Billion-Scale Graph Analytics

Toyotaro Suzumura, Koji Ueno, Charuwat Houngkaew, Masaru Watanabe, Hidefumi Ogata, Miyuru Dayarathna, ScaleGraph Team

Tokyo Institute of Technology / JST CREST
IBM Research
Outline

- Introduction
- Graph500 Benchmark
- ScaleGraph: Billion-Scale Graph Analytics Library
- Time-Series Analysis for Whole Twitter Network
- Summary
Large-Scale Graph Mining is Everywhere

Internet Map

Cybersecurity
Medical Informatics
Data Enrichment
Social Networks
Symbolic Networks

Social Networks

Symbolic Networks:

Cyber Security (15 billion log entries / day for large enterprise)

Protein Interactions
Large-Scale Graph Processing System (2011-2018)

Disaster Management

Transportation, Evacuation, Logistics

Energy・Power Saving

Social Network Analysis

Sensors
- Smart Meters
- Smart Grid
- GPS
- SNS (Twitter)

Large-Scale Graph Visualization

Real-Time Graph Stream Processing

Large-Scale Graph Library

Centrality
- Shortest Path
- Quickest Flow Problem

PageRank / RWR
- Clustering
- Semi-Definite Programming
- Mix Integer Programming

Real-Time Stream Processing System

X10 Language

100 Peta Flops Heterogeneous Supercomputer

Large-Scale Graph Store
Graph500: Big Graph Competition

Graph500 is a new benchmark that ranks supercomputers by executing a large-scale graph search problem.

- The benchmark is ranked by so-called **TEPS (Traversed Edges Per Second)** that measures the number of edges to be traversed per second by searching all the reachable vertices from one arbitrary vertex with each team’s optimized BFS (Breadth-First Search) algorithm.
We propose an optimized method based on 2D based partitioning and other various optimization methods such as communication compression and vertex sorting.

We developed CPU implementation and GPU implementation.

Our optimized GPU implementation can solve BFS (Breadth First Search) of large-scale graph with $2^{35}$ (34.4 billion) vertices and $2^{39}$ (550 billion) edges for 1.275 seconds with 1366 nodes (16392 cores) and 4096 GPUs on TSUBAME 2.0.

This record corresponds to 431 GTEPS.

Vertex Sorting by utilizing the scale-free nature of the Kronecker Graph

2D Partitioning Optimization

Scalable 2D partitioning based CPU Implementation with Scale 26 per 1 node

Performance Comparison with CPU and GPU Implementations
TSUBAME 2.5 Supercomputer in Tokyo

TOP 10 Systems - 06/2011

1. K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect
2. Tianhe-1A - NUDT TH MPP, X5670 2.93GHz 6C, NVIDIA GPU, FT-1000 8C
3. Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz
4. Nebulae - Dawning TC3600 Blade, Intel X5650, NVIDIA Tesla C2050 GPU
5. TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, NVIDIA GPU, Linux/Windows

Complete Results - November 2011

<table>
<thead>
<tr>
<th>Rank</th>
<th>Machine</th>
<th>Owner</th>
<th>Problem Size</th>
<th>TEPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NNSA/SC Blue Gene/Q Prototype II (4096 nodes / 65,536 cores)</td>
<td>NNSA and IBM Research, T.J. Watson</td>
<td>32</td>
<td>254,349,000,000</td>
</tr>
<tr>
<td>2</td>
<td>Lomonosov (4096 nodes / 32,768 cores)</td>
<td>Moscow State University</td>
<td>37</td>
<td>103,251,000,000</td>
</tr>
<tr>
<td>3</td>
<td>TSUBAME (2732 processors / 1366 nodes / 16,392 CPU cores)</td>
<td>GSIC Center, Tokyo Institute of Technology</td>
<td>36</td>
<td>100,366,000,000</td>
</tr>
<tr>
<td>4</td>
<td>Jugene (65,536 nodes)</td>
<td>Forschungszentrum Jülich</td>
<td>37</td>
<td>92,876,900,000</td>
</tr>
<tr>
<td>5</td>
<td>Intrepid (32,768 nodes / 131,072 cores)</td>
<td>ANL</td>
<td>35</td>
<td>78,869,900,000</td>
</tr>
</tbody>
</table>
Our Scalable Algorithm continuously achieves 3rd or 4th place in the World since 2011/11

