Irregular Graph Algorithms on Parallel Processing Systems

George M. Slota[1,2] Kamesh Madduri[1] (advisor)
Sivasankaran Rajamanickam[2] (Sandia mentor)

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

SC15 PhD Forum 19 Nov 2015

Summary of accomplishments

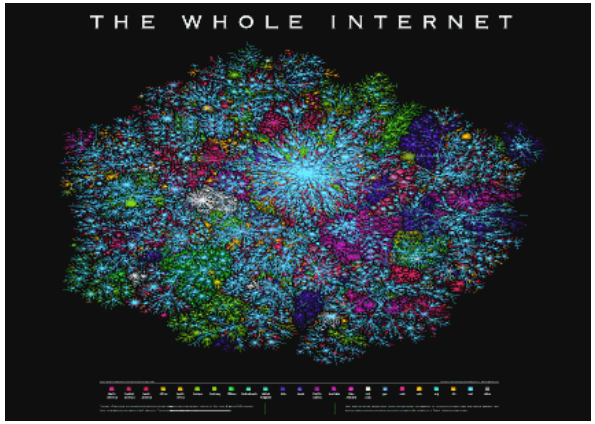
- **Multistep** graph connectivity algorithms; on average $2\times$ faster than prior state-of-the-art.
- **Manycore optimization** methodology for graph algorithm implementation on GPU
- **FASCIA** and **FastPath** subgraph counting and min-weight path finding programs; up to orders-of-magnitude execution time improvement over prior art.
- **PuLP** partitioner; order of magnitude faster, order of magnitude less memory, comparable or better partition quality than other state-of-the-art utilities. Scales to 100B+ edge graphs, which it can partition in minutes.
- **Distributed Graph Layout** methodology for distributed memory graph storage; up to $12\times$ performance improvements over naive methods
- **Complex Analysis** of largest publicly available web crawl using techniques derived from above work (3.5B vertices and 129B edges), analytic suite completes in 20 minutes on 256 nodes.

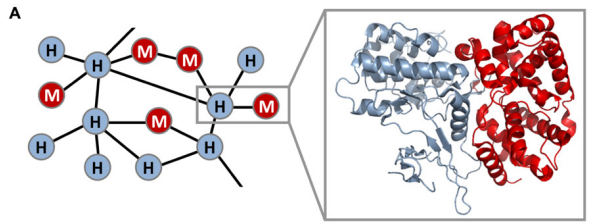
Graphs are...

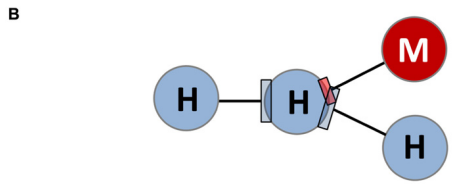
- **Everywhere**

Graphs are...

