Extreme-scale Graph Analysis on Blue Waters
2016 Blue Waters Symposium

George M. Slota$^{1,2}$, Siva Rajamanickam$^1$, Kamesh Madduri$^2$, Karen Devine$^1$

$^1$Sandia National Laboratories$^a$
$^2$The Pennsylvania State University
www.gmslot.com  gslota@psu.edu

13 June 2016

---

$^a$Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.
About Me

Past:

- PhD in Computer Science & Engineering from Penn State in 2016
- Supported by *Blue Waters Fellowship* in 2014-2015
- Intern at Sandia National Labs from 2013-2016

Present:

- Staff at Sandia National Labs

Future:

- Assistant Professor at Rensselaer Polytechnic Institute
What?
Or, given modern extreme-scale graph-structured datasets (web crawls, brain graphs, human interaction networks) and modern high performance computing systems (Blue Waters), how can we develop a generalized approach to efficiently study such datasets on such systems?
Why?
Why do want to study these large graphs?

**Human Interaction Graphs:**
- Finding hidden communities, individuals, malicious actors
- Observe how information and knowledge propagates

**Brain Graphs:**
- Study the topological properties of neural connections
- Finding latent computational substructures, similarities to other information processing systems

**Web Crawls:**
- Identifying trustworthy/important sites
- Spam networks, untrustworthy sites
Prior Approaches

Can we use them to analyze large graphs on HPC?

- Some limited by shared-memory and/or specialized hardware
- Some run in distributed memory but graph scale is still limited
- Others, graph scale isn’t limiting factor but performance can be
Graph analytics on HPC

So why do we want to run graph analytics on HPC?

- Scalability for analytic performance and graph size
  - Efficient implementations should be limited only by distributed memory capacity
  - Graph500.org - demonstration of performance achievable for irregular computations through breadth-first search (BFS)
- Relative availability of access in academic/research communities
  - Private clusters of various scales, shared supercomputers
  - Access for domain experts, those using analytics on real-world graphs

Can we create an approach that is as simple to use as the aforementioned frameworks but runs on common cluster hardware and gives state-of-the-art performance?
Challenges
This work considers “extreme-scale” graphs – billion+ vertices and up to trillion+ edges.

Processing these graphs requires at least hundreds to thousands of compute nodes or tens of thousands of cores.

Graph analytic algorithms are generally memory-bound instead of compute-bound; in the distributed space, this results in a ratio of communication versus computation that increases with core/node count.
Complexity

- Real-world extreme-scale graphs have similar characteristics: small-world nature with skewed degree distributions
- Small-world graphs are difficult to partition for distributed computation or to optimize in terms of cache due to “too much locality”
- Skewed degree distributions make efficient parallelization and load balance difficult to achieve
- Multiple levels of cache/memory and increasing reliance on wide parallelism for modern HPC systems compounds the above challenges
Approach
Observation: many iterative graph algorithms have similar communication patterns

- (Vanilla) \textit{BFS-like}: frontier expansion, information \textit{pushed} from vertices to adjacencies, volume of data exchanged is \textbf{variable} or fixed across iterations

- (Vanilla) \textit{PageRank-like}: information \textit{pulled} from incoming arcs, either \textbf{fixed} or variable communication pattern in every iteration

We develop optimized skeleton code for these two (or four) patterns, and can use it to fill in analytic-specific details
Analytics Fitting these Patterns

Some examples

**BFS-like:**
- **SCC**: Strongly connected components
- **WCC**: Weakly connected components
- **K-Core**: Iterative approach to find approximate vertex coreness
- **Harmonic Centrality**: Routine for calculating harmonic centrality value of any given vertex

**PageRank-like:**
- **PageRank**: Well-known centrality algorithm
- **Label Propagation**: Community detection algorithm
- **Color Propagation**: Connectivity algorithm for CC, WCC, SCC
Implementation Considerations

Choices, choices, choices ...

**Tradeoffs** (ease of implementation vs. scalability):

- **1D** (vertex-based) vs. **2D** (edge-based) partitioning and graph layout
- **Bulk-synchronous** vs. asynchronous communication
- Programming language and parallel programming model
  - High-level language (e.g., Scala) vs. **C/C++**
  - High-level model (e.g., Spark) vs. MPI-only vs. **MPI+OpenMP**

**Other considerations:**

- In-memory graph representation
  - **Vanilla CRS-like** vs. compressed (e.g., with RLE) adjacencies
- Partitioning strategy (with 1D layout)
  - **Vertex-balanced, Edge-balanced, Random** vs. Explicit partitioning
Performance Results
Experimental Setup
Test systems, Graphs

- **Blue Waters**: dual-socket AMD Interlagos 6276, 16 cores, 64 GB memory
- **Compton cluster**: dual-socket Intel Xeon E5-2670, 16 cores, 64 GB memory

