Characterizing Parallel Graph Analysis Algorithms on Multicore Systems

Pervasive Parallelism Laboratory, Stanford University

Nicole Rodia and Kunle Olukotun

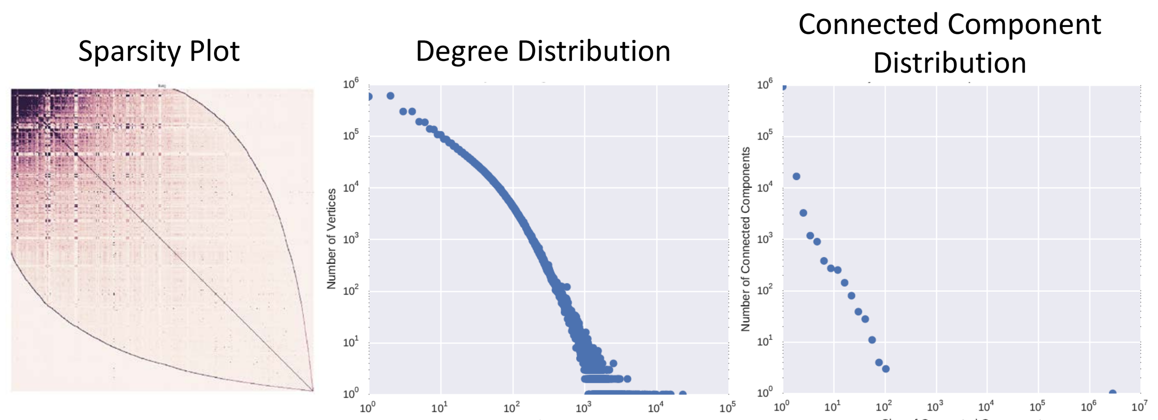
Graph Analytics

- Graph datasets contain millions to billions of nodes (n) and edges (m)
- Algorithms are:
  - Expensive, e.g. $O(n+m)$
  - Memory intensive
    - Low locality (random access)
    - Large data size footprint
    - Little computation to hide memory latency
  - Difficult to partition

Graph Properties

- Scale-free: power-law degree distribution
- Small-world: $O(\log n)$ diameter

Example: LiveJournal social network



Methodology

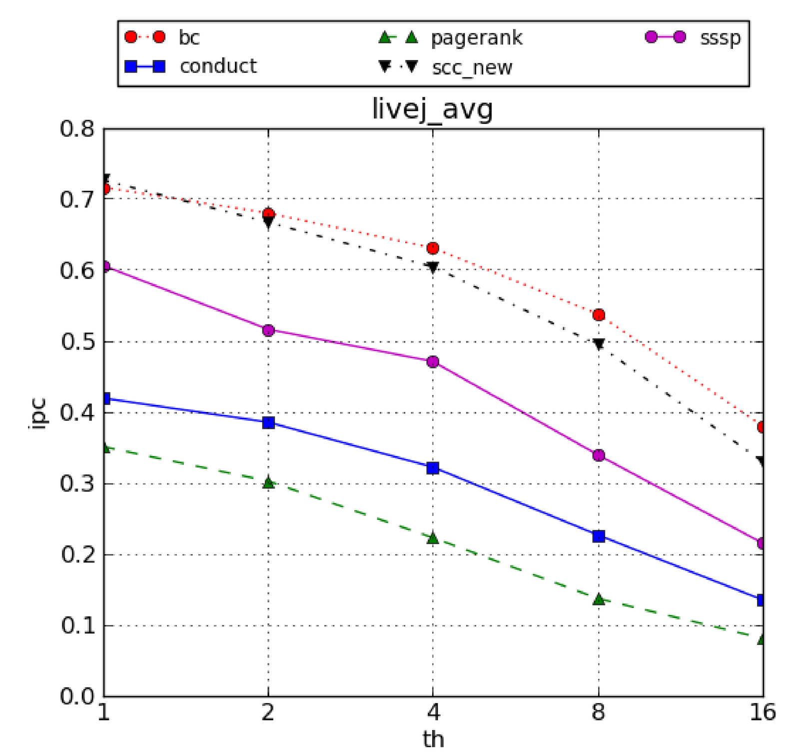
Applications and Datasets

- Social & information network analysis applications
  - Implemented in Green-Marl DSL [1]
- Datasets from social, web link, road, FE mesh, and synthetic graphs

