A Case Study of Complex Graph Analysis in Distributed Memory: Implementation and Optimization

George M. Slota[1,2], Siva Rajamanickam[1], Kamesh Madduri[2]

[1]Sandia National Laboratories[a]
[2]The Pennsylvania State University
www.gmslota.com gslota@psu.edu

24 May 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.

Presentation Overview

- Motivating massive-scale distributed-memory analytics
- Parallel implementations of six analytics for processing massive (hyperlink) graphs
  - PageRank, Harmonic centrality, finding largest SCC, WCC decomposition, approximate K-core computation, community structure detection
- Common optimizations
- Performance results on the Blue Waters supercomputer

Graphs are ...

- **Everywhere**
  - Internet, Social networks, Biology, Scientific computing
- **Massive**
  - Internet: e.g., Google crawls trillions of pages, index size is over 100 PB
  - Social networks: e.g., Facebook has 1.6 B active users
  - Neuroscience: e.g., human brain has 86 B neurons
- **Complex**
  - Real-world graph characteristics impose computational challenges: skewed degree distributions (power law, irregular) and small-world nature
  - Many interesting graph problems are NP-complete

Parallel platforms are ...

- **Everywhere**
  - mobile phones to supercomputers
- **Powerful**
  - Energy-efficient multicore and manycore processors
  - Aggregate memory capacities and bandwidths growing at an exponential rate
- **Challenging to program**
  - Using compute resources efficiently
  - Load balancing, memory locality
  - Reducing inter-node communication

Questions Motivating our Work

- **Q**: How efficiently can we analyze **large publicly-available graph instances** on **multi-node** platforms?
  - e.g., 2012 Web Data Commons (WDC12) hyperlink graph: 3.6 billion vertices (URLs) and 129 billion edges (directed links)
- **Q**: What **optimizations strategies and abstractions** are common to multiple graph analytics?
  - Best practices for distributed-memory graph analytics
  - Guide and simplify new implementations
- **Q**: Can we write **simple, yet high-performance** code?
  - 1000s of lines of code; within small factor of Graph500 BFS performance

Graphs+HPC, The State-of-the-art

- Many publicly-available frameworks exist for graph analysis
- Shared-memory libraries (Galois, Ligra, etc.) cannot process 100 B+ edge graphs (yet)
- External memory frameworks (e.g., FlashGraph) might require specialized hardware (SSD arrays) and big shared-memory nodes (512 GB+)
- MapReduce-like frameworks (e.g., Giraph) are limited by disk I/O and untuned inter-node communication
- Several distributed-memory graph frameworks (e.g., GraphLab and its derivatives, GraphX) fail to the process WDC12 graph

Challenges and Research Goals

- Skewed vertex degree distributions of graphs make distributed-memory parallelization difficult
  - Use hybrid programming models to fully exploit shared memory on a node
  - Investigate several distributed-memory graph layout alternatives
- Optimizations may be specialized for graph analytic (e.g., BFS, SSSP) and not portable across platforms
  - Investigate algorithms for multiple analytics
  - Optimize end-to-end running time (including parallel I/O)

Talk Outline

- Motivating massive-scale distributed-memory analytics
- Parallel implementations of six analytics for processing massive (hyperlink) graphs
- Performance results
- 2012 Web Data Commons graph analysis

Massive Distributed-memory Graph Analytics

- Optimized implementations of six analytics
- End-to-end tuning with almost no serial routines
- Hybrid parallelism with MPI and OpenMP
- Parallel I/O
- Compact and efficient: $\sim$2,000 total lines of code

Graph Analytics Considered

- Centrality: PageRank iterations, Harmonic centrality
- Connectivity: finding the largest strongly connected component (SCC), weakly connected component decomposition (WCC)
- Approximate K-core decomposition, or computing coreness upper bound for every vertex
- Global community structure detection using label propagation
- We apply all these analytics to the 2012 Web Data Commons graph (3.6 billion vertices, 129 billion edges)
- Though optimized for large scale, also efficient at small scale

Design Tradeoffs and Considerations

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

Graph Representation

Data	Size	Description
n_global	1	Global vertex count
m_global	1	Global edge count
n_loc	1	Task-local vertex count
n_gst	1	Ghost vertex count
m_out	1	Task-local out-edges count
m_in	1	Task-local in-edges count
out_edges	m_out	Array of out-edges
out_indexes	n_loc	Start indices for local out-edges
in_edges	m_in	Array of in-edges
in_indexes	n_loc	Start indices for local in-edges
map	n_loc+n_gst	Global to local id hash table
unmap	n_loc+n_gst	Array for local to global id conv.
tasks	n_gst	Array storing owner of ghost vertices