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







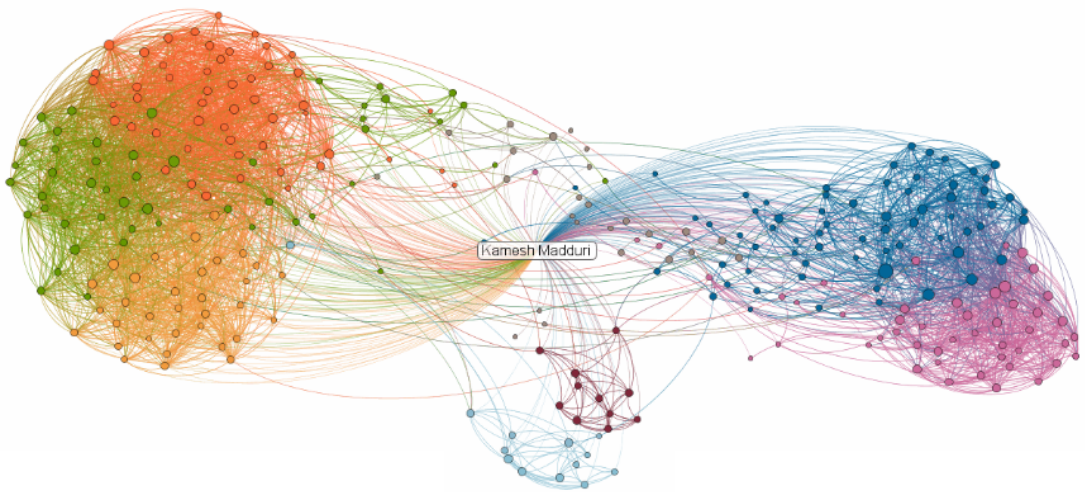


Figure sources: Franzosa et al. 2012, `http://www.unc.edu/ unclng/Internet_History.htm`

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
    - Dynamically changing

Scope of Thesis Research
Key challenges and goals

- **Challenge**: Irregular and small-world graphs make parallelization difficult
  - **Goal**: Optimize graph algorithm design at shared-memory level
  - **Goal**: Investigated distributed memory graph layout (partitioning-ordering) and optimization
  - **Goal**: Introduce methodology for core-to-system level parallelization of large-scale analytics

Scope of Thesis Research

Key challenges and goals

- **Challenge**: Irregular and small-world graphs make parallelization difficult
  - **Goal**: Optimize graph algorithm design at shared-memory level
  - **Goal**: Investigated distributed memory graph layout (partitioning-ordering) and optimization
  - **Goal**: Introduce methodology for core-to-system level parallelization of large-scale analytics
- **Challenge**: Perform end-to-end execution of graph 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

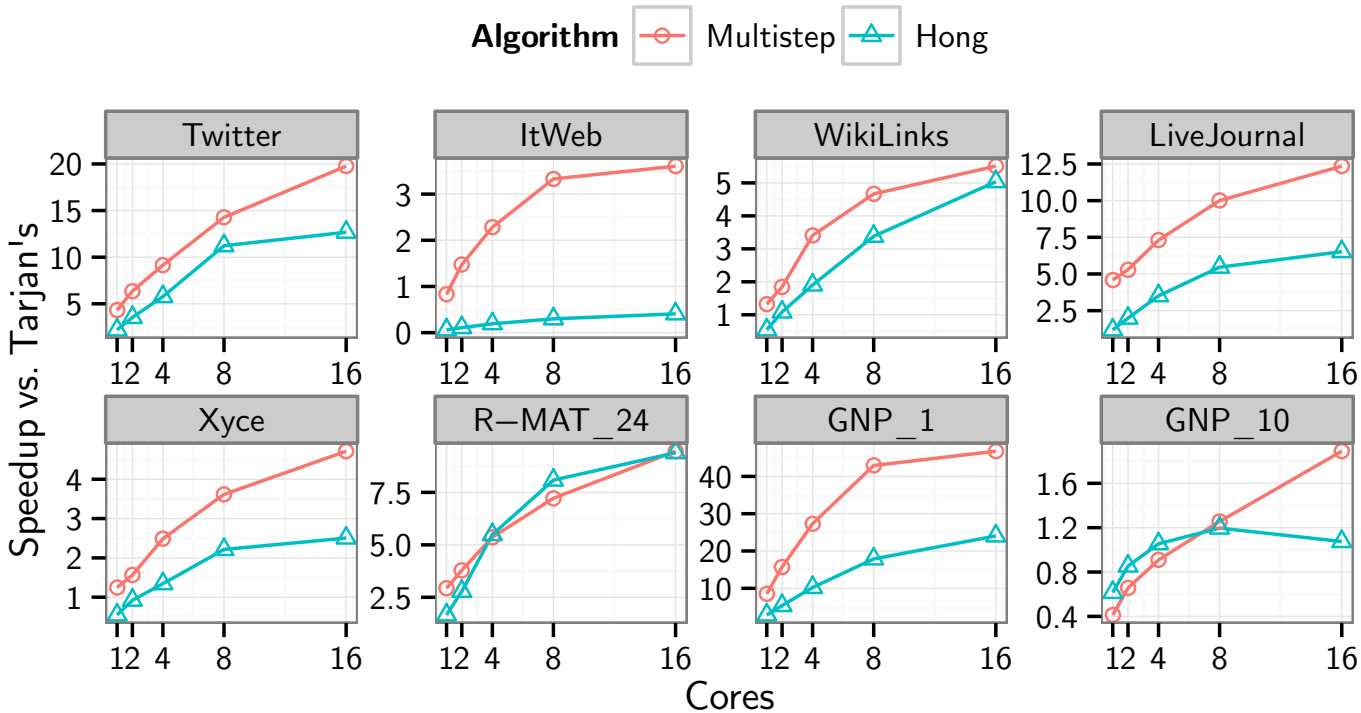
Shared-Memory Algorithm Optimization
Multistep for Graph Connectivity

