XGDBench: A Benchmarking Platform for Graph Stores in Exascale Clouds

Miyuru Dayarathna and Toyotaro Suzumura

Suzumura Laboratory
Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology

5/12/2012

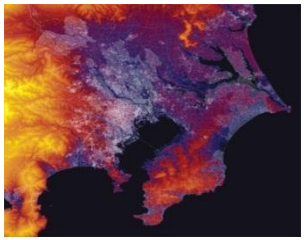
Introduction – Graph Data Analysis on Cloud

- Graphs have become an important workload in cloud systems.

1000 Genomes Project

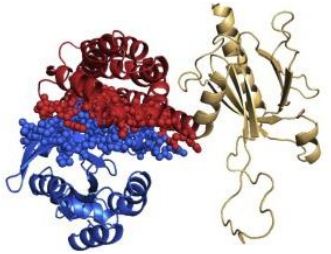
1000 Genomes Project reveals human variation (L. WILLATT, EAST ANGLIAN REGIONAL GENETICS SERVICE / SCIENCE PHOTO LIBRARY)

Open Street Map (Tokyo, Japan)



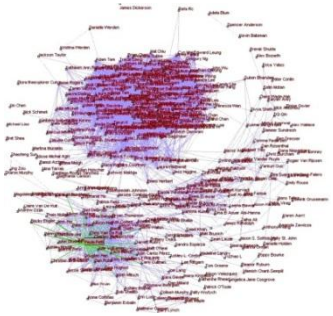
A visualization of Tokyo using the SRTM (Shuttle Radar Topography Mission) and Open Street Map (Ted Ngai)

Protein-Protein interaction



A structural protein-protein interaction (Christof Winter et al)

Social Network

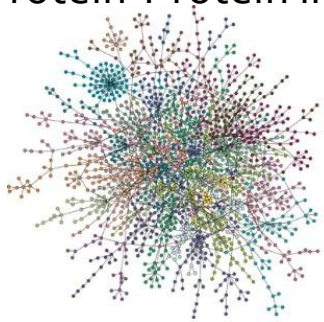


Graph of Dina Westland 385 friend's connections from Facebook, J. Christopher Westland, 2009

Communication Network

A node-link network disgram visualization of Twitter users, Derek L. Hansen et al, 2011

Protein-Protein interaction

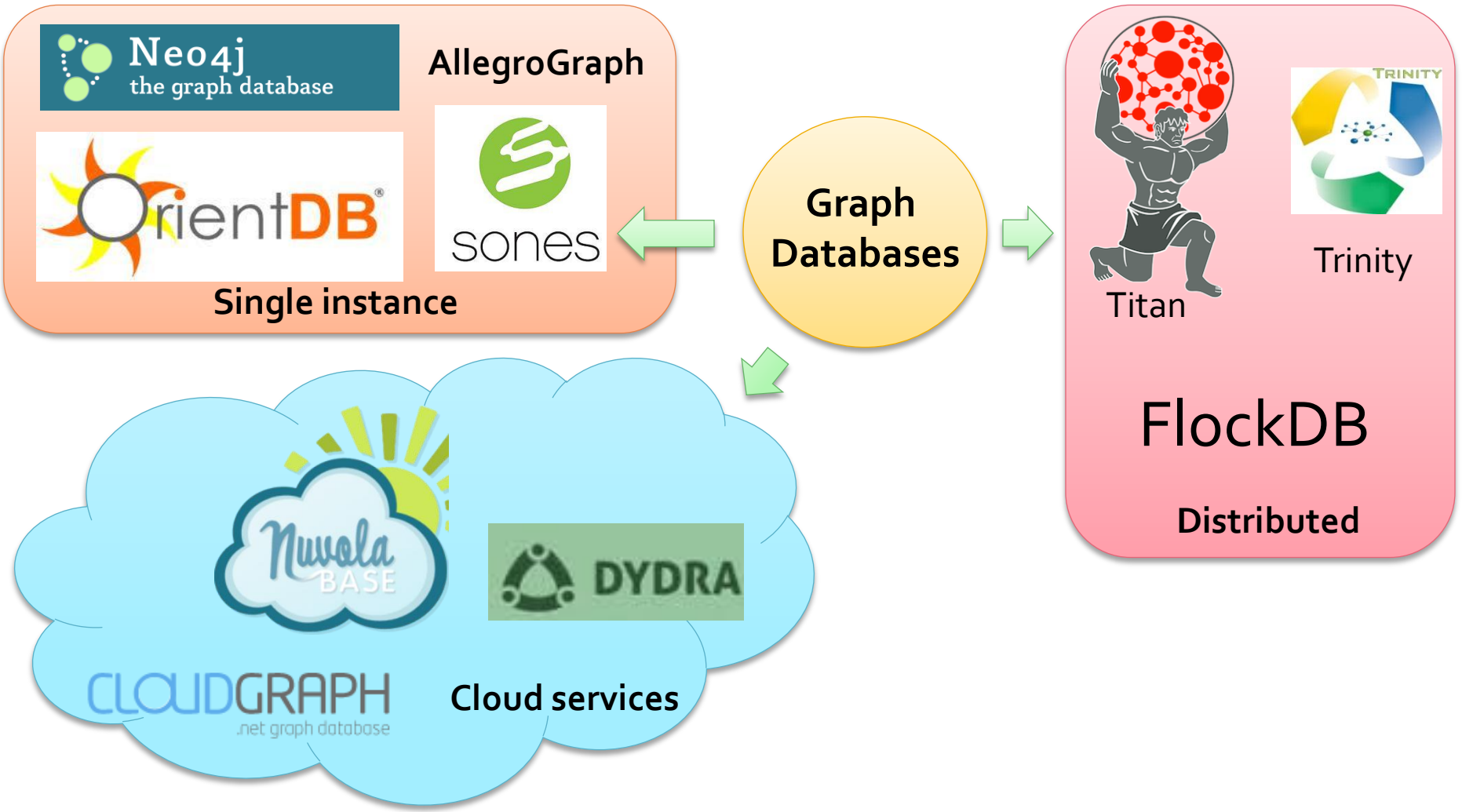


Network of protein interactions in yeast (Roger Guimerà at al, 2006)

Introduction – Graph Database Models

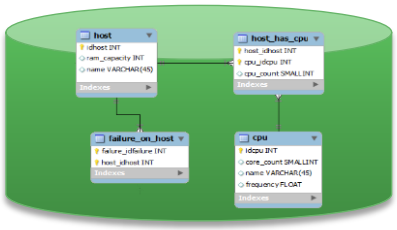
- Relational databases have been used to store graph data.
- Graph data storage and analysis in the form of graphs is more effective.
  - Optimized performance
  - Query productivity
- Many commercial and open source graph databases have appeared recently.
- Graph database services on cloud