- No.4 (2012/06)
- No.3 (2011/11)
- No.3 (2012/06)
Outline

- Introduction
- Graph500 Benchmark
- ScaleGraph: Billion-Scale Graph Analytics Library
- Time-Series Analysis for Whole Twitter Network
Programming models that offer performance and programmer productivity are very important for conducting big data analytics in Exascale Systems.

HPCS languages are an example for such initiatives.

It is very important for having complex network analysis software APIs in such languages.
**ScaleGraph** : Large-Scale Graph Analytics Library

- **Aim** - Create an open source X10-based Large Scale Graph Analytics Library (beyond the scale of billions of vertices and edges).

- **Objectives**
  - To define concrete abstractions for Massive Graph Processing
  - To investigate use of X10 (i.e., PGAS languages) for massive graph processing
  - To support significant amount of graph algorithms (e.g., structural properties, clustering, community detection, etc.)
  - To create well defined interfaces to Graph Stores
  - To evaluate performance of each measurement algorithms and applicability of ScaleGraph using real/synthetic graphs in HPC environments.

**URL:** http://www.scalegraph.org/
Programming Language X10

X10 is a new parallel distributed programming language being developed by IBM Research.

- X10 aims at improving the productivity of highly parallel and distributed applications.
  - Enables scalable programming for parallel distributed environment, where many multicore SMP chips and GPGPUs are interconnected.

- **X10 adopts APGAS** (Asynchronous Partitioned Global Address Space) **programming model**.
  - Can manage multiple machines as a global memory space partitioned into “Places”.
  - Can create lightweight asynchronous “Activities”.
  - Supports creation and reference of activities and objects in remote places.

- **X10 supports various execution environments**.
  - Can run both on Java execution environments and native environments.
  - Provides development tools integrated into Eclipse.

- **X10 is being developed as an open source project**.
  - See http://x10-lang.org/ for more information.

```
public class MyDistCalc {
    public static def main(Array[String]) {
        val R = 1..1000; val D = Dist.makeBlock(R);
        val arr = DistArray.make[Int](D, (([i]:Point)=>i));

        val places = arr.dist.places();
        val tmp = new Array[Int](places.size);
        finish for ((i) in 0..places.size-1) async {
            tmp(i) = at (places(i)) {
                val a = arr [here];
                var s:int = 0;
                for (pt in a) s += a(pt)*a(pt);
                s // return value of at
            };
        }

        var result:Int = 0;
        for (pt in tmp) result += tmp(pt);
        Console.OUT.println(result); // -> 333833500

        // We can actually use DistArray.map and reduce
        val r = arr.map((i:Int)=>i*i).reduce[Int](+)(0);
        Console.OUT.println(r); // -> 333833500
    }
}
```
Features of ScaleGraph

- XPregel framework which is based on Pregel computation model\(^1\) proposed by Google
- Optimized collective routines (e.g., alltoall, allgather, scatter and barrier)
- Highly optimized array data structure (i.e., MemoryChunk) for very large chunk of memory allocation
- Rich graph algorithms (e.g., PageRank, spectral clustering, degree distribution, betweenness centrality, HyperANF, strongly-connected component, maximum flow, SSSP, BFS)
- We achieved running PageRank, spectral clustering, degree distribution on huge Twitter graph with 469M of users and 28.5B of relationships

The scale-28 graphs we used have $2^{28}$ ($\approx 268M$) of vertices and $16 \times 2^{28}$ ($\approx 4.29B$) of edges.
Outline

- Introduction
- Graph500 Benchmark
- ScaleGraph: Billion-Scale Graph Analytics Library
- Time-Series Analysis for Whole Twitter Network
Understanding time-series nature of large-scale social networks (e.g. separation of degree, diameter, clustering, ..)