- **Multistep** approach to graph connectivity, weak connectivity, strong connectivity and biconnectivity (Slota et al. 2014, Slota and Madduri, 2014)
- **Key Optimizations**:
  - Emphasis on data locality: multiple queue levels
  - Minimize synchronization costs: few atomics and no criticals
  - Subroutines optimization: direction-optimizing BFS, push- and queue-based color propagation

Shared-Memory Algorithm Optimization

Multistep for Graph Connectivity: Performance

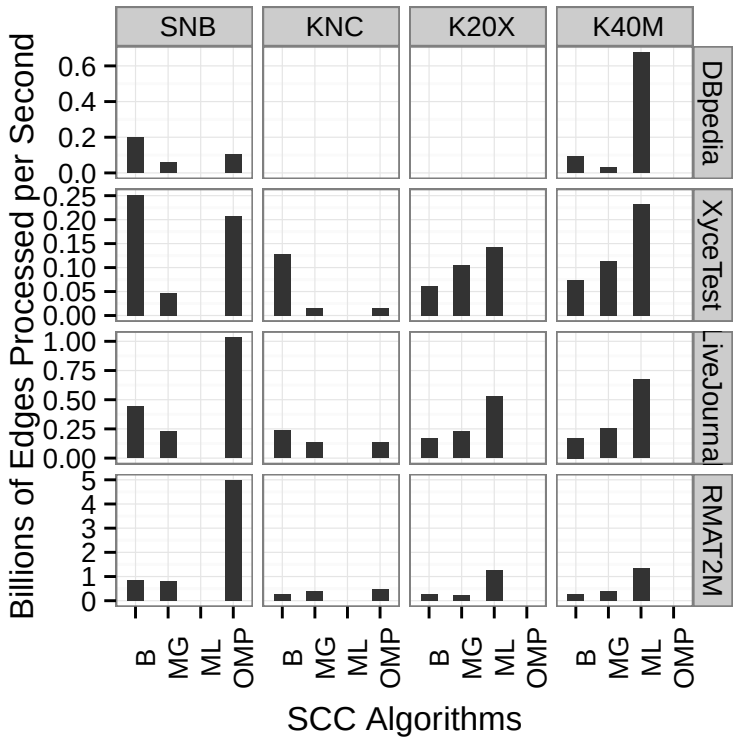
- On average $2\times$ faster than prior state-of-the-art for SCC, up to $7\times$ faster for biconnectivity
- Compare SCC times to state-of-the-art (Hong et al. 2013) in terms of speedup versus optimal serial algorithm



Shared-Memory Algorithm Optimization

Multistep for Graph Connectivity: Manycore Optimization

- Optimized SCC code for high performance on manycore processors such as GPUs (Slota et al. 2015)
- Up to $3.25\times$ performance improvement over CPU
- Methodology generalizable to broad class of graph algorithms
- **Key optimizations**:
  - Loop manipulation for higher parallelism (Manhattan Collapse)
  - Shared-memory and other locality considerations
  - Warp and team-based atomics and operations (team scan, team reduce, etc.)



Cross-platform comparison of optimized manycore code. Performance given in billions of edges processed per second.