Optimizing Inter-process Communication

Observation: many iterative graph algorithms have similar communication patterns

- (Vanilla) *BFS-like*: frontier expansion, information *pushed* from vertices to adjacencies, volume of data exchanged is **variable** or fixed across iterations
- (Vanilla) *PageRank-like*: information *pulled* from incoming arcs, either **fixed** or variable communication pattern in every iteration

We use optimized skeleton code for these two (or four) patterns, fill in analytic-specific details

Analytic-specific Details

BFS-like:

- **SCC**: 1st stage of MULTISTEP-SCC (FW-BW algorithm)
- **WCC**: 1st stage of MULTISTEP-WCC
- (Approx.) **K-Core**: Iterative searches to find upper bound power-of-2 coreness
- **Harmonic Centrality**: Routine for calculating centrality value of any given vertex

PageRank-like:

- **PageRank**: Standard iterative algorithm
- **Label Propagation**: Community detection algorithm
- **WCC**: 2nd stage of MULTISTEP-WCC

BFS-like Algorithmic Pattern

procedure BFS-LIKE($G(V, E)$) ▷ Task Parallel
    for all $v \in V$ **do** ▷ Thread Parallel
        $D(v) \leftarrow \text{init}()$
        if addToQ(v) **then**
            $Q_{next} \leftarrow \langle v, D(v) \rangle$
    while any $Q_{next} \neq \varnothing$ **do**
        $\langle Q, D \rangle \leftarrow \text{AllToAllExchange}(Q_{next})$ ▷ Thread Parallel
        $Q_{next} \leftarrow \varnothing$
        for all $v \in Q$ **do** ▷ Thread Parallel
            for all $\langle v, u \rangle \in E$ **do**
                $D(u) \leftarrow \text{update}()$
                if addToQ(u) **then**
                    $Q_{next} \leftarrow \langle u, D(u) \rangle$
    return D

BFS-like Algorithmic Pattern

procedure BFS-LIKE($G(V, E)$) ▷ Task Parallel
  for all $v \in V$ **do** ▷ Thread Parallel
    $D(v) \leftarrow \text{init}()$
    if addToQ(v) **then**
      $Q_{next} \leftarrow \langle v, D(v) \rangle$
  while any $Q_{next} \neq \varnothing$ **do**
    $\langle Q, D \rangle \leftarrow$ **AllToAllExchange(Q_{next})** ▷ Thread Parallel
    $Q_{next} \leftarrow \varnothing$
    for all $v \in Q$ **do** ▷ Thread Parallel
      for all $\langle v, u \rangle \in E$ **do**
        $D(u) \leftarrow$ **update()**
        if addToQ(u) **then**
          $Q_{next} \leftarrow \langle u, D(u) \rangle$
  return D

PageRank-like Algorithmic Pattern

```
procedure PageRank-like(G(V, E))                          ▷ Task Parallel
    for all v ∈ V do                                      ▷ Thread Parallel
        D(v) ← init()
        if addToQ(v) then
            Q_next ← ⟨v, D(v)⟩
    while any Q_next ≠ ∅ do
        ⟨Q, D⟩ ← AllToAllExchange(Q_next)                 ▷ Thread Parallel
        Q_next ← ∅
        for all v ∈ Q do                                  ▷ Thread Parallel
            for all ⟨v, u⟩ ∈ E do
                D(v) ← update()
            if addToQ(v) then
                Q_next ← ⟨v, D(v)⟩
    return D
```

Talk Outline

- Motivating massive-scale distributed-memory analytics
- Parallel implementations of six analytics for processing massive (hyperlink) graphs
- Performance results
- 2012 Web Data Commons graph analysis

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

Graph	n	m	D_{avg}	Source
Web Crawl (WC)	3.6 B	129 B	36	[Meusel et al., 2015]
R-MAT	3.6 B	129 B	36	[Chakrabarti et al., 2004]
Rand-ER	3.6 B	129 B	36	Erdös-Rényi
R-MAT	2^{25}-2^{32}	2^{29}-2^{36}	16	[Chakrabarti et al., 2004]
Rand-ER	2^{25}-2^{32}	2^{29}-2^{36}	16	Erdös-Rényi
Pay	39 M	623 M	16	[Meusel et al., 2015]
LiveJournal	4.8 M	69 M	14	[Leskovec et al., 2009]
Google	875 K	5.1M	5.8	[Leskovec et al., 2009]