Introduction – Graph Database Models

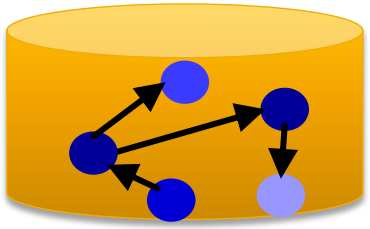


Introduction – Graph Databases

- A type of No-SQL Databases
- Follow network data model



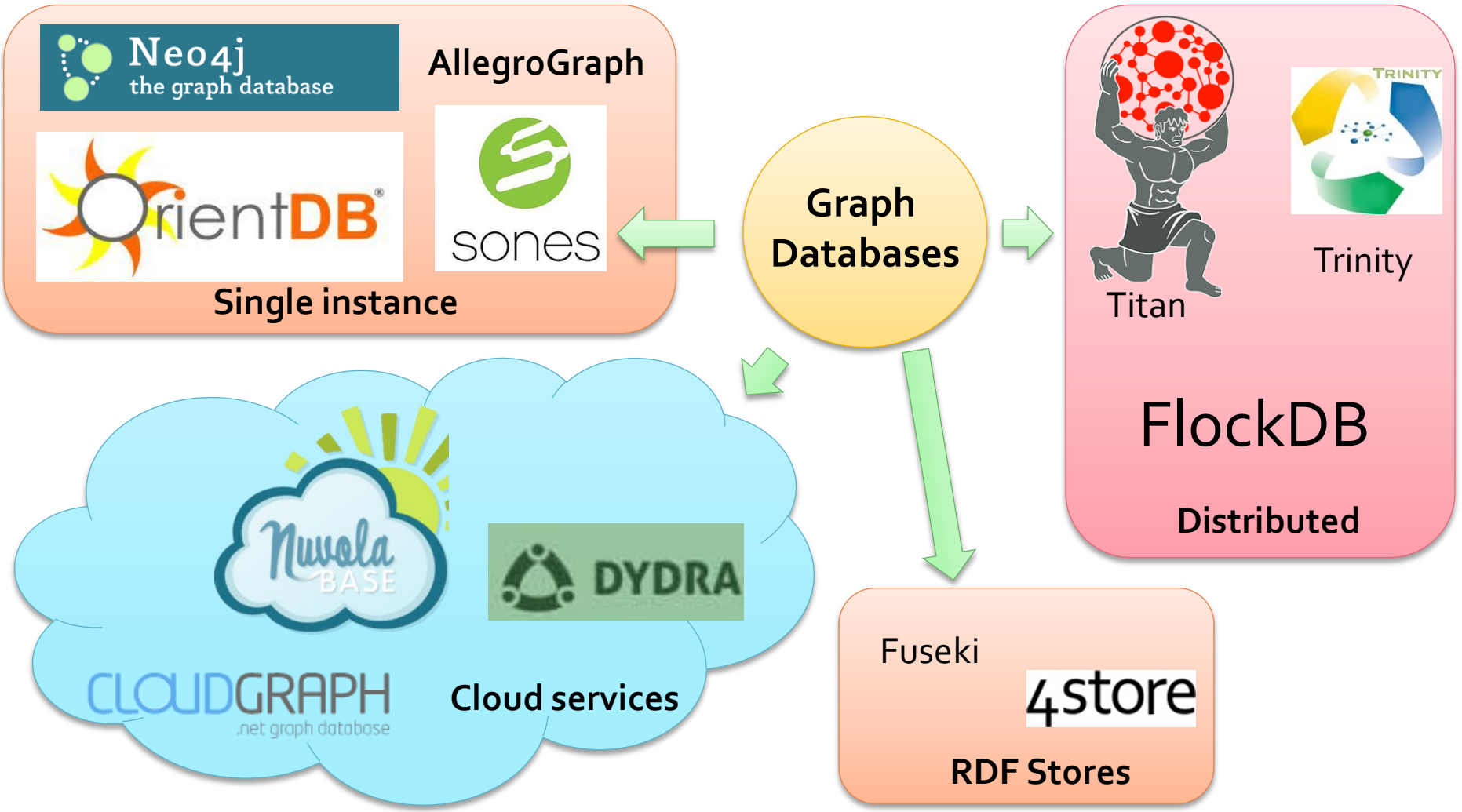
Relational Database



Graph Database

- Have close similarity to RDF (Resource Description Framework) stores
  - RDF triples can represent vertices and edges between them.

Introduction – Graph Database Models



Presentation Outline

- **Introduction**
- **Research Problem** (Graph Database Benchmarking)
- **Proposed Solution** (XGDBench)
- **Related Work**
- **Methodology**
- **Evaluation**
- **Conclusion**

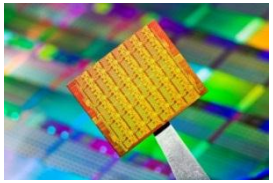
Research Problem – Graph Database Benchmarks

- Does not realistically model real application scenarios.
  - HPC Scalable Graph Analysis Benchmark
    - Focus on some core network analysis features
- Do not follow a statistical model, hence they are not smoothly scalable.
  - Benchmarks from Semantic web
    - LUMB, SP2Bench, DBPedia

Exascale Clouds

- Dawn of Exascale computing →2018~2020.
- Power efficient cloud computing systems with huge performance per watt values.
- Completely new programming techniques and models are needed.
- Partitioned Global Address Space (PGAS) languages is one approach for programming such systems.

Intel Xeon Phi



Single-chip Cloud Computer (SCC) of Intel contains 48 P54C Pentium cores (Intel ,2009)

NVIDIA Tesla

PGAS model for Cloud computing systems.

Proposed Solution and Contributions

- A benchmarking platform for graph databases in Exascale systems.
  1. Graph database benchmarking platform
  2. Benchmarking Exascale Clouds
  3. Workload characterization of graph databases
     - AllegroGraph, Fuseki, Neo4j, and OrientDB

Related Work

- HPC Scalable Graph Analysis benchmark [2]
  - Does not evaluate features such as object labeling, attribute management, etc.
- A benchmark for graph traversal operations in graph databases [23]
  - Our work is purely based on graph database servers

[27] R. Nambiar, N. Wakou, F. Carman, and M. Majdalany. Transaction processing performance council (tpc): State of the council 2010. In R. Nambiar and M. Poess, editors, Performance Evaluation, Measurement and Characterization of Complex Systems, volume 6417 of Lecture Notes in Computer Science, pages 1–9. Springer Berlin / Heidelberg, 2011.

[2] D. A. Bader, J. Feo, J. Gilbert, J. Kepner, D. Koester, E. Loh, K. Madduri, B. Mann, T. Meuse, and E. Robinson. Hpc scalable graph analysis benchmark. Feb 2009.

[23] L. H. Marek Ciglan, Alex Averbuch. Benchmarking traversal operations over graph databases. In Proceedings of the 3rd International Workshop on Graph Data Management: Techniques and Applications, GDM '12, 2012.

Related Work (Cont.)

- Benchmarks for Semantic data stores (LUBM [17], Berlin [3], DBpedia [24])
  - Does not use a statistical graph generator model
- Graph 500
  - Benchmark for data intensive supercomputing applications.
  - Current implementation does not consider applications such as graph databases.

[27] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web, 3(23):158 – 182, 2005.

[3] C. Bizer and A. Schultz. The berlin sparql benchmark. International Journal On Semantic Web and Information Systems, 2009.

[24] M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. Dbpedia sparql benchmark - performance assessment with real queries on real data. In International Semantic Web Conference (1)'11, pages 454–469, 2011.

[27] R. Murphy, J. Berry, W. McLendon, B. Hendrickson, D. Gregor, and A. Lumsdaine. Dfs: A simple to write yet difficult to execute benchmark. In Workload Characterization, 2006 IEEE International Symposium on, pages 175 –177, oct. 2006.