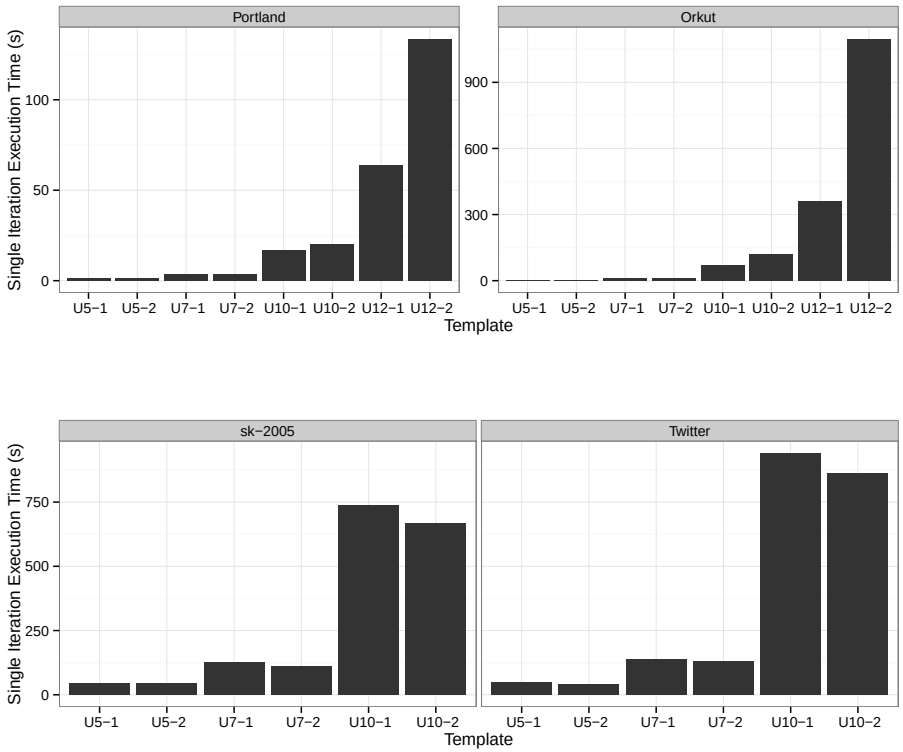
Shared-Memory Algorithm Optimization
FASCIA for Subgraph Counting and Enumeration

- **FASCIA**: implementation of color-coding subgraph counting algorithm (Alon et al. 1995)
- Subgraph counting in shared and distributed memory, and minimum weight path finding on weighted networks (**FastPath**)
- **Key optimizations**:
  - Abstraction of key dynamic programming phase
  - Design storage structures for optimal cache utilization
  - Use of communication and computation avoidance techniques
  - Multiple memory reduction strategies

Shared-Memory Algorithm Optimization

FASCIA for Subgraph Counting and Enumeration: Performance

- Can count nontrivial subgraphs in minutes on multi-million edge graphs in shared memory (top) and multi-billion edge graph in distributed memory (bottom)
- *Orders-of-magnitude improvement* over prior art (Slota and Madduri 2013, 2014, 2015)