Crawled the entire Twitter follower/followee network of 826.10 million vertices and 29.23 billion edges. How could we analyze this gigantic graph?

Supercomputers
Crawling Billion-Scale Twitter Follower-Followee Network

- with Twitter API (v1.0) from Jul. 2012 to Oct. 2012 (around 3 months).
- begin with top 1,000 users*¹ with the largest number of followers
- according to breadth-first search along the direction of follower

![Diagram showing follower network](http://twitaholic.com/top100/followers/)

*¹: Twitaholic. http://twitaholic.com/top100/followers/
We stopped our crawling at depth 29
- Because the user after depth 26 was less than 100.
- Finally, we collected 469.9 million user data.

Collect two kinds of user data by crawling for 3 months
- 1. User profile
  - Include user id, screen_name, description, account creation time, timezone, etc.
  - The serialized data size is 91GB
- 2. Follower-friend
  - Adjacency list of followers and friends
  - The compressed (gzip) data size is 231GB

To perform the Twitter network analysis
- Apache Hadoop for large-scale data processing
- HyperANF for approximate calculation of degree of separation and diameter
  - Lars Backstrom*1 also use HyperANF for Facebook network analysis

*1: “Four degrees of separation” ACM Web Science 2012
Explore Twitter Evolution (1/2)
- Transition of the number of users

- Total user count (left fig.)
  - Twitter started at June 2006 and rapidly expanded from beginning of 2009.
  - Haewoon Kwak *1 studied Twitter network on July 2009

- Monthly increase of users (right fig.)
  - Twitter users increase, but it seems a little unstable...

*1: “What is Twitter, a social network or a news media?”
Explore Twitter Evolution (2/2)
- Transition of the number of users by regions-

- Classify 131 million users by “Time zone” property under 6 regions
  - Africa, Asia, Europe, Latin America and Caribbean (Latin), Northern America (NA), Oceania
  - Only 131 million user correctly set one’s own “Time zone”

- Massive change of ratio of users by region
  - Asia users: 8.30% => 20.8% (12.5% up)
  - NA users: 54.4% => 40.4% (14.0% down)

<table>
<thead>
<tr>
<th>Region</th>
<th>July 2009 # users</th>
<th>Ratio (%)</th>
<th>October 2012 # users</th>
<th>Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>0.13M</td>
<td>0.66</td>
<td>1.27M</td>
<td>0.96</td>
</tr>
<tr>
<td>Asia</td>
<td>1.65M</td>
<td>8.30</td>
<td>27.4M</td>
<td>20.8</td>
</tr>
<tr>
<td>Europe</td>
<td>3.01M</td>
<td>15.1</td>
<td>19.8M</td>
<td>15.1</td>
</tr>
<tr>
<td>Latin</td>
<td>3.80M</td>
<td>19.0</td>
<td>28.5M</td>
<td>21.6</td>
</tr>
<tr>
<td>NA</td>
<td>10.9M</td>
<td>54.6</td>
<td>53.1M</td>
<td>40.4</td>
</tr>
<tr>
<td>Oceania</td>
<td>0.45M</td>
<td>2.29</td>
<td>1.52M</td>
<td>1.15</td>
</tr>
<tr>
<td>Total</td>
<td>19.9M</td>
<td>100</td>
<td>131M</td>
<td>100</td>
</tr>
</tbody>
</table>

Characteristic of Twitter network also change?

Monthly increase of users by region
Monthly Increase of Users by Regions
Degree Distribution: **Unexpected value in in-degree distribution**

- “Scale-free” is one of the features of a social graph
- **Unexpected value in in-degree distribution**
  - at $x=20$ due to Twitter recommendation system
  - at $x=2000$ due to upper bound of friends before 2009
Reciprocity: decline from 22.1% to 19.5%

- Reciprocity is a quantity to specifically characterize directed networks. Traditional Definition:

\[ r = \frac{L^{\leftrightarrow}}{L} \]

- \( L^{\leftrightarrow} \) : # of edges pointing in both directions
- \( L \) : # of total edges

• As a result, **Twitter network reciprocity decline from 22.1% to 19.5%**

<table>
<thead>
<tr>
<th></th>
<th>July 2009</th>
<th>October 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td># of users</td>
<td>41.6 M</td>
<td>465.7 M</td>
</tr>
<tr>
<td># of edges</td>
<td>1.47 B</td>
<td>28.7 B</td>
</tr>
<tr>
<td>Reciprocity</td>
<td>22.1% *1</td>
<td>19.5%</td>
</tr>
</tbody>
</table>

*1: “What is Twitter, a social network or a news media?”
How many edges do celebrities have in Twitter network? ➔ Only 0.06% celebrities control most of the edges.

