Massively Parallel Graph Analytics

Supercomputing for large-scale graph analytics

George M. Slota[1,2,3] Kamesh Madduri[1]
Sivasankaran Rajamanickam[2]

[1]Penn State University, [2]Sandia National Laboratories, [3]Blue Waters Fellow
gslota@psu.edu, madduri@cse.psu.edu, srajama@sandia.gov

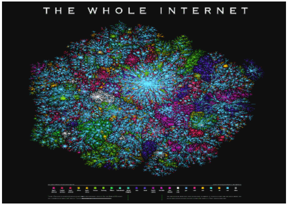
Blue Waters Symposium 12 May 2015

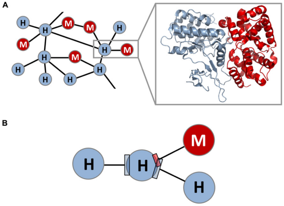
Graphs are...

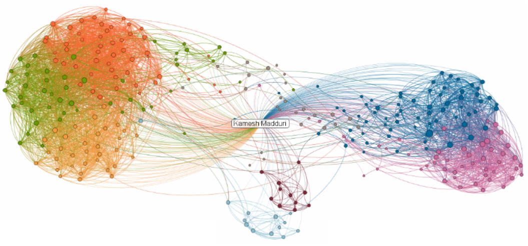
- **Everywhere**

Graphs are...

- **Everywhere**
  - Internet
  - Social networks, communication
  - Biology, chemistry
  - Scientific modeling, meshes, interactions







- Figure sources: Franzosa et al. 2012, `http://www.unc.edu/ unclng/Internet_History.htm`

Graphs are...

- **Big**

Graphs are...

- **Big**
  - Internet - 50B+ pages indexed by Google, trillions of hyperlinks
  - Facebook - 800M users, 100B friendships
  - Human brain - 100B neurons, 1,000T synaptic connections

- Figure sources: Facebook, Science Photo Library - PASIEKA via Getty Images

Graphs are...

- **Complex**

Graphs are...

- **Complex**
  - Graph analytics is listed as one of DARPA's 23 toughest mathematical challenges
  - Extremely variable - $O(2^{n^2})$ possible simple graph structures for n vertices
  - Real-world graph characteristics makes computational analytics tough

Graphs are...

- **Complex**
  - Graph analytics is listed as one of DARPA's 23 toughest mathematical challenges
  - Extremely variable - $O(2^{n^2})$ possible simple graph structures for n vertices
  - Real-world graph characteristics makes computational analytics tough
    - Skewed degree distributions
    - Small-world nature
    - Dynamic

Scope of Fellowship Work

Key challenges and goals

- **Challenge**: Irregular and skewed graphs make parallelization difficult
    - **Goal**: Optimization for wide parallelization on current and future manycore processors

Scope of Fellowship Work

Key challenges and goals

- **Challenge**: Irregular and skewed graphs make parallelization difficult
  - **Goal**: Optimization for wide parallelization on current and future manycore processors
- **Challenge**: Storing large graphs in distributed memory
  - Layout - partitioning & ordering, what objectives and constraints should be used?
  - **Goal**: Improve execution time (computation & communication) for simple and complex analytics

Scope of Fellowship Work

Key challenges and goals

- **Challenge**: Irregular and skewed graphs make parallelization difficult
  - **Goal**: Optimization for wide parallelization on current and future manycore processors
- **Challenge**: Storing large graphs in distributed memory
  - Layout - partitioning & ordering, what objectives and constraints should be used?
  - **Goal**: Improve execution time (computation & communication) for simple and complex analytics
- **Challenge**: End-to-end execution of analytics on supercomputers
  - End-to-end - read in graph data, create distributed representation, perform analytic, output results
  - **Goal**: Using lessons learned to minimize end-to-end execution times and allow scalability to massive graphs

Optimizing for Wide Parallelism

GPUs on Blue Waters and Xeon Phis on other systems

- **Observation**: most graph algorithms follow a tri-nested loop structure
  - Optimize for this general algorithmic structure
  - Transform structure for more parallelism

Initialize temp/result arrays $A_t[1..n]$, $1 \le t \le l$. $\triangleright\ l = O(1)$
Initialize $S_1[1..n]$.
for $i = 1$ **to** $niter$ **do** $\triangleright\ niter = O(\log n)$
    Initialize $S_{i+1}[1..n]$. $\triangleright\ \sum_i |S_i| = O(m)$
    for $j = 1$ **to** $|S_i|$ **do** $\triangleright\ |S_i| = O(n)$
        $u \leftarrow S_i[j]$
        Read/update $A_t[u]$, $1 \le t \le l$.
        for $k = 1$ **to** $|E[u]|$ **do** $\triangleright\ |E[u]| = O(n)$
            $v \leftarrow E[u][k]$
            Read/update $A_t[v]$.
            Read/update S_{i+1}.
        Read/update $A_t[u]$.