Distributed-memory layout for graphs

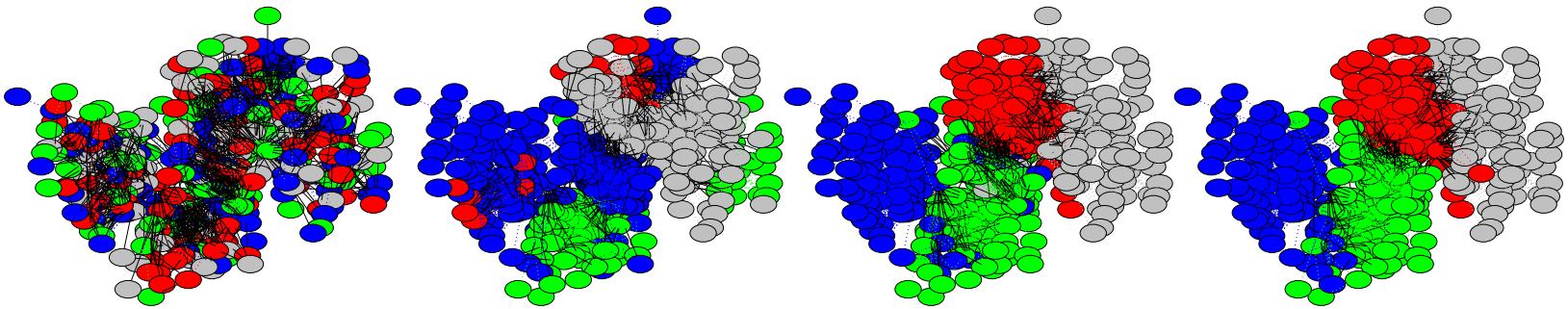
Partitioning and ordering

- 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 with PuLP

- PuLP partitioner: generating multi-constraint multi-objective partitions
- Designed specifically to partition small-world and irregular graphs
- Only partitioner available that's scalable to multi-billion edge graphs and able to satisfy multiple constraints and objectives, all without sacrificing cut quality

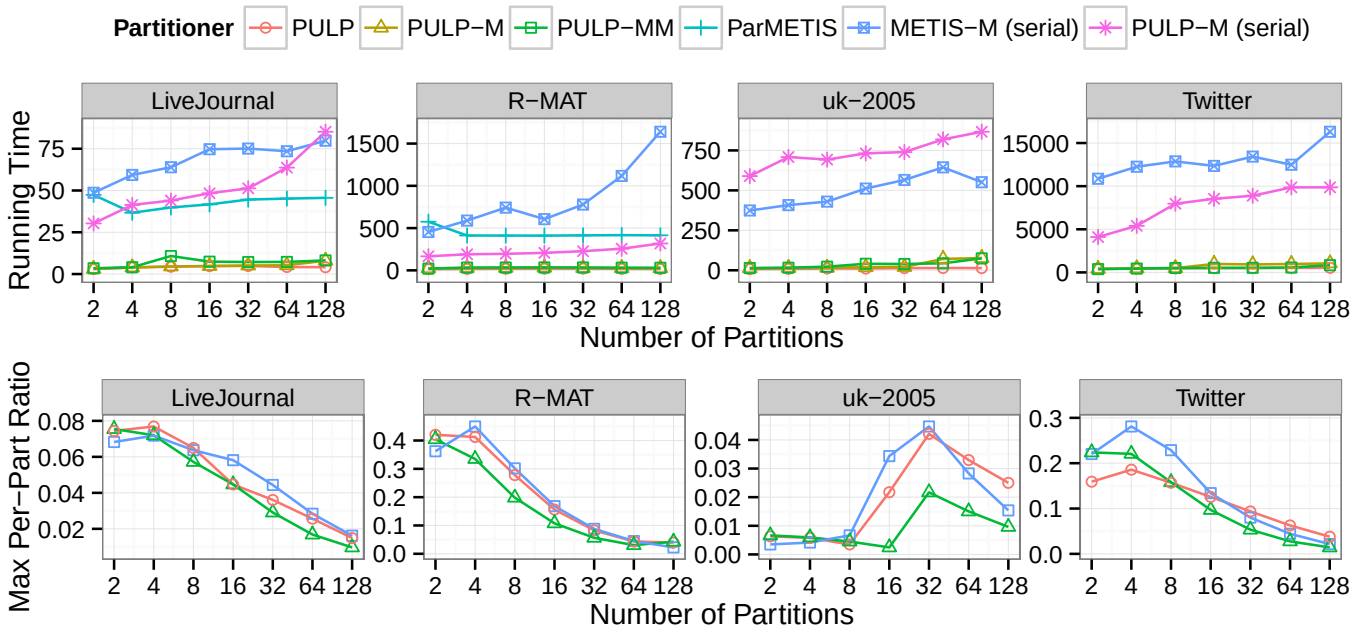


PuLP Algorithm: initialize, balance vertices, refine, balance edges, refine, balance and minimize cut