93% users have less than or equal to 100 followers. 99.94% users have less than or equal to 10,000 followers. However, their followers count are only 11% of total followers count. But still... 57.6% of total followers count.
Both degree of separation and diameter are measures to characterize networks in terms of scale of graph.

Definition

- **Degree of Separation**
  - *Average* value of the shortest-path length of all pairs of users.

- **Diameter**
  - *Maximum* value of the shortest-path length of all pairs of users

- Note: unreachable pairs are excluded from calculation
Experimental environment

- Using HyperANF [Paolo, WWW’12] on TSUBAME 2.0 (Supercomputer at TITECH)
  - TSUBAME 2.0 Fat node
    - 64 cores, 512 GB memory, SUSE Linux Enterprise Server 11 SP1
  - HyperANF Parameters
    - We set the logarithm of the number of registers per counter to 6 in order to reduce an error.

- Four times executions
  - Degree of Separation
    - take a average of 4 calculation
  - Diameter
    - take a minimum value of 4 calculation
    - because HyperANF guarantee lower bound of diameter
  - Each execution on 2012 took more than 42,000 sec.
Degree of Separation and Network Diameter (3/3)

- **Degree of Separation**
  - Only a little difference between ‘09 and ’12 in spite of the lapse of three years.

- **Diameter**
  - Diameter of 2012 is much larger than the one of 2009.

- **Cumulative Distribution**
  - In 2009
    - 89.2% of node pairs whose path length is 5 or shorter
    - 99.1% pairs whose it is 6 or shorter.
  - In 2012
    - 85.2% pairs whose it is 5 or shorter
    - 94.6% pairs whose it is 6 or shorter
Degree of Separation and Diameter for Time-Evolving Twitter Network

![Graph showing the degree of separation and diameter over time. The x-axis represents months from May 2005 to August 2013. The y-axis represents the degrees of separation and diameter values.]
## Classifying Degree of Separation by Spoken Language

<table>
<thead>
<tr>
<th></th>
<th>Spanish</th>
<th>Portuguese</th>
<th>Japanese</th>
<th>Turkish</th>
<th>French</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># of Users</strong></td>
<td>64,927,267</td>
<td>22,456,938</td>
<td>20,279,402</td>
<td>10,402,846</td>
<td>10,743,511</td>
</tr>
<tr>
<td><strong>Follow ratio to its own language</strong></td>
<td>64%</td>
<td>58%</td>
<td>89%</td>
<td>57%</td>
<td>51%</td>
</tr>
<tr>
<td><strong>Follow ratio to English</strong></td>
<td>31%</td>
<td>36%</td>
<td>9%</td>
<td>39%</td>
<td>44%</td>
</tr>
<tr>
<td><strong># of Nodes for DOS</strong></td>
<td>60,708,434</td>
<td>21,152,308</td>
<td>19,682,116</td>
<td>9,638,906</td>
<td>8,964,888</td>
</tr>
<tr>
<td><strong># of Edges for DOE</strong></td>
<td>2,266,838,184</td>
<td>1,098,723,999</td>
<td>1,394,986,423</td>
<td>271,513,323</td>
<td>177,419,512</td>
</tr>
<tr>
<td><strong>Average Degree</strong></td>
<td>37.33</td>
<td>51.94</td>
<td>70.87</td>
<td>28.16</td>
<td>19.79</td>
</tr>
<tr>
<td><strong>Degree of Separation (Average path length between two users)</strong></td>
<td>4.625</td>
<td>4.253</td>
<td>4.014</td>
<td>4.340</td>
<td>4.699</td>
</tr>
<tr>
<td><strong>Diameter (Lower bound value)</strong></td>
<td>42</td>
<td>23</td>
<td>27</td>
<td>39</td>
<td>22</td>
</tr>
</tbody>
</table>
Summary and Call for Collaboration

