Graph Processing:
Big Data as Large-Scale Graph Analysis on Parallel and Distributed Platforms

Tim Hegeman and Wing-Lung Ngai.

Yong Guo and Stijn Heldens.

Ana Lucia Varbanescu.

LDBC Graphalytics co-authored with LDBC team:
Arnau Prat-Pérez, Thomas Manhardt, Siegfried Depner, Hassan Chafi, Mihai Capotă, Narayanan Sundaram, Michael Anderson, Ilie Gabriel Tănase, Yinglong Xia, Lifeng Nai, Peter Boncz

Prof. dr. ir. Alexandru Iosup
Massivizing Computer Systems
### Graph Processing in Academic Publications

Title Keywords in Computer Systems Conferences (CCGRID, CLOUD, Cluster, HPDC, ICPP, IPDPS, NSDI, OSDI, SC, SIGMETRICS, SoCC, SOSP, ) and Journals (CCPE, FGCS, JPDC, TPDS)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cloud</td>
<td>-</td>
<td>cloud</td>
<td>0</td>
<td>cloud</td>
<td>0</td>
<td>cloud</td>
<td>0</td>
<td>cloud</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>data</td>
<td>-</td>
<td>data</td>
<td>0</td>
<td>data</td>
<td>0</td>
<td>data</td>
<td>0</td>
<td>data</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>graph</td>
<td>-</td>
<td>dynamic</td>
<td>-3</td>
<td>management</td>
<td>-1</td>
<td>architecture</td>
<td>+1</td>
<td>framework</td>
<td>-2</td>
</tr>
<tr>
<td>30</td>
<td>machine</td>
<td>-</td>
<td>machine</td>
<td>0</td>
<td>architecture</td>
<td>+1</td>
<td>analysis</td>
<td>-3</td>
<td>mapreduce</td>
<td>+4</td>
</tr>
<tr>
<td>31</td>
<td>virtual</td>
<td>-</td>
<td>architecture</td>
<td>+1</td>
<td>mapreduce</td>
<td>+2</td>
<td>center</td>
<td>+4</td>
<td>core</td>
<td>+6</td>
</tr>
<tr>
<td>32</td>
<td>architecture</td>
<td>-</td>
<td>graph</td>
<td>-5</td>
<td>machine</td>
<td>-2</td>
<td>machine</td>
<td>0</td>
<td>model</td>
<td>-10</td>
</tr>
<tr>
<td>33</td>
<td>time</td>
<td>-</td>
<td>mapreduce</td>
<td>+7</td>
<td>time</td>
<td>-1</td>
<td>graph</td>
<td>+1</td>
<td>center</td>
<td>-2</td>
</tr>
<tr>
<td>34</td>
<td>approach</td>
<td>-</td>
<td>time</td>
<td>-1</td>
<td>graph</td>
<td>-2</td>
<td>mapreduce</td>
<td>-3</td>
<td>virtual</td>
<td>-10</td>
</tr>
<tr>
<td>35</td>
<td>center</td>
<td>-</td>
<td>center</td>
<td>0</td>
<td>center</td>
<td>0</td>
<td>simulation</td>
<td>-1</td>
<td>multicore</td>
<td>+7</td>
</tr>
<tr>
<td>36</td>
<td>optimization</td>
<td>-</td>
<td>approach</td>
<td>-2</td>
<td>simulation</td>
<td>+1</td>
<td>heterogeneous</td>
<td>+2</td>
<td>graph</td>
<td>-3</td>
</tr>
</tbody>
</table>

**Warning:** Linear regressions may be deceiving.
Graphs Are at the Core of Our Society: The LinkedIn Example

The State of LinkedIn

A very good resource for matchmaking workforce and prospective employers

Vital for your company’s life, as your Head of HR would tell you

Vital for the prospective employees

Tens of “specialized LinkedIns”: medical, mil, edu, science, ...