Distributed-memory layout for graphs

Partitioning with PuLP: Performance

- PuLP demonstrates 14.5× average speedup across a suite of test graphs compared to METIS and ParMETIS
- PuLP uses up to 38× less memory than METIS or KaFFPa
- PuLP's generated partitions are equivalent or better in quality

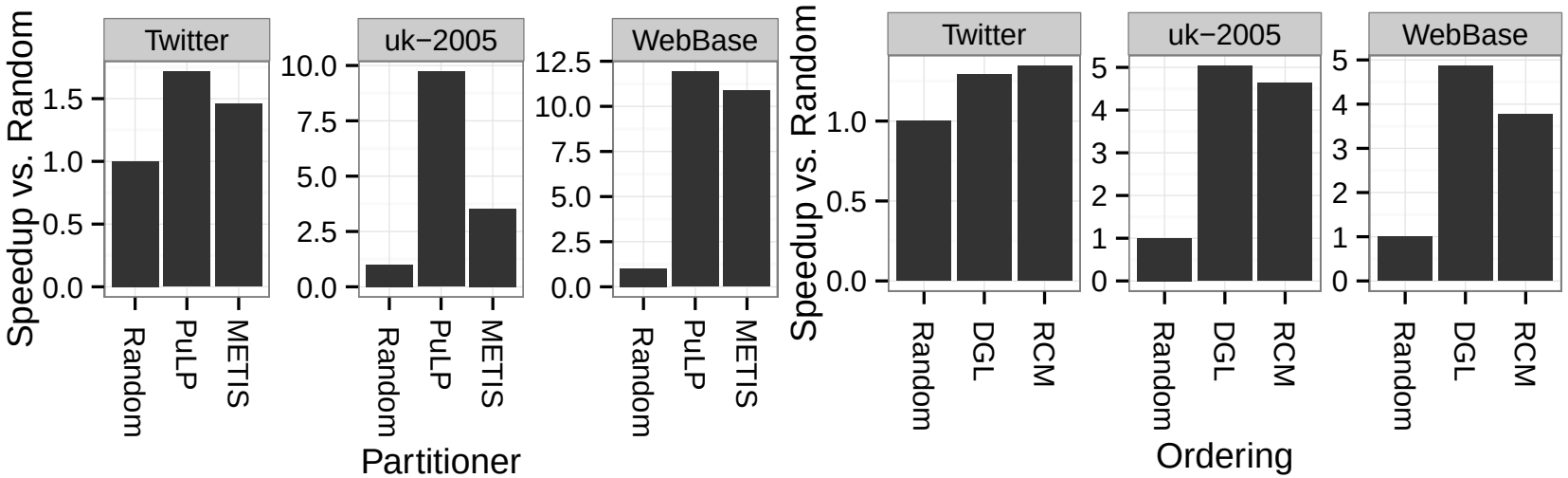


Execution times for PuLP relative to METIS and ParMETIS (top) and partition quality in terms of edge cut for PuLP and ParMETIS (bottom).

Distributed-memory layout for graphs

Performance results

- DGL: Distributed graph layout (partitioning+ordering)
- Developed new fast ordering method and analyzed graph analytic performance impacts of PuLP partitioning and our ordering strategy
- PuLP partitioning can have up to a **12× speedup** on communication times; our vertex ordering can have up to **5× speedup** on computation times against naive methods