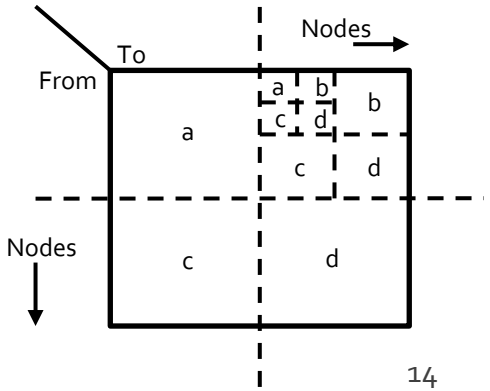
Benchmarking Graph Stores in Exascale – Yahoo! Cloud Serving Benchmark

- XGDBench is an extension of Yahoo! Cloud Serving Benchmark (YCSB).
- YCSB is a benchmarking framework for cloud data serving systems.
- The framework is composed of,
  - workload generator client
  - package of standard workloads that cover interesting parts of the performance space.

Benchmarking Graph Stores in Exascale – Synthetic Graph generators

- Synthetic Graph models can be classified in to five categories
  - Random graph models (e.g., Erdos Renyi)
  - Preferential attachment models (e.g., Barabasi-Albert)
  - Optimization-based models (e.g., Highly Optimized Tolerance)
  - Tensor-based models (e.g., R-MAT)
  - Internet-specific models (e.g., Inet)
- The best generator model depends on the application area
- R-MAT Model
  - Graph generated by R-MAT depends on few parameters.

    Scale(n), a, b, c, and d. The sum of a,b,c,d are 1.

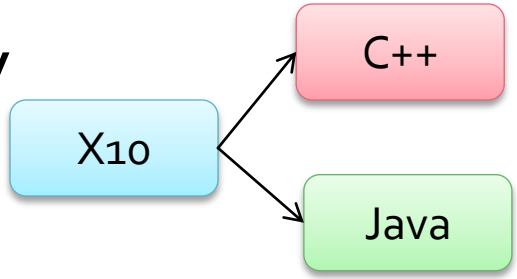


Benchmarking Graph Stores in Exascale – X10

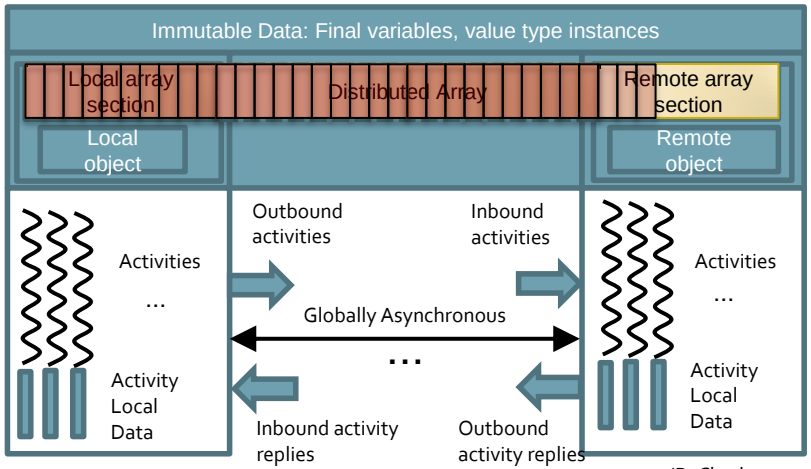
- X10
  - robust programming model
  - withstand architectural challenges
  - multi-core systems, hardware accelerators, clusters, and supercomputers.
- X10 simplifies the programming model
  - increase in programmer productivity
- X10 is a strongly typed, object-oriented language
  - static type-checking
  - static expression of program invariants.

Benchmarking Graph Stores in Exascale – X10 (Cont.)

- X10 applications are developed by source-to-source compilation.
- Managed X10 for XGDBench.
- Distributed data structures (e.g., DistArray)
  - Distributed storage of large graphs that could not be stored in single place.



X10 allows for writing extensions for XGDBench for future Exascale graph stores with less effort.

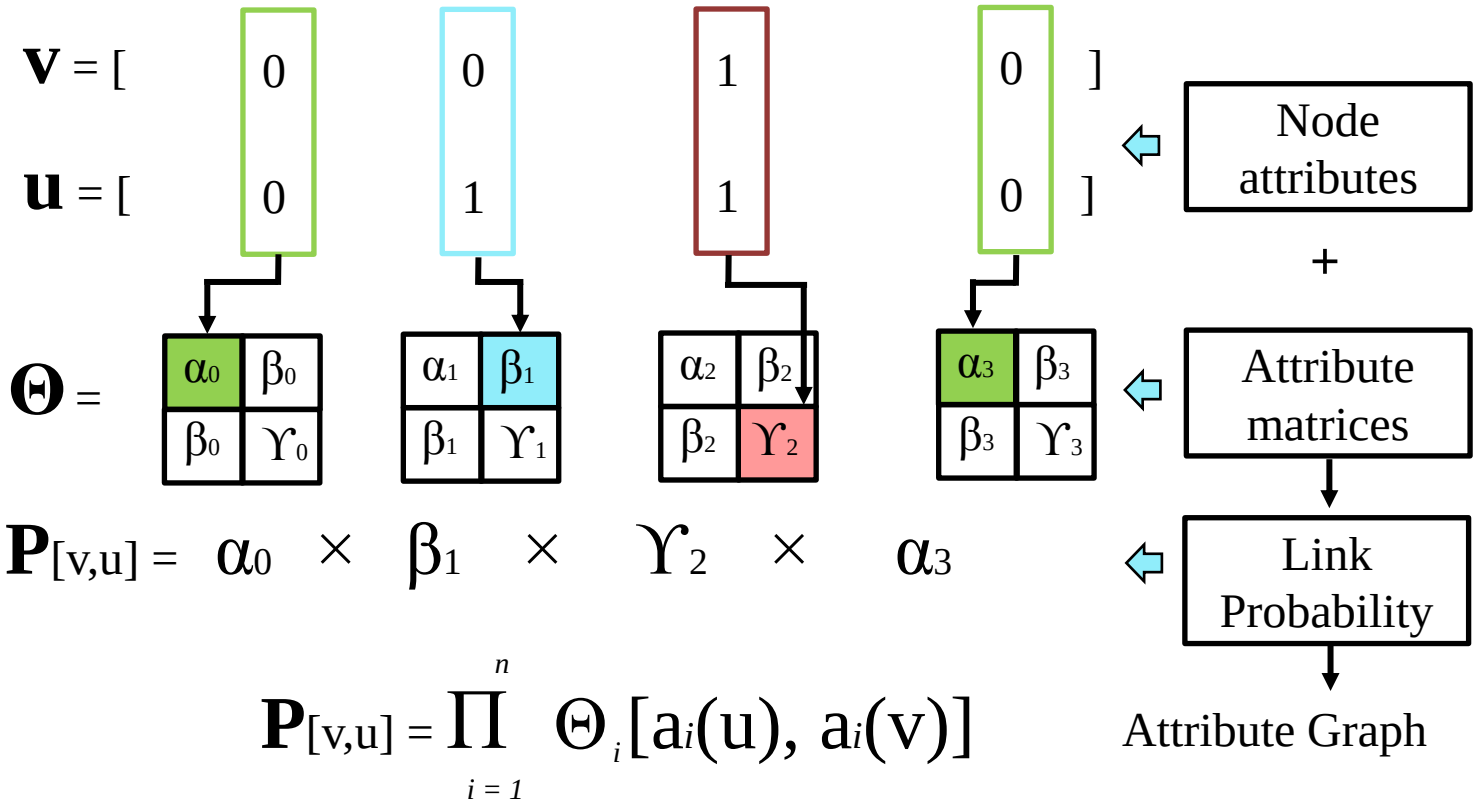


(P. Charles, et. al. 2005)

Benchmarking Graph Stores in Exascale – An overview of Multiplicative Attribute Graphs (MAG)

- An approach for modeling the structure of networks which have node attributes.
- MAG naturally models
  - interactions between the network structure
  - node attributes.
- MAG graphs
  - are analytically tractable
  - have statistically interesting properties.
- MAG creates realistic attribute graphs much suited for benchmarking graph databases.