  - Project information
  - Source code distribution / VM Image
  - Documentation

- **Call for Collaboration**
  - Sharing our whole Twitter network and all the user profile as of 2012/10
  - More application-driven research and development in the ScaleGraph project
Supplemental materials
ScaleGraph Software Stack

User Program

Graph Algorithm

XPregel Graph Processing System
BLAS for Sparse Matrix
File IO

Third Party Library (ARPACK, METIS)
ScaleGraph Base Library
X10 Standard Lib

Team

MPI

X10 & C++
Degree of Separation
HyperANF – Strong Scaling Performance Analysis

Strong Scaling (Scale 25)

Strong Scaling (Scale 28)

(Scale = 25, B=7, R-MAT Graph, 33.33 Million Vertices and 536 Million Edges
Scale = 28, R-MAT Graph, 268.43 Million Vertices and 4.295 Billion Edges)
HyperANF – Weak Scaling Performance Analysis

Weak Scaling (Scale 22)

Elapsed Time (s)

0 20 40 60 80 100 120 140 160

1 2 4 8 16 32 64 128

Number of Nodes

Weak Scaling (Scale 25)

Elapsed Time (s)

0 100 200 300 400 500 600 700 800

1 2 4 8 16 32 64

Number of Nodes

Scale 29 for 128 nodes

Scale 31 for 64 nodes

RMAT
Random
Weak Scaling – Profiling (Scale 25 per node, RMAT)
Workflow for Temporal Analysis (1/3)

- Convert Twitter user profile and network files to input format for WebGraph API

1. User Profile (xml)
   - get ID
   - 4 nodes, 20 min.
   - Hadoop Serialized 91 GB

2. ID list sorted by creation time
   - assign all IDs to new serial IDs
   - 8 nodes, 1 hour.
   - .gz 10 GB

3. Follower Friend Network (adjacency list)
   - replace destination ID to new serial ID
   - 8 nodes, 1 hour.
   - .gz 231 GB

4. Intermediate Graph data (edge list)
   - replace source ID to new serial ID
   - 8 nodes, 1 hour.
   - Numbering Graph data (edge list)
   - .gz 113 GB

- Not use HDFS. Use only GPFS. Hadoop can read directly gzip files.
Workflow for Temporal Analysis (2/3)

: with Shell Command
remove nodes and edges for timestamp graph (every 3 months)

Input format divided by hadoop reducer

Sequential processing on each timestamp graph

6
merge
merge
merge

Parallel processing every month in one go

Input format for WebGraph API

12/2006 raw text 892KB
12/2006 .gz 300KB
9/2012 raw text 263GB
9/2012 .gz 85GB
...
...
8 nodes 1 hour

Numbering Graph data (edge list)
.gz 113GB
.gz 300KB
.gz 85GB
.decompressed data size 1.5 TB

Total data size: 500GB (every 3 months)
Total data size: 1.5TB (every 3 months)
Workflow for Temporal Analysis (3/3)

Input format for WebGraph API

12/2006 raw text 892KB
...
9/2012 raw text 263GB

compress BVGraph

1 node 7 hours

BVGraph for WebGraph API

2006年 Object 240KB
...
2012年 Object 73GB

Total data size: 1.5TB (every 3 months)

Total data size: 470 GB (every 3 months)

Parallel processing every month in one go

Sequential processing on each timestamp graph

Compute Degree of Separation and Diameter with HyperANF

: with WebGraph API
Workflow: Degree of Separation

- Use HyperANF in WebGraph on TSUBAME 2.0 Fat Node
  - take 16 hours with 1 node (64 cores, 512 GB RAM)

BVGraph for WebGraph API

- 2006年 Object 240KB
- 2012年 Object 73GB

Total data size: 470 GB (every 3 months)