150,000,000 registered members (Q1 ’12)

Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
LinkedIn’s Service Analysis

By processing the graph: opinion mining, hub detection, etc. Always new questions about whole dataset.

Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
LinkedIn’s Service Analysis

Periodic and/or continuous full-graph analysis

Sources: Vincenzo Cosenza, The State of LinkedIn, http://vincos.it/the-state-of-linkedin/
Why Is LinkedIn Doing Big Data Processing?
LinkedIn Is Part of the “Data Deluge” (Volume)

Data Deluge = data generated by humans and devices (IoT)
- Interacting
- Understanding
- Deciding
- Creating

Sources: IDC, EMC.
The Fourth Paradigm: The Why (An Anecdotal Example)

The Overwhelming Growth of Knowledge

“When 12 men founded the Royal Society in 1660, it was possible for an educated person to encompass all of scientific knowledge. […]”

Professionals already know they don’t know [it all]

In the last 50 years, such has been the pace of scientific advance that even the best scientists cannot keep up with discoveries at frontiers outside their own field.”

Tony Blair, PM Speech, May 2002

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>289,751</td>
<td>336,858</td>
<td></td>
</tr>
<tr>
<td>France</td>
<td>203,814</td>
<td>232,058</td>
<td></td>
</tr>
<tr>
<td>Canada</td>
<td>168,331</td>
<td>166,216</td>
<td></td>
</tr>
<tr>
<td>Italy</td>
<td>122,398</td>
<td>147,023</td>
<td></td>
</tr>
<tr>
<td>Switzerland</td>
<td>57,664</td>
<td>66,761</td>
<td></td>
</tr>
<tr>
<td>Netherlands</td>
<td>83,600</td>
<td>92,526</td>
<td></td>
</tr>
</tbody>
</table>

Data: King, The scientific impact of nations, Nature’04.
The Fourth Paradigm: The What

From Hypothesis to Data Exploration (and Back)

1. Thousand years ago:
   science was empirical describing natural phenomena

2. Last few hundred years:
   theoretical branch using models, generalizations

3. Last few decades:
   a computational branch simulating complex phenomena

4. Today (the Fourth Paradigm):
   data exploration unify theory, experiment, and simulation
   • Data captured by instruments or generated by simulator
   • Processed by software
   • Information/Knowledge stored in computer
   • Scientist analyzes results using data management and statistics

The Fourth Paradigm is suitable for professionals who already know they don’t know [enough to formulate good hypotheses], yet need to deliver quickly

Why Is LinkedIn Doing Graph Processing?

... and What Else Can We Use Graph Processing For?
Processing large-scale graphs, at large

Social network
~1 billion vertices
~100 billion connections

Web graph
~50 billion pages
~1 trillion hyperlinks

Brain network
~100 billion neurons
~100 trillion connections

Source: Smith, CHI’10; Blog webpage; Gigandet et al., PLoS ONE 3(12)]
What is a graph?

A graph/network is a collection of vertices/nodes connected by edges/links.
What is a graph?

Edges may be **directed** (from A to B)

or **undirected** (between A and B)

Vertices and edges may have **properties**

- name: “Jane”
  - bio: “…”
- type: “follows”
  - since: 2016-01-01
- name: “John”
  - joined: 2015-10-05
Example 1/3: Social networks
Example 2/3: Financial transactions
Example 3/3: Navigation systems [1/2]
Example 3/3: Navigation systems [2/2]
What is a graph?

A graph represents a collection of entities and relationships/interactions.
What is graph processing?