Benchmarking Graph Stores in Exascale – An overview of Multiplicative Attribute Graphs (Cont.)

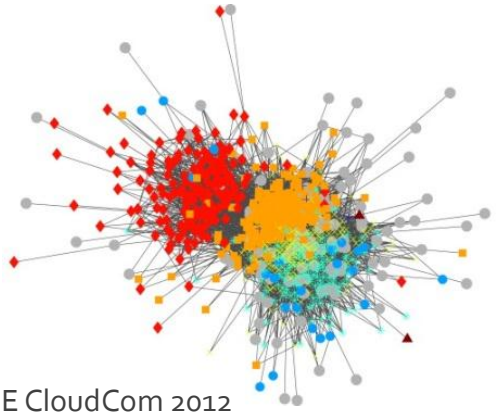


Benchmarking Graph Stores in Exascale – An overview of Multiplicative Attribute Graphs (Cont.)

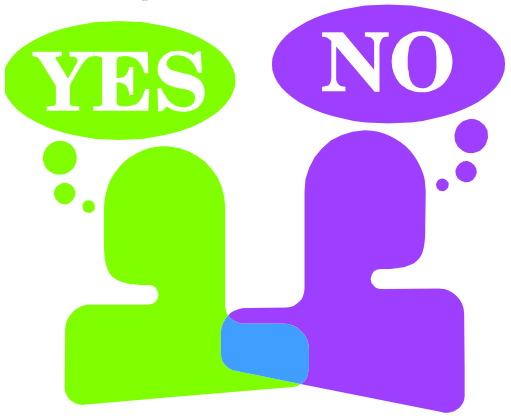
- Two vertices v and u each having a vector v of n categorical attributes.
- Each attribute has a cardinality di (i=1,2,…,n)
- There are also n matrices denoted by θi, where θi ∈ di × di for i=1,2,…,n. Each entry of θi is the affinity of a real value between 0 and 1.
- The values α, β, and γ are floating point values between 0 and 1.

Benchmarking Graph Stores in Exascale – Simplification of Multiplicative Attribute Graphs

- MAG algorithm used in XGDBench
  - simplified version considering the undirected graphs.
- The simplification is achieved by making,
  - each θ symmetric.
  - The node attributes are made binary.

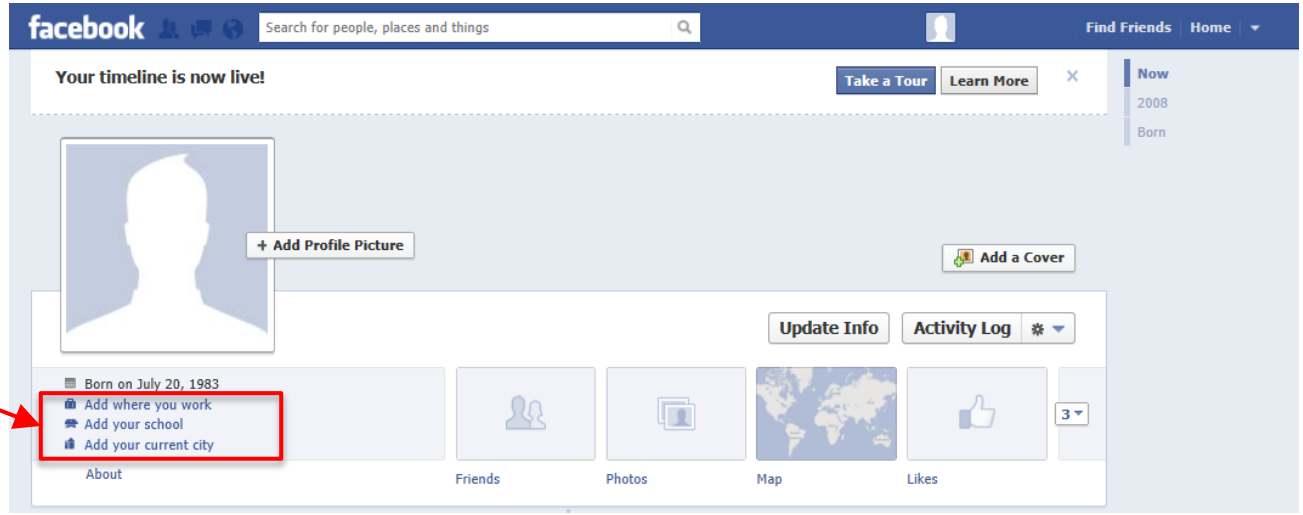


A student-only subset of the Reed College Facebook network (Amanda L. Traud et al., 2012)



Benchmarking Graph Stores in Exascale – Simplification of Multiplicative Attribute Graphs

Information that can be treated as Yes/No questions



- Makes the θi to be a 2 × 2 matrix.
- We assume that all the affinity matrices are equal (i.e., θi = θ).

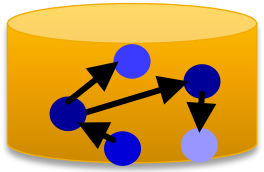
Benchmarking Graph Stores in Exascale – Implementation of Multiplicative Attribute Graphs Algorithm

Algorithm 1 mag(nVertices, nAttributes, attribThresh, pThresh, theta)

```
1: nodeAttribs ← randZeroOrOne(nVertices, nAttributes,
                                attribThreshold)
2: result ← ones(nVertices, nVertices)
3: for i ← 0 to nVertices do
4:    for j ← 0 to nVertices do
5:       for k ← 0 to nAttribs do
6:          if nAtt[i,k] = nAtt[j,k] then
7:             if nAtt [i,k] = 0 then
8:                result[i,j] = result[i,j] * theta[0]
9:             else
10:               result[i,j] = result[i,j] * theta[3]
11:            end if
12:         else
13:            if nAtt[i,k] = 0 and nAtt[j,k] = 1 then
14:               result[i,j] = result[i,j] * theta[1]
15:            else if nAtt[i,k] = 1 and nAtt[j,k] = 0 then
16:               result[i,j] = result[i,j] * theta[2]
17:            end if
18:         end if
19:      end for
20:   end for
21:end for
22: for i ← 0 to nVertices do
23:    for j ← 0 to nVertices do
24:       if result[i,k] > pThresh then
25:          result[i,k] = 1
26:       else
27:          result[i,k] = 0
28:       end if
29:    end for
30: end for
31: return (result)
```

The probability of an edge between pairs of vertices is controlled by the product of individual attribute-attribute affinities.

Methodology of XGDBench – Why Social Networks?



Graph Database

- Graph database applications for social networking services.
  - Online Social Networks (OSN) is one of the rapidly growing application areas.
  - Data storage and analysis is conducted on cloud infrastructures.
- OSNs represent a general representative application of graph databases.