Optimizing for Wide Parallelization

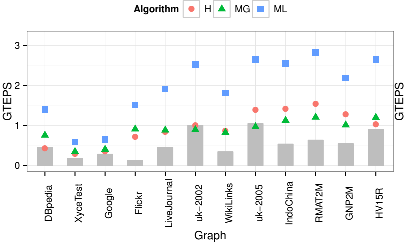
Approaches for improving intra-node parallelism

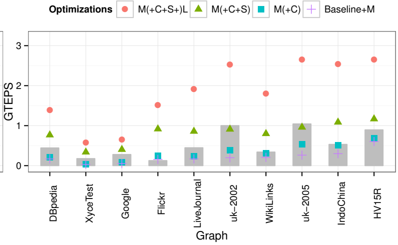
- **Hierachical expansion**
  - Depending on degree of a vertex, parallelism handled per-thread, per-warp, or per-multiprocessor
- **Local Manhattan Collapse**
  - Inner two loops (across vertices and adjacent edges in queue) collapsed into multiple single loop per-multiprocessor
- **Global Manhattan Collapse**
  - Inner two loops collapsed globally among all warps and multiprocessors
- **General optimizations**
  - Optimizations applicable to all parallel approaches - cache consideration, coalescing memory access, explicit shared memory usage, warp and MP-based primitives

Optimizing for Wide Parallelization

Performance results - K20 GPUs on Blue Waters

- H: Hierarchical, ML: Local collapse, MG: Global collapse, gray bar: Baseline
- M: local collapse, C: coalescing memory access, S: shared memory use, L: local team-based primitives
- Up to 3.25× performance improvement relative to optimized CPU code!





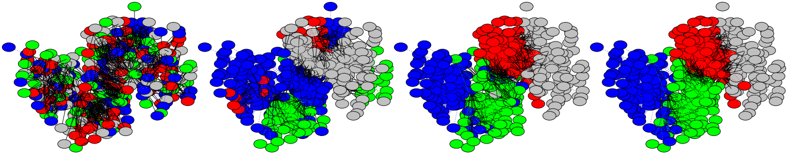
Distributed-memory layout for graphs
Partitioning and ordering

- Partitioning - how to distribute vertices and edges among MPI tasks
  - Objectives - minimize both edges between tasks (cut) and maximal number of edges coming out of any given task (max cut)
  - Constraints - balance vertices per part and edges per part
  - **Want balanced partitions with low cut to minimize communication, computation, and idle time among parts!**
- Ordering - how to order intra-part vertices and edges in memory
  - Ordering affects execution time by optimizing for memory access locality and cache utilization
- Both are very difficult with small-world graphs

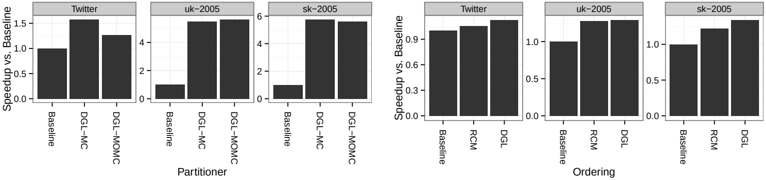
Distributed-memory layout for graphs

Partitioning and ordering part 2

- Partitioning
  - Used PuLP partitioner for generating multi-constraint multi-objective partitions
  - Only partitioner available that's both scalable to graphs tested on and able to satisfy objectives/constraints
- Ordering
  - Used traditional bandwidth reduction methods from numerical analysis
  - Also used more graph-centric methods based around breadth-first search



- Speedups for subgraph counting algorithm for communication and computation
- Effective partitioning can make considerable impact, ordering still important as graphs get large



Large-scale graph analytics

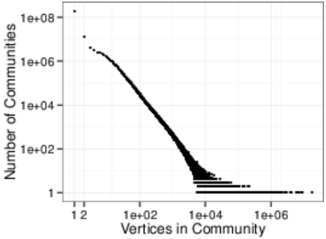
- Previous work for large graph analysis
  - External-memory systems - MapReduce/Hadoop-like, flash memory
  - Tend to be slow and energy intensive
- Using optimizations and techniques from fellowship work efforts
  - Implemented analytic suite for large-scale analytics (connectivity, k-core, community detection, PageRank, centrality measures)
  - Ran on largest currently available public web crawl (3.5B vertices, 129B edges)
  - First known work that has successfully analyzed graph of that scale on a distributed memory system

Large-scale graph analytics

- Ran algorithm suite on only 256 nodes of Blue Waters, execution time in minutes
- Novel insights gathered from analysis - largest communities discovered, communities appear to have scale-free or heavy-tailed distribution

Largest Communities Discovered (numbers in millions)

Pages	Internal Links	External Links	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



Summary of accomplishments

- Optimizations for manycore parallelism result in up to a **3.25× performance improvement** for graph analytics executing on GPU
- Modifications to in-memory storage of graph structure results in up to a **1.48× performance improvement** for distributed analytics running with MPI+OpenMP on Blue Waters
- **First-ever analysis** of largest to-date web crawl (129B hyperlinks) on a distributed memory system
- Running on 256 nodes of Blue Waters, we are able to run several complex graph analytics on the web crawl **in minutes of execution time**

Summary of accomplishments - publications

- **High-performance Graph Analytics on Manycore Processors**
  - To appear in the Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium (IPDPS15)
- **Distributed Graph Layout for Scalable Small-world Network Analysis**
  - In submission
- **Supercomputing for Web Graph Analytics**
  - In submission
- **Poster at IPDPS15**
- **Poster at SC15** (tentative)

Conclusions and Going Forward

- Real-world graphs = big, complex, difficult to effectively run on in parallel
- Demonstrated methodology for thread-node-system level optimization for small-world skewed graphs
- Hopefully this work will enable:
  - Implementation of more complex analytics for large networks
  - Scaling to larger networks and on larger future systems
  - Greater insight into larger networks than currently possible
- Thanks to Blue Waters and NCSA!

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. This work is also supported by NSF grants ACI-1253881, 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 Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.