End-to-end Analysis

256 nodes of Blue Waters

- Executed all six analytics on WC (with three partitioning strategies) and synthetic (R-MAT, Rand-ER) graphs of the same size
- With vertex block ($\frac{n}{p}$) and edge block ($\frac{m}{p}$) partitioning strategies, cumulative time on WC is about 20 minutes ($+$ 3 minutes for I/O and preprocessing)
- We use vertex block ($\frac{n}{p}$) partitioning for R-MAT and Rand-ER

	Execution time in seconds				
	WC			R-MAT	Rand-ER
Analytic Partitioning	$\frac{n}{p}$	$\frac{m}{p}$	Rand	$\frac{n}{p}$	$\frac{n}{p}$
PageRank (20 iter)	**87**	111	227	125	121
Label Propagation (10 iter)	400	435	**367**	993	992
WCC	88	**63**	112	68	77
Harmonic Centrality (1 iter)	54	**46**	101	252	84
K-core ($\approx$ 27 BFS'es)	445	**363**	583	579	481
Largest SCC ($\approx$ 2 BFS'es)	184	**108**	184	89	83

WC performance rates

256 nodes of Blue Waters, best partitioning strategy chosen

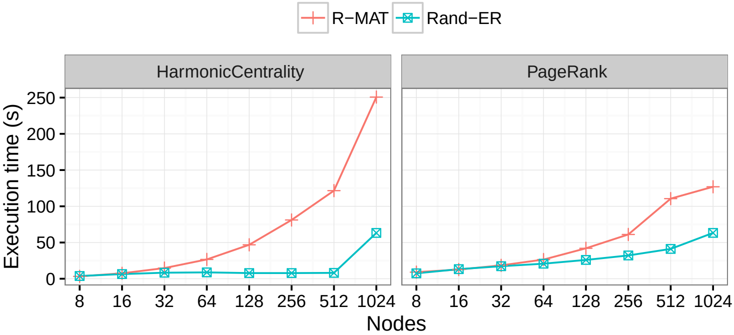
- Perf. units are similar to GTEPS (Giga Traversed Edges Per Second): $\frac{m * n_{\text{iter}}}{t \times 10^9}$

Analytic	Time (s)	Perf.	Our evaluation
PageRank	87	29.6	🙂
Label Propagation	367	3.5	😐
WCC	63	2.0	🙁
Harmonic Centrality	46	2.8	🙁
K-core	363	9.6	😐
Largest SCC	108	2.4	🙁
Overall	1034	7.6	😐
Graph500 (estimate)		119.2	🙂

Weak Scaling on Synthetic Graphs

Blue Waters: 8 to 1024 nodes

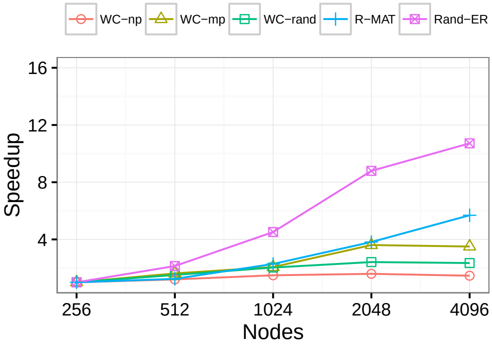
- With vertex block $(\frac{n}{p})$ partitioning
- 2^{22} vertices per node and 2^{26} edges per compute node



Label Propagation: Strong Scaling Results

Blue Waters: 256 to 4096 nodes

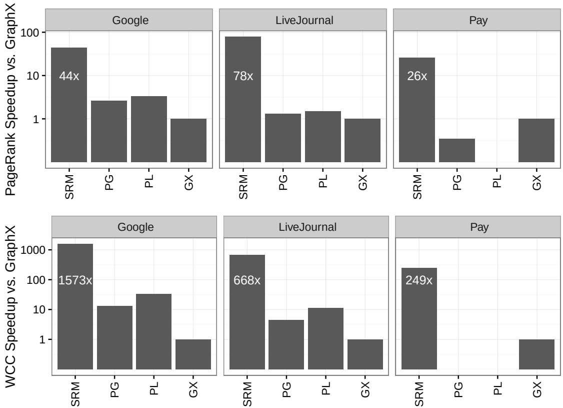
- PageRank-like in general strong scales nicely; BFS-like is more dependent on graph structure (high number of synchronizations and low computation per iteration)