Requirements of XGDBench – Attribute Read/Update

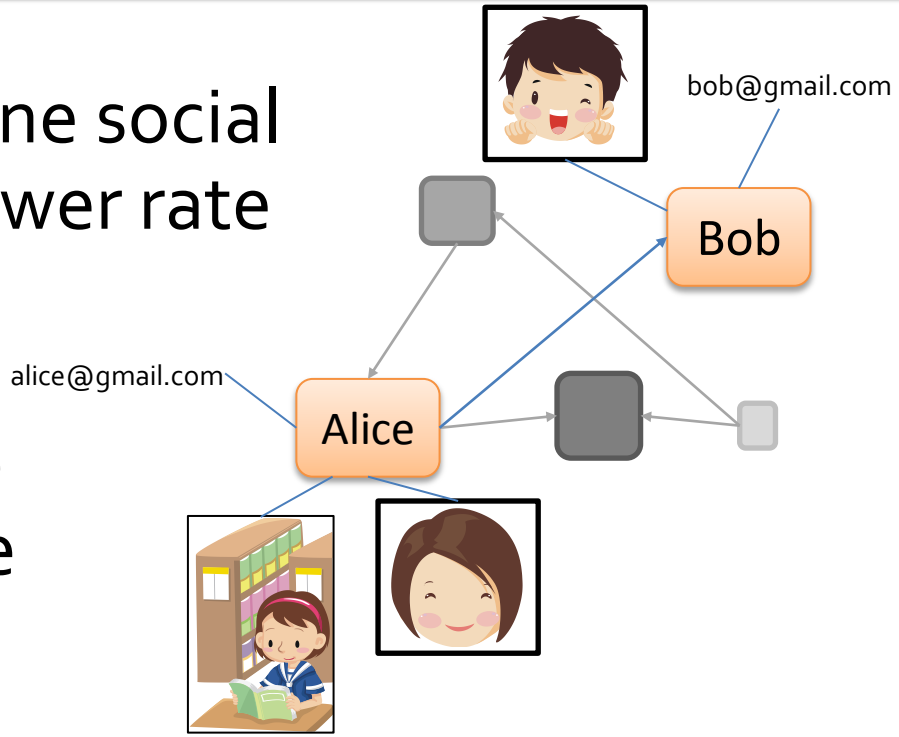
- Massive graphs will be handled online
- Graphs will be partially loaded in to memory
- The workloads will include both read/update operations.

In most of the applications read operations will dominate the workload.

- Therefore we included a read-heavy workload (0.95 probability of read and 0.05 probability of write operations)

Requirements of XGDBench – Attribute Read/Update (Cont.)

- Friendship graphs of online social networks change at a slower rate compared to their node properties.
- Performance of attribute update operation is more important compared to node/edge update.
- Benchmarking platform needs to be scalable to store data in-memory for update operations.



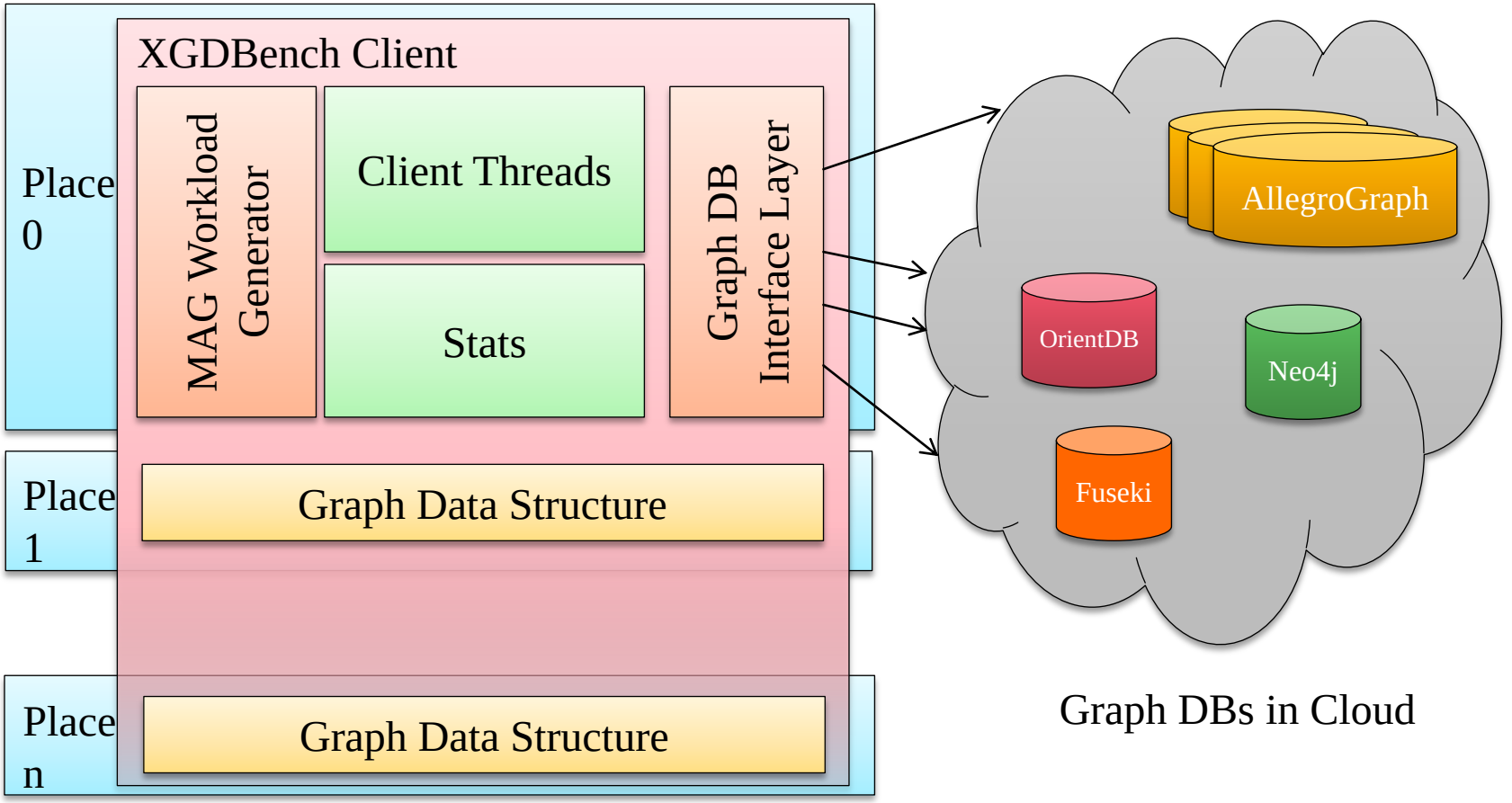
Requirements of XGDBench – Graph Traversal

- Graph databases
  - data encoded in its graph structure that could be obtained by traversing them.
- Use a traversal algorithm
  - most frequently executed against the graph database.
- Listing friends of friends is one of frequently used traversal operations on OSNs.
  - Execute breadth-first search (BFS) from a particular vertex for detecting the connected component of a graph.

Requirements of XGDBench – Graph Traversal – Basic operations of Graph Databases

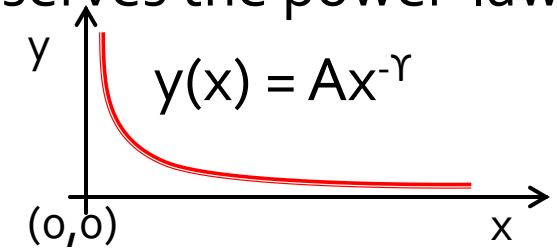
Operation	Description
Read	Read a vertex and its properties
Insert	Inserts a new vertex
Update	Update all the attributes of a vertex
Delete	Delete a vertex from the DB
Scan	Load the list of neighbors of a vertex
Traverse	Traverse the graph from a given vertex using BFS. This represents finding friends of a person in social networks.