Graph processing enables **analytics with a graph as input**

Graph algorithms typically...
- ... extract information from the graph structure
- ... touch a large portion of the input graph
- ... are executed offline (i.e., batch processing)
Typical graph processing applications

Identifying communities of entities
Typical graph processing applications

Identifying “important” entities
Typical graph processing applications

Predicting or recommending new relationships (product recommendations, friends-of-friends)
Ahm… This Is Too Abstract… Any Example?
Ex.: Identifying Connected Components

Goal: assign identical labels to all vertices in a component
Ex.: Identifying Connected Components

Goal: assign identical labels to all vertices in a component
Ex.: Identifying Connected Components

The algorithm, in one of the many possible implementations:

Step 0: Assign a **unique ID as initial value** to every vertex

Step 1+: Iterate until completion, for every vertex:

1. Receive values of neighbours from previous iteration
2. New value is the **smallest value among current value of self and received value from any neighbour**
3. Send new value to neighbours for next iteration
Ex.: Identifying Connected Components

Step 0: Initialize every vertex with a unique ID as value
Ex.: Identifying Connected Components

Step 1.1: Receive values of neighbours
Ex.: Identifying Connected Components

Step 1.2: Pick smallest value (No messages, so no change)
Ex.: Identifying Connected Components

Step 1.3: Send own value to neighbours
Ex.: Identifying Connected Components

Step 2.1: Receive values of neighbours
Ex.: Identifying Connected Components

Step 2.2: Pick smallest value
Ex.: Identifying Connected Components

Step 2.3: Send own value to neighbours
Ex.: Identifying Connected Components

Step 3.1: Receive values of neighbours

1 1 2
1 2
1

2 2
2

3
Ex.: Identifying Connected Components

Step 3.2: Pick smallest value
Ex.: Identifying Connected Components

Step 3.3: Send own value to neighbours
Ex.: Identifying Connected Components

Step 4.1: Receive values of neighbours
Ex.: Identifying Connected Components

Step 4.2: Pick smallest value
Ex.: Identifying Connected Components

Algorithm completed (because there was no change)!
How to Do Graph Processing (Analysis)?
How to Do Graph Analysis? Naïve Attempt

Straightforward implementation of any graph algorithm using a programming language like Java, on your own machine.

- Compiles!
- Does not scale!
How to do Graph Analysis? Make It A System!

A Graph Processing Platform

ETL (Extraction, Transf, Loading)

Active Storage (filtering, compression, replication, caching)

Distribution to processing platform

Algorithm

Interactive processing not considered in this presentation. Streaming not considered in this presentation.
Distributed Big Data: The Promise of Performance

A Graph Processing Platform

Ideally, distributed N cores/disks $\rightarrow$ Nx faster

(Delivering: compression, replication, caching)

Distribution to processing platform

Ideally, Parallel/Distributed/Heterogeneous

Ideally, distributed N cores/disks $\rightarrow$ Nx faster

Interactive processing not considered in this presentation.
Streaming not considered in this presentation.
How does it work at scale?

Pregel in a nutshell:

Step 0: Assign an initial value to every vertex

Step 1+: Iterate until completion, for every vertex:

1. Receive messages from neighbours from previous iteration
2. Compute new value as function of own value and incoming messages
3. Send messages to neighbours for next iteration
How does it work at scale?

Under the hood of Pregel systems:

– Vertices are distributed across machines in a cluster
– Machines synchronize between iterations to exchange messages
– Computation for all vertices can be done in parallel
How does it work at scale?

Machines

Vertices
So…We’re Done!
Your network is so large...

Sorry, but your network is too large to be computed, we are working to increase the limit, stay tuned!
What would you do to solve this?

Data-intensive workload
10x graph size $\Rightarrow$ 100x—1,000x slower

Compute-intensive workload

unfriendly graphs $\Rightarrow$ ??x slower

Supporting multiple users
10x number of users $\Rightarrow$ ?????x slower

Pick the right system, of course!
Matt Turck’s Big Data Landscape 2016 (zoom in on a part of the whole picture)

Ok, which one of these systems?
We Need an Ecosystem Navigator

* Plus Zookeeper, CDN, etc.
Graph processing systems

- Systems for graph processing
- Separate users from backends
- Think Giraph

- Use existing distributed systems
- Mapping is difficult
- Parallelism is “free”
- Think Hadoop/Spark

- Specify application
- Choose the hardware
- Implement & optimize
- Think Graph500 performers

Graph processing systems

- Dedicated Systems
- Custom Systems
- Generic Systems

Development Effort

Performance
System diversity (CPU-based)
System diversity (GPU-enabled)

- medusa-gpu: Medusa: Simplified Graph Processing on GPUs
- mapgraph Beta: Massively Parallel Graph processing on GPUs
- Gunrock: High-performance Graph Primitives on GPU
- NetSysLab
- TOTEM
- VertexAPI2
The Ecosystem Navigator for Graph Processing Platforms

Which platforms perform well?