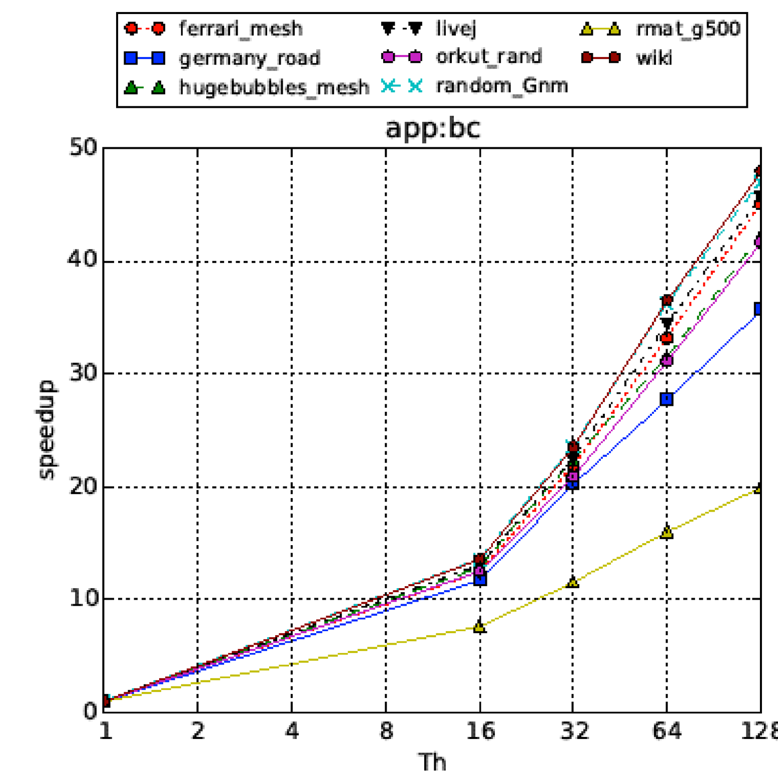
Analysis Environments

- Execution-driven multiprocessor simulation with multi-level memory subsystem (ZSim) [2]
- Performance counter measurements on Intel x86 Nehalem-based machine

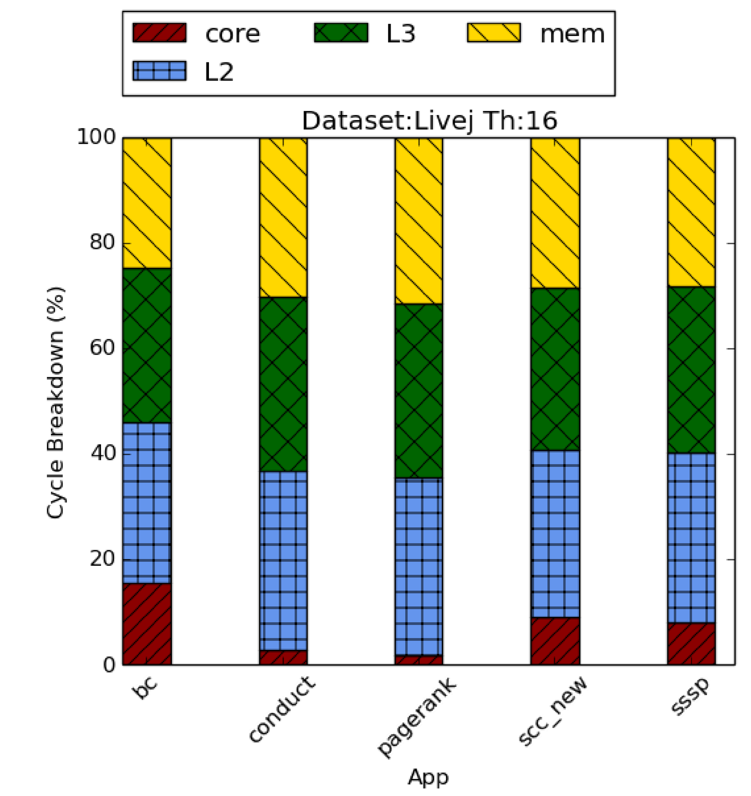
Scaling Behavior and Performance Bottlenecks



Instructions per cycle (IPC) for LiveJournal dataset on 16-core simulated system

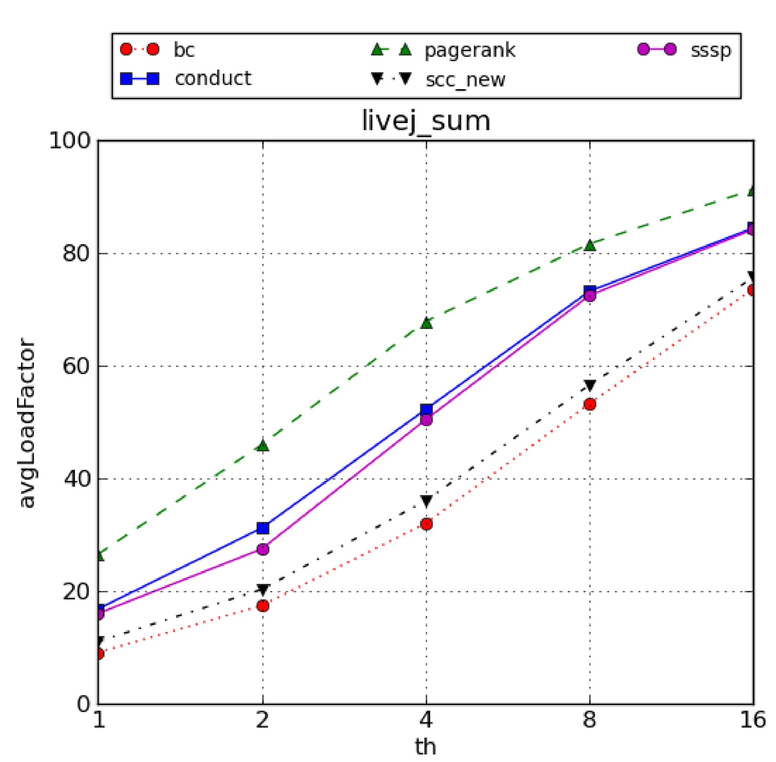


Parallel speedup for 128-core simulated system for Betweenness Centrality algorithm