Implementation



Implementation – Two phases of execution

- Data Loading Phase
  - Generates an attribute graph using MAG algorithm
  - load the graph data to the server.
- Transaction Phase
  - Invokes the basic operations on the loaded graph data.
- Update operation on the graph data
  - preserves the power-law distribution.
  - Update operations are conducted *only on the attributes that are not related to calculation of probability of an edge*.
  - Insert operations of the vertices preserves the power-law structure.



Implementation – Core workloads of XGDBench

A : Update heavy
Workload A is a mix of 50/50 read/update workload. Read operations query a vertex V and reads all the attributes of V. Update operation changes the last login time of Attributes related to vertex affinity are not changed.
B : Read mostly
A mix of 95/5 read/update workload. Read/update operations are similar to A.
C : Read only
Consists of 100% read operations. The read operations are similar to A.
D : Read latest
This workload inserts new vertices to the graph. The inserts are made in such a way that the power law relations of the original graph are preserved.
E : Short Ranges
This workload reads all the neighbor vertices and their attributes of a Vertex A. This represents the scenario of loading the friendliest of person A on to an application.

Evaluation – Properties of the MAG Data Generator

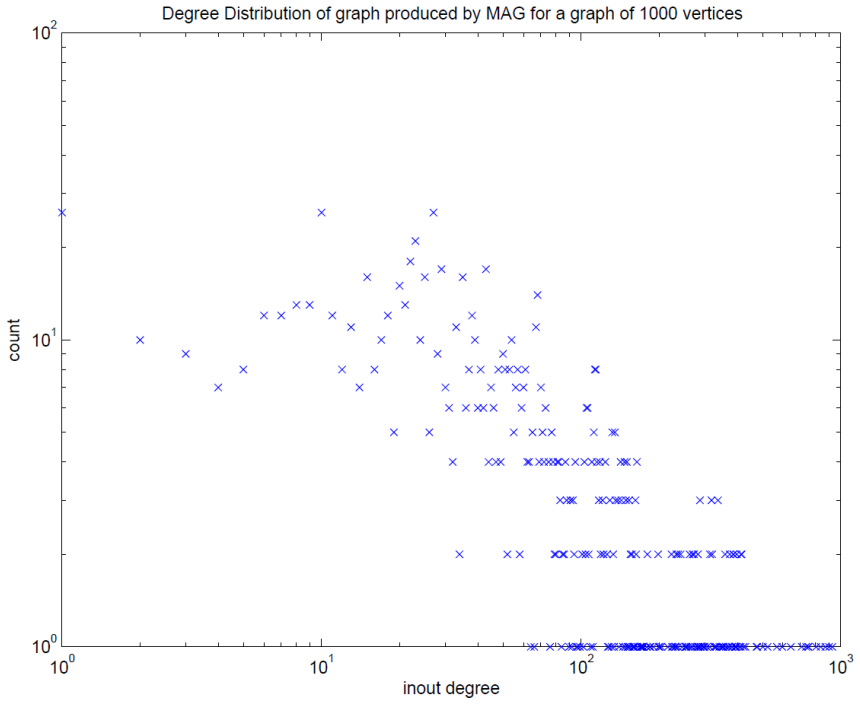
- Evaluated
  - Degree distribution
  - graph community structure of MAG model
- Power-law distribution – Degree distribution of many real world graphs (web, social networks, etc.) satisfy power-law distribution.

$$y(x) = Ax^{-\gamma}$$

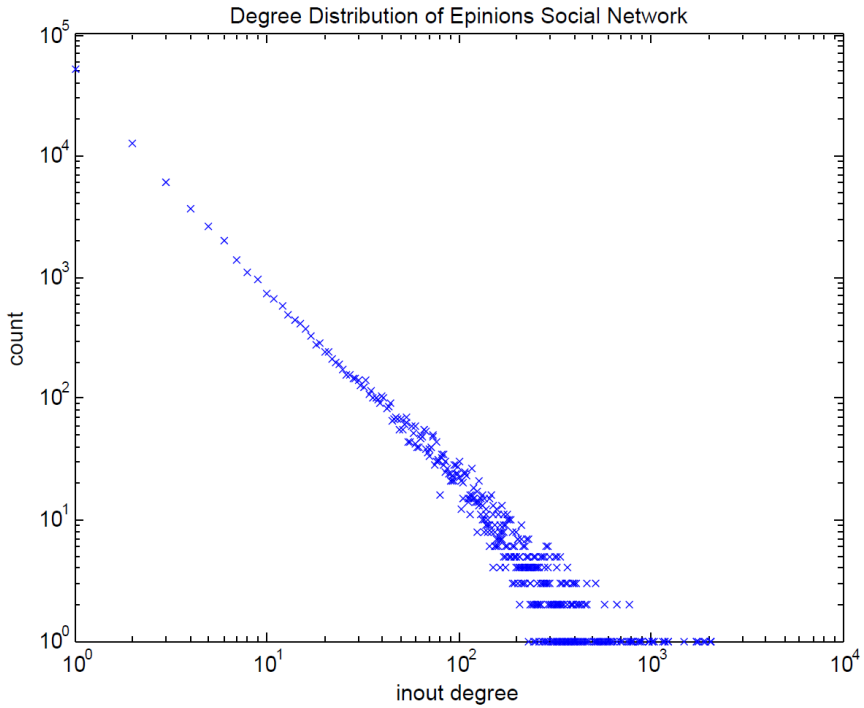
- Plotted the degree distribution of a graph with 1000 vertices produced by XGDBench generator

.

Evaluation – Properties of the MAG Data Generator – Comparison of the degree distribution.



MAG



Epinions Social Network

MAG creates a degree distribution that is similar to a power-law distribution.

Evaluation – Comparison of MAG with R-MAT – Categorical Prominence

- Graph databases
  - designed to store colorful graphs (With node/edge attributes)
- The generator should create such realistic graphs to generate realistic workload scenarios.
- Implemented R-MAT version of XGDBench
  - Replaced data generator algorithm with R-MAT algorithm.
- Randomly populate the vertex attributes to mimic the attribute graphs produced by MAG model.

Evaluation – Comparison of MAG with R-MAT – Categorical Prominence (Cont.)

- We used five graphs from each model with R-MAT scale (n) 10 to 14.
- R-MAT graph was generated with the parameters

a	0.6
b	0.15
c	0.15
d	0.1

- For MAG we used a probability threshold of 0.25.
- Each graph had 4 attributes per vertex.

Evaluation – Comparison of MAG with R-MAT – Categorical Prominence (Cont.)

- Community cluster analysis on each graph using Cytoscape.
- The vertices in the top three resulted clusters were further clustered using vertex attributes.
- Next, Take the percentages of vertex counts in each cluster and rank them based on their percentage values.
- Cluster Prominence Metric (Cp) the difference between the largest sub cluster and the second largest sub cluster.

The graphs generated by MAG model had sub clusters with higher prominence.

- The communities creates by MAG represented phenomenon of social affinity that is present in real social networks.

Evaluation – Comparison of MAG with R-MAT – Categorical Prominence (Cont.)