What to tune?

What to re-design?
The Ecosystem Navigator for Graph Processing Platforms Is A…

Benchmark!
What Is the Performance of Graph Processing Platforms?

- **Graph500**
  - Single application (BFS), Single class of synthetic datasets. @ISC16: future diversification.

- Few existing platform-centric comparative studies
  - Prove the superiority of a given system, limited set of metrics

- **GreenGraph500, GraphBench, XGDBench**
  - Issues with representativeness, systems covered, metrics, ...
What Is the Performance of Graph Processing Platforms?

Graphalytics = comprehensive benchmarking suite for graph processing across many platforms

http://ldbcouncil.org/ldbc-graphalytics

http://graphalytics.ewi.tudelft.nl/
Graphalytics, in a nutshell

- An LDBC benchmark
- Advanced benchmarking harness
- Many classes of algorithms used in practice
- Diverse real and synthetic datasets
- Diverse set of experiments representative for practice
- Renewal process to keep the workload relevant
- Extended toolset for manual choke-point analysis
- Enables comparison of many platforms, community-driven and industrial

http://ldbcouncil.org/ldbc-graphalytics
Graphalytics = Benchmarking Harness

Iosup et al. LDBC Graphalytics: A Benchmark for Large Scale Graph Analysis on Parallel and Distributed Platform, PVLDB’16.
Graphalytics = Representative Classes of Algorithms and Datasets

- 2-stage selection process of algorithms and datasets

<table>
<thead>
<tr>
<th>Class</th>
<th>Examples</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Statistics</td>
<td>Diameter, Local Clust. Coeff., PageRank</td>
<td>20</td>
</tr>
<tr>
<td>Graph Traversal</td>
<td>BFS, SSSP, DFS</td>
<td>50</td>
</tr>
<tr>
<td>Connected Comp.</td>
<td>Reachability, BiCC, Weakly CC</td>
<td>10</td>
</tr>
<tr>
<td>Community Detection</td>
<td>Clustering, Nearest Neighbor, Community Detection w Label Propagation</td>
<td>5</td>
</tr>
<tr>
<td>Other</td>
<td>Sampling, Partitioning</td>
<td>&lt;15</td>
</tr>
</tbody>
</table>

+ property/weighted graphs: Single-Source Shortest Paths (~35%)

Graphalytics = Distributed Graph Generation with DATAGEN

- Rich set of configurations
- More diverse degree distribution than Graph500
- Realistic clustering coefficient and assortativity
Graphalytics = Diverse Set of Automated Experiments

<table>
<thead>
<tr>
<th>Category</th>
<th>Experiment</th>
<th>Algo.</th>
<th>Data</th>
<th>Nodes/Threads</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td>Dataset variety</td>
<td>BFS, PR</td>
<td>All</td>
<td>1</td>
<td>Run, norm.</td>
</tr>
<tr>
<td></td>
<td>Algorithm variety</td>
<td>All</td>
<td>R4(S), D300(L)</td>
<td>1</td>
<td>Runtime</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Vertical vs. horiz.</td>
<td>BFS, PR</td>
<td>D300(L), D1000(XL)</td>
<td>1—16/1—32</td>
<td>Runtime, S</td>
</tr>
<tr>
<td></td>
<td>Weak vs. strong</td>
<td>BFS, PR</td>
<td>G22(S)—G26(XL)</td>
<td>1—16/1—32</td>
<td>Runtime, S</td>
</tr>
<tr>
<td><strong>Robustness</strong></td>
<td>Stress test</td>
<td>BFS</td>
<td>All</td>
<td>1</td>
<td>SLA met</td>
</tr>
<tr>
<td></td>
<td>Variability</td>
<td>BFS</td>
<td>D300(L), D1000(L)</td>
<td>1/16</td>
<td>CV</td>
</tr>
<tr>
<td></td>
<td>Self-Test</td>
<td>--</td>
<td>Datagen</td>
<td>1—16</td>
<td>Runtime</td>
</tr>
</tbody>
</table>
Graphalytics = Modern Software Engineering Process