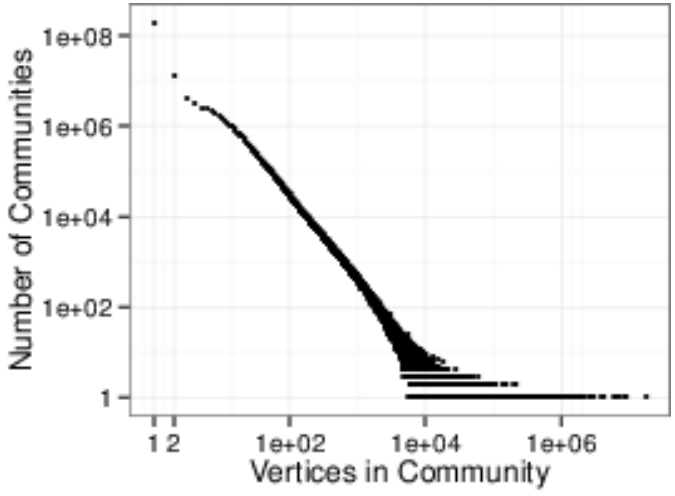
Impact of partitioning quality (left) and intra-task vertex orderings (right) on execution times of graph analytics.

Large-scale Graph Analytics

- Using optimizations and techniques from thesis research efforts, implemented analytic suite for large-scale analytics (connectivity, k-core, community detection, PageRank, centrality measures)
- Ran algorithm suite on only 256 nodes of Blue Waters system, full end-to-end execution time in 20 minutes
- Novel insights gathered from analysis - largest communities discovered, communities appear to have scale-free or heavy-tailed distribution

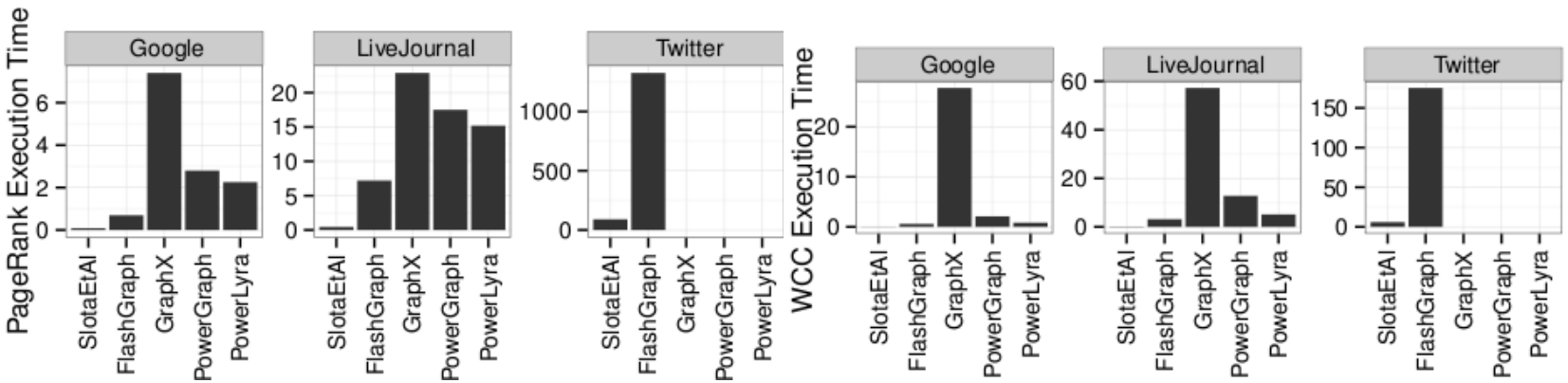
Largest Communities Discovered (numbers in millions)

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



Large-scale Graph Analytics
Comparison to other approaches

- We compare our implementation approach to several popular distributed/shared memory frameworks (GraphX, PowerGraph, PowerLyra, FlashGraph)
- Across suite of test graphs and frameworks, our PageRank implementation (left) is 37× faster on average and our WCC implementation (right) is 97× faster on average while running on 16 nodes



Comparison of graph analysis frameworks for PageRank (left) and WCC (right).

Conclusions and Going Forward

- Real-world graphs = big, complex, difficult to effectively run on in parallel
- Demonstrated various optimization approaches for shared and distributed memory
- 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 NSF, Penn State (Kamesh Madduri), Sandia Labs (Siva Rajamanickam), 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.