	MAG		R-MAT	
Vertices (Scale)	Edges	Cluster prominence (Cp)	Edges	Cluster prominence (Cp)
1024 (10)	23077	24.00	2704	6.33
2048 (11)	121298	23.33	3912	3.33
4096 (12)	413281	29.33	1218	1.33
8192 (13)	1634377	26.67	8782	3.33
16384 (14)	6363791	36.67	15974	3.67

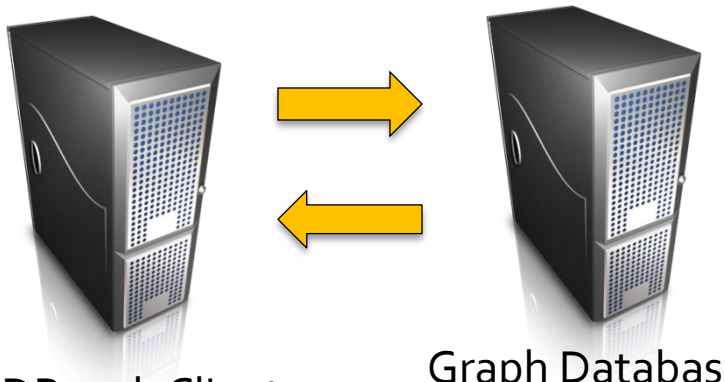
Performance evaluation of graph databases using XGDBench

Graph databases

Name	Data Model	Programming Language	Version	JVM Heap Size (GB)
OrientDB	Network	Java	v1.0rc9	2
Neo4j	Network	Java	community v1.6.1	4
Fuseki	RDF	Java	v0.2.1	2
AllegroGraph	RDF	LISP	v4.6	-

Specifications of a single node on Tsubame 2.0

CPU	Two Intel Xeon X5670 @2.93GHz, each CPU has 6 cores (total 12 cores)
RAM (GB)	54
HDD (GB)	-
Network	SDR Infiniband × 2
SSD (GB)	120
OS	SUSE Linux Enterprise Server 11 SP1
File System	Lustre



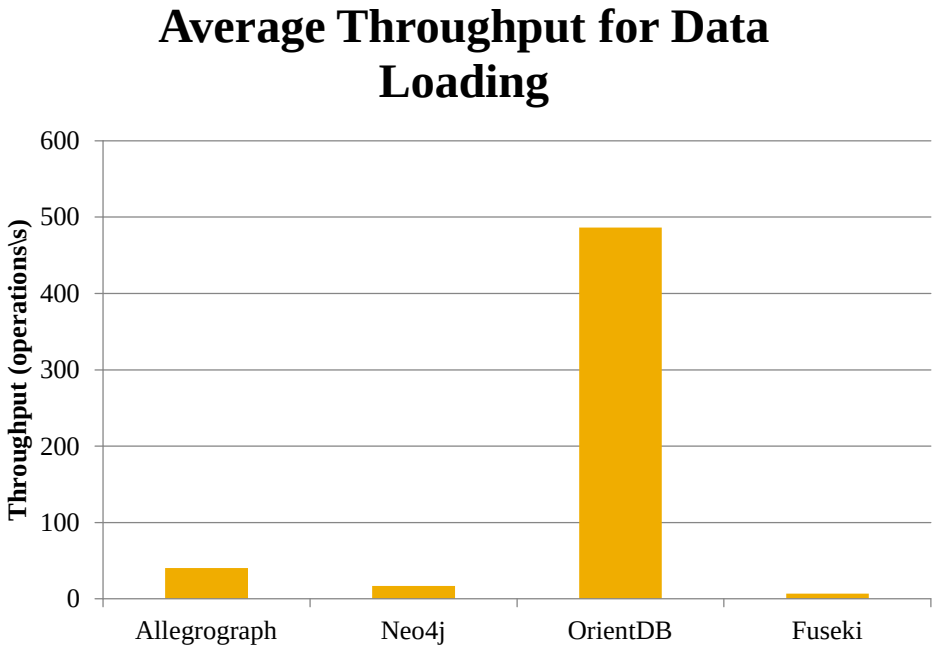
XGDBench Client

Graph Database server

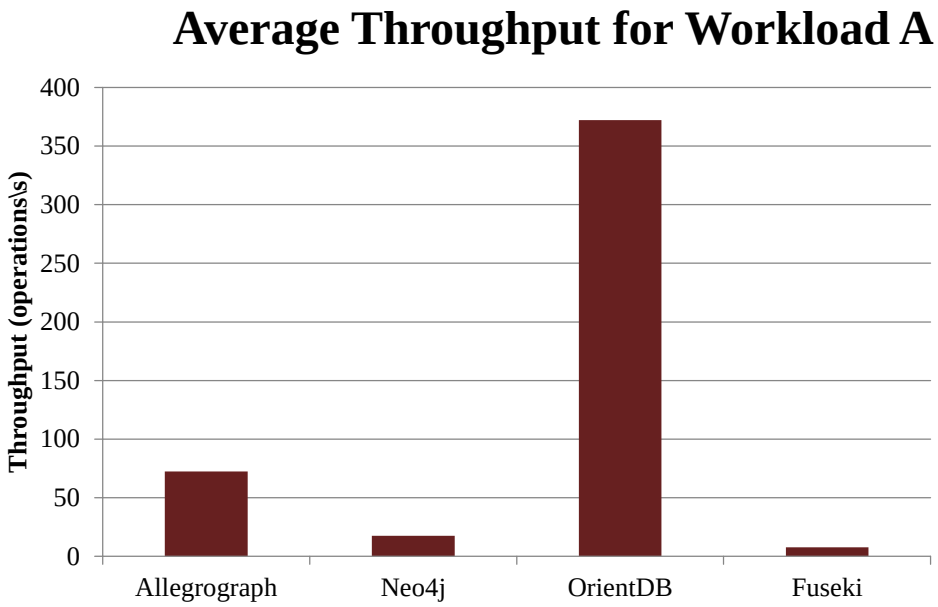
Performance evaluation of graph databases using XGDBench (Cont.)

- Done on Tsubame 2.0 cloud computing environment.
- Used two nodes, one node ran the graph database server the other node ran the XGDBench.
- XGDBench was set up to use 8GB heap for X10 runtime.
- Use X10 2.2.2 which was build with fully optimized settings.
- Graph sizes used for evaluation was 1024.

Performance evaluation of graph databases using XGDBench (Cont.)