<table>
<thead>
<tr>
<th>Graph</th>
<th>$n$</th>
<th>$m$</th>
<th>$D_{avg}$</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Crawl (WC)</td>
<td>3.6 B</td>
<td>129 B</td>
<td>36</td>
<td>[?]</td>
</tr>
<tr>
<td>R-MAT</td>
<td>3.6 B</td>
<td>129 B</td>
<td>36</td>
<td>[?]</td>
</tr>
<tr>
<td>Rand-ER</td>
<td>3.6 B</td>
<td>129 B</td>
<td>36</td>
<td>Erdös-Rényi</td>
</tr>
<tr>
<td>R-MAT</td>
<td>$2^{25} \cdot 2^{36}$</td>
<td>$2^{29} \cdot 2^{40}$</td>
<td>16-64</td>
<td>[?]</td>
</tr>
<tr>
<td>Rand-ER</td>
<td>$2^{25} \cdot 2^{36}$</td>
<td>$2^{29} \cdot 2^{40}$</td>
<td>16-64</td>
<td>Erdös-Rényi</td>
</tr>
<tr>
<td>Pay</td>
<td>39 M</td>
<td>623 M</td>
<td>16</td>
<td>[?]</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>4.8 M</td>
<td>69 M</td>
<td>14</td>
<td>[?]</td>
</tr>
<tr>
<td>Google</td>
<td>875 K</td>
<td>5.1M</td>
<td>5.8</td>
<td>[?]</td>
</tr>
</tbody>
</table>
Comparison to Distributed Graph Frameworks

Our approach vs. GraphX, PowerGraph, PowerLyra

- Compared GraphX (GX), PowerGraph (PG), and PowerLyra (PL) on 16 nodes of Compton to our code (SRM)
- About $38 \times$ faster on average for PageRank (top), $201 \times$ faster for WCC (bottom) against distributed memory frameworks

![PageRank Speedup vs. GraphX](chart1.png)

![WCC Speedup vs. GraphX](chart2.png)
Weak and Strong Scaling
Label propagation-based analytics

- Strong scaling on *Blue Waters* for label propagation community detection with WC and random graphs
- Weak scaling on *Blue Waters* for label propagation-based algorithm on random graphs and meshes
Performance on WC with 256 node of Blue Waters

How can we improve?

- Perf. units are similar to GTEPS (Giga Traversed Edges Per Second): $$\frac{m \times n_{\text{iter}}}{t \times 10^9}$$

<table>
<thead>
<tr>
<th>Analytic</th>
<th>Time (s)</th>
<th>Perf.</th>
<th>Our evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageRank</td>
<td>87</td>
<td>29.6</td>
<td>😊</td>
</tr>
<tr>
<td>Label Propagation</td>
<td>367</td>
<td>3.5</td>
<td>😞</td>
</tr>
<tr>
<td>WCC</td>
<td>63</td>
<td>2.0</td>
<td>😞</td>
</tr>
<tr>
<td>Harmonic Centrality</td>
<td>46</td>
<td>2.8</td>
<td>😞</td>
</tr>
<tr>
<td>K-core</td>
<td>363</td>
<td>9.6</td>
<td>😞</td>
</tr>
<tr>
<td>Largest SCC</td>
<td>108</td>
<td>2.4</td>
<td>😞</td>
</tr>
<tr>
<td>Overall</td>
<td>1034</td>
<td>7.6</td>
<td>😊</td>
</tr>
<tr>
<td>Graph500 (estimate)</td>
<td>119.2</td>
<td></td>
<td>😊</td>
</tr>
</tbody>
</table>
Possible Future Extensions

- Processing quadrillion-edge (petascale) graphs?
- 10x performance improvement by next year? Direction optimization, asynchronous communication, graph compression, other partitioning strategies
- Identify and implement additional analytics that fit push/pull/fixed/variable communication patterns
- Open-source code
  - Contact gslota@psu.edu for current version
Acknowledgments

- **Sandia and FASTMATH**
  - This research is supported by NSF grants CCF-1439057 and the DOE Office of Science through the FASTMath SciDAC Institute. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

- **Blue Waters Fellowship**
  - This research is part of the Blue Waters sustained petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070, ACI-1238993, and ACI-1444747) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana Champaign and its National Center for Supercomputing Applications.

- **Kamesh Madduri’s CAREER Award**
  - This research was also supported by NSF grant ACI-1253881.
Conclusions and Thanks!

- Graphs are ubiquitous, massive, and complex: scalability and efficiency are important considerations for real-world analytics
- We identified and optimized several distinct communication patterns that fit large classes of graph algorithms
- Implemented several algorithms fitting these patterns and demonstrated scalability up to 131k cores of Blue Waters
- Demonstrated $26-1573\times$ speedup vs. GraphX on 256 cores of Compton

Thank you! Questions? gslota@psu.edu, www.gmslota.com