mining operations to be described effectively. ●A set of operations (e.g., Triplet and Join) provides a good way to describe pattern matching tasks.
Titan	●Its default query language, Gremlin, provides an intuitive way to write graph traversal queries but it has limitations on describing pattern matching queries, which did not allow us to write LUBM query 9.
uRiKA	●Its query language, SPARQL, does not provide a way to describe iterative mining tasks in a single query but several wrapper languages (Python and Java) can be used for that. ●SPARQL provides a fairly easy and intuitive way to describe pattern matching tasks.

Graph Mining Result

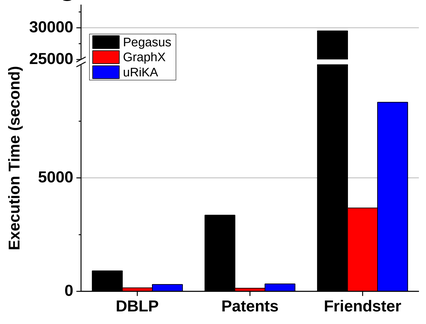
❖Degree Distribution



❖Connected Components

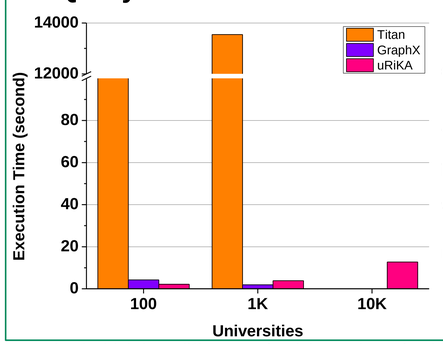


❖PageRank

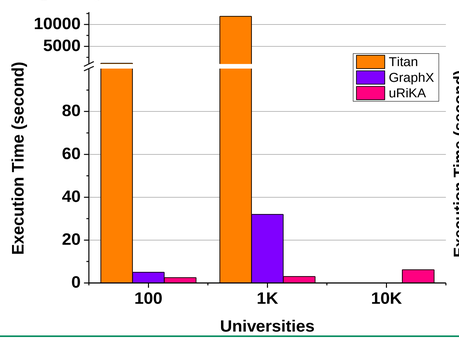


Pattern Matching Result[*]

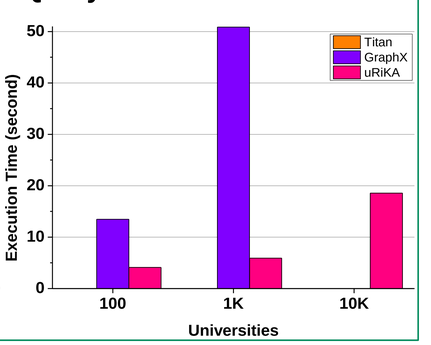
❖Query6



❖Query7



❖Query9



[*] We re-wrote 9 out of 14 original LUBM queries and tested all the nine queries. Due to the page limitation, we presented the execution times of three queries each of which has different selectivity and a selectivity pattern as graph size increases.

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Seokyong Hong (shong3@ncsu.edu)
Department of Compuater Science, North Carolina State University