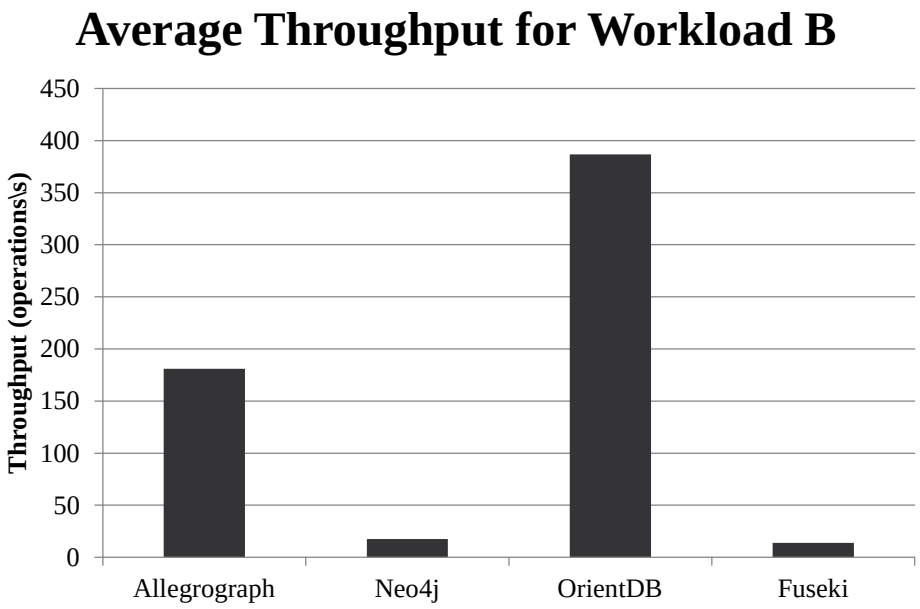


(a)

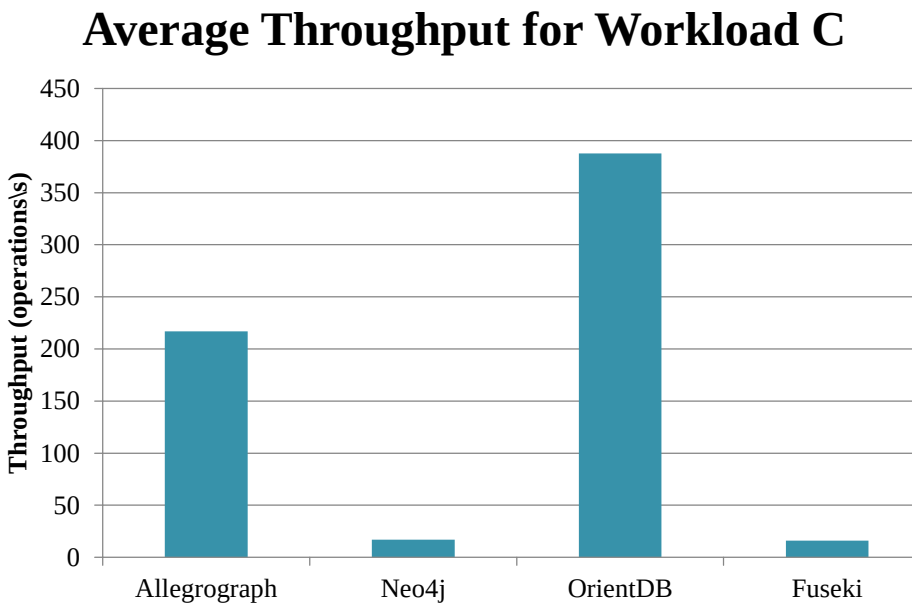


(b)

Performance evaluation of graph databases using XGDBench (Cont.)

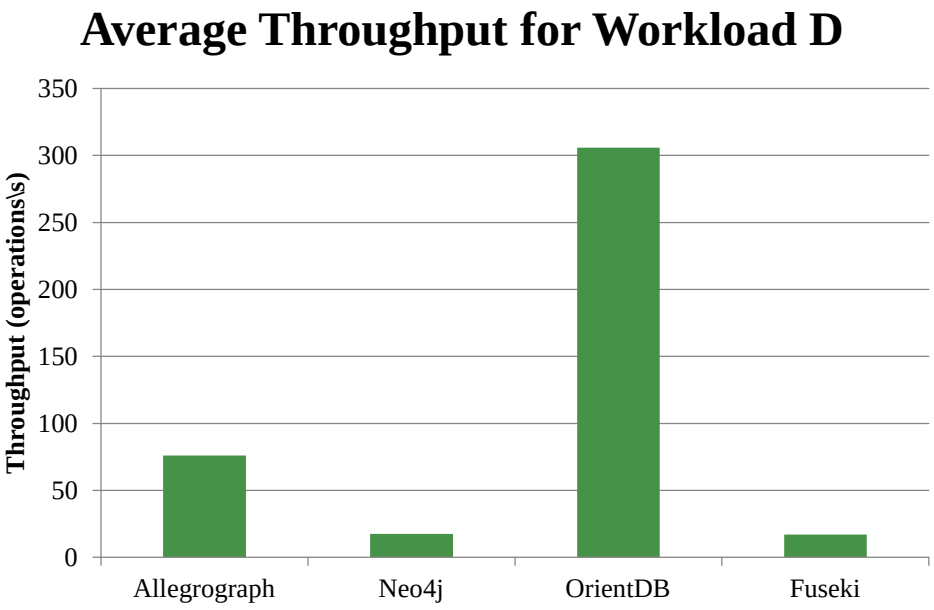


(c)

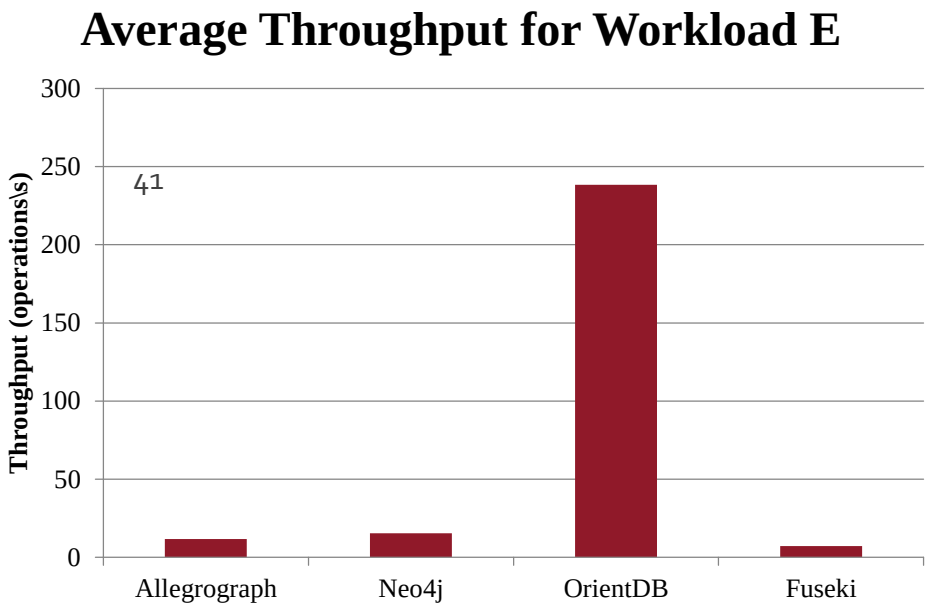


(d)

Performance evaluation of graph databases using XGDBench (Cont.)



(e)



(f)

Discussion and Limitations

- XGDBench's graph generator model is much suited for evaluating performance of graph databases.
- Attribute graphs produced by MAG follow Power-law distribution
- Most of the current graph databases are not distributed.
- Number of vertices (1024)

Conclusion and Further Work

- XGDBench is a graph database benchmarking platform for Exascale clouds.
- The data generator of XGDBench is based on MAG model
  - enables realistic modeling of attribute graphs.
- XGDBench implemented using X10
  - enables easy extension of the framework in future.
- Evaluated the applicability of MAG model for graph database benchmarking
  - conducted a performance evaluation.
- From the community cluster analysis we observed that MAG model creates much realistic attribute graphs compared to the popular R-MAT model.

Conclusion and Further Work (Cont.)

- Conduct thorough evaluation of graph databases using XGDBench.
- Implement travers operation based workloads.
- Investigate the reason why graph databases perform poorly and find methods to improve their performance.

Thank You!