https://github.com/ldbc/ldbc_graphalytics

Graphalytics code reviews
Internal release to LDBC partners (first, Feb 2015; last, Feb 2016)
Public release, announced first through LDBC (Apr 2015)
First full benchmark specification, LDBC criteria (Q1 2016)

Jenkins continuous integration server
SonarQube software quality analyzer
Graphalytics has been implemented for 3 community-driven platforms (Giraph, GraphX, PowerGraph) and 3 industry-driven platforms (PGX, GraphMat, OpenG).
Results: Experimental Setup (2)

All experiments performed by TU Delft on DAS-5 (Distributed ASCI Supercomputer, the Dutch national supercomputer for Computer Science research).

Environment: 1 machine (64GB, 2x8 cores)
The Platform Has Large Impact

PageRank on Datagen-300

Throughput [EPS]

2 orders of magnitude difference due to platform

Better
The Algorithm Has Large Impact

PageRank on DG-300
Community Detection on DG-300

GraphMat fastest for PR, slow for CD

Failure

Better
The Dataset Has Large Impact

BFS on KGS

BFS on cit-Patents

Giraph & GraphMat better for KGS
The Dataset Has Large Impact

BFS on KGS

BFS on cit-Patents

OpenG & PGX better for cit-Patents
The Dataset Has Large Impact

BFS on KGS

BFS on cit-Patents

OpenG & PGX benefit from small output
Giraph & GraphMat benefit from small diameter

Better
Lessons learned

Performance of graph processing is a non-trivial function of (Platform, Algorithm, Dataset, …), the **P-A-D triangle** (Ana’s term)

Understanding performance requires in-depth analysis
We are building tools for manual/automated choke-point analysis

All current platforms can also have drawbacks
Ease-of-use/programmability of a platform is very important
Significant knowledge required to tune a system
<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Competition</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-01-30</td>
<td>v0.2.8</td>
<td>Beta Competition: R2</td>
<td>Refine standard benchmark definition + cost-efficiency + performance</td>
</tr>
<tr>
<td>2017-03-13</td>
<td>v0.2.9</td>
<td>Beta Competition: R3</td>
<td>Refine system specification, cost model</td>
</tr>
<tr>
<td>2017-04-10</td>
<td>v0.2.10</td>
<td>Beta Competition: R3</td>
<td>Refine full benchmark definition + scalability + robustness</td>
</tr>
<tr>
<td>2017-05-08</td>
<td>v0.2.11</td>
<td>Beta Competition: R3</td>
<td>Refine competition, auditing Rules</td>
</tr>
<tr>
<td>2017-06-05</td>
<td>v0.2.12</td>
<td>Beta Competition: R3</td>
<td>[reserved slot]</td>
</tr>
<tr>
<td>2017-06-19</td>
<td>v1.0.0</td>
<td>2017, Edition 1: Completed</td>
<td>Internal participation</td>
</tr>
<tr>
<td>2017-06-26</td>
<td>v1.0.0</td>
<td>2017, Edition 2: Started</td>
<td>Global participation</td>
</tr>
</tbody>
</table>
Take-Home Message

Graphalytics: ecosystem navigator for graph processing platforms, with little effort

The P-A-D Triangle:
Performance = f (Algorithm, Dataset, Platform, …)

Towards addressing the graph data deluge with high performance, low development effort