The Future of Graph Computing

Toyo (Toyotaro) Suzumura
IBM T.J. Watson Research Center
Barcelona Supercomputing Center
The University of Tokyo
Graph Ecosystem : What we have achieved so far

- **Algorithms** :
  - Tons of new algorithms … (ICML/ KDD / SIGMOD...)
  - Approximate algorithm, Lower time/space complexity

- **High Performance Computing**
  - Parallel and Distributed Processing
  - Leveraging new accelerators and storage layers

- **Graph Analytics Library**
  - GraphX / Spark, Apache Giraph, ScaleGraph, etc

- **Graph Database**
  - IBM System G, Neo4J, Titan, … etc

- **Benchmarking**
  - Graph500, LDBC, etc
Statistical Survey on Graph-Related Papers

- Top-Tier Conferences over 5 years (2011-2015) in 3 research fields
  - **HPC**: Supercomputing, IPDPS
  - **DB**: ICDE, SIGMOD, VLDB
  - **ML**: ICML, SDM, KDD, ICDM
Graph500 Trend: Innovative Graph Algorithms and Implementation brings us 7x performance

![Graph500 Performance Trend (2010-2015)](image)

- K Computer: 38,621 GTEPS
- Sequoia: 5,524 GTEPS
- 2010/11 to 2015/11 performance trend over years.
Insights from our Graph500 Challenge

We learned that algorithmic and implementation innovation greatly accelerated the performance on the same platforms.

What’s the next to do?

Tokyo Tech – TSUBAME 2.5
1024 nodes (12,288 cores)

Riken K Computer
82,944 nodes (663,552 cores)
Even if you optimize popular frameworks based on JVM, it is still slow ;(. How do we solve this dilemma?

ScaleGraph vs. GraphX/Spark

Graph analytics involves more IO-bounded operation considering efficient data communication among compute nodes, memory access, workload imbalance, etc.
My Observation

• General computational model for incremental graph analytics
  (vs. Pregel, GMI-V for batch-analytics)
  • A bunch of work on incremental algorithms (incremental PageRank/Community Detection)
    • Could be generalized ?
    • Incremental Pregel ? Other method ?

• How would we align with other areas such as database and machine learning / data mining area ?
  • We just follow the outcome from the ML field ?
Financial Risk Prediction using System G

Risk Factors

- Adjunct Relationship
- Guarantee Relationship
- Adjunct and Guarantee
- Transaction Relationship
- Stockholder Relationship
- EgoNet Relationship
- Enterprise Type

Feature Pattern of Clients with high risk

Graph Analysis
Machine Learning
Cognitive Reasoning

Perception
Judgement
Abstract
comprehension
Observation
Memory
Graph Database
Manage and retrieve Linked Big Data
IBM System G: Graph Computing Framework

**Visualization**
- Huge Network Visualization
- Network Propagation
- I2 3D Network Visualization
- Geo Network Visualization
- Graphical Model Visualization

**Analytics**
- Communities
- Graph Search
- Network Info Flow
- Bayesian Networks
- Centralities
- Graph Query
- Shortest Paths
- Latent Net Inference
- Ego Net Features
- Graph Matching
- Graph Sampling
- Markov Networks

**Middleware**
- Graph Processing Interface
  - Shared Memory Run Time Library
  - Distr. Memory RT Library
  - Graphs RDMA
  - MPI
  - Graphs FPGA/HMC
  - Infosphere Streams (ISS)
- BigInsights
- Pthreads
- PERCS Coh. Clus.
- Cluster (BladeCenter, BlueGene)

**Database**
- Graph Data Interface
  - GBase (update, scan, operators, indexing)
  - Native Store
  - Netezza
  - DB2 RDF
  - TinkerPop Compliant DBs
- HBase
- HDFS
- DB2
- DB2 RDF
- TinkerPop Compliant DBs
- System G Assets
- Open Source
- IBM Product
- Hardware

© 2016 IBM Corporation
Graph Analytics Project with Financial Institute

- **Project Duration**: 1-2 months (1 – 2 people fully involved with the project)

- **Graph Size and Hardware**:  
  - Property graph with more than 10 attributes  
  - **Graph Size**:  
    - **First phase**: **Millions of vertices and edges**  
    - **Second phase**: **Billion-scale graph** with more fine-grained time-series transaction data  
  - **Hardware**: 60 CPU cores in total with shared-memory machine

- **Some Graph Analytics**  
  - Cycle detection with various condition on properties  
  - Egonet extraction  
  - Prediction based on Machine Learning
Graph Analytics + Machine Learning

Graph Analytics

Feature Extraction

Risk Prediction

Machine Learning

Risk Score

Graph Database / Other data stores
Missing Graph Component from Industry Point of View

1. **Scalable Graph Database with trillion-scale time-series data**
   - Seamless programming model with regular graph processing
   - Leveraging Memory Hierarchy

2. **More powerful Graph Query language for temporal graphs**
   - Gremlin, Sparql, Cypher… are not enough

3. **Multi-layer and Multiplex networks**
   - Multi-modal data source brings requirements for multi-layer networks

4. **How to leverage Accelerators for high performance graph processing?**
   - Vertex-centric model (Pregel) on GPGPUs / NVLink
   - **Domain-specific language** for Pregel that allows you to write your Graph algorithms on CPU, or GPU, or hybrid engines.
5. **General Incremental Processing Model for Dynamic Graphs**
   - Vertex-centric model is still attractive since it brings better parallelism.
   - Pregel + Spark RDD-like paradigm might be a better combination.
   - But like the modularity-optimization based paradigm, it needs global synchronization.

6. **Workflow language for ETL, Graph processing, and other machine learning**

7. **Deep learning for Static/Dynamic Graphs**
   - How could we apply CNN / RNN for static / dynamic graphs?
   - Applications: Fraud detection, Non performing loan, and anti-money laundering.
Modelling Time-Series Graph

- **Two possible models**
  - **Design A**: More intuitive way for expressing graphs
    - Add_edge(A, B, tk, 100)
  - **Design B**: Separating time-series data from connectivity
    - Users can define graph model as Design A, but the system could convert Design A to Design B automatically.
      - Eab.addTransaction(tk, 100)

### Design A

- Many edges?
- 86400 edges for 1 day with the granularity of 1 second

![Design A Diagram](image)

### Design B

- Big Time-Series Table?
- Time-series database

![Design B Diagram](image)
Getting a snapshot of Temporal Graph

- Give me a snapshot at 2015/11/13 8:30 from temporal graph
Getting a snapshot of Temporal Graph

- How can efficiently store a temporal graph efficiently?
  - 1) Space efficiency for storing many snapshots
  - 2) Or read efficiency to access particular snapshot

Versioning Issue

One year ago – 2015/11/13
Leveraging Memory Hierarchy

- **Vertex property (time, space)**
  - Latest data should be in DRAM
  - Frequently accessed vertices such as Superhubs
- **Edge property (time, space)**
Discussion!

Thank You