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× faster on average for PageRank (top), 201× faster for WCC (bottom) against distributed memory frameworks



Talk Outline

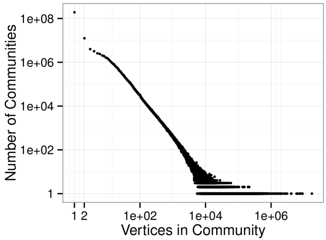
- Motivating massive-scale distributed-memory analytics
- Parallel implementations of six analytics for processing massive (hyperlink) graphs
- Performance results
- 2012 Web Data Commons graph analysis

Community Structure of WC

- Used label propagation to identify disjoint communities
- Community size distribution appears to follow a heavy-tailed power law

Largest Communities (numbers in millions)

Size	m_{comm}	m_{cut}	Rep. Page
112	2126	32	`YouTube`
18	548	277	`Tumblr`
9	516	84	`Creative Commons`
8	186	85	`WordPress`
7	57	83	`Amazon`
6	41	21	`Flickr`



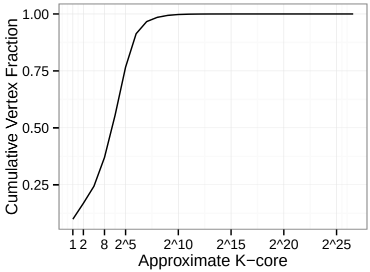
Centrality Measurements of Web Crawl

- Determined the top 10 web pages according to different centrality indices
- Similar to results found in prior work using smaller host-level graph [Meusel et al., 2014]
- Note that out degree is meaningless as a centrality index

Out-degree	**In-degree**	**PageRank**	**Harmonic Centrality**
photoshare.ru/..	youtube.com	youtube.com	wordpress.org
dvderotik.com/..	wordpress.org	youtube.com/t/..	twitter.com
zoover.be/..	youtube.com/t/..	youtube.com/testtube	twitter.com/privacy
cran.r-project.org/..	youtube.com/..	youtube.com/t/..	twitter.com/about
cran.rakanu.com/..	youtube.com/t/..	youtube.com/t/..	twitter.com/tos
linkagogo.com/..	youtube.com/..	tumblr.com	twitter.com/account/..
cran.r-project.org/..	youtube.com/t/..	google.com/intl/en/..	twitter.com/account/..
fussballdaten.de/..	gmpg.org/xfn/11	wordpress.org	twitter.com/about/resources
fussballdaten.de/..	google.com	google.com/intl/..	twitter.com/login
fussballdaten.de/..	google.com/intl/..	google.com	twitter.com/about/contact

Approximate K-core Decomposition of WC

- We estimate coreness upper bound of every vertex
- At least 75% of the vertices have coreness value less than 32, only 0.5% have a coreness greater than 1024



Possible Future Extensions

Beat us if you can!

- Processing **quadrillion-edge** (petascale) graphs?
- $10\times$ performance improvement (20 min to 2 min) by next IPDPS? 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 code

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 analytics
- We identified two distinct communication patterns that fit a large class of graph algorithms
- Implemented several algorithms fitting these patterns and demonstrated scalability up to 65k cores of *Blue Waters*
- Analyzed the 2012 Web Data Commons hyperlink graph
- Demonstrated 26-1573$\times$ speedup vs. GraphX on 256 cores of *Compton* with graphs less than 0.5% the size of the Web Crawl

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

Bibliography I

M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in Twitter: The million follower fallacy. In *Proc. Int'l. Conf. on Weblogs and Social Media (ICWSM)*, 2010.

Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *Proc. Int'l. Conf. on Data Mining (SDM)*, 2004.

J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

Robert Meusel, Sebastiano Vigna, Oliver Lehmberg, and Christian Bizer. Graph structure in the web - revisited: A trick of the heavy tail. In *Proc. WWW*, 2014.

Robert Meusel, Sebastiano Vigna, Oliver Lehmberg, and Christian Bizer. The graph structure in the web - analyzed on different aggregation levels. *J. Web Sci.*, 1(1):33–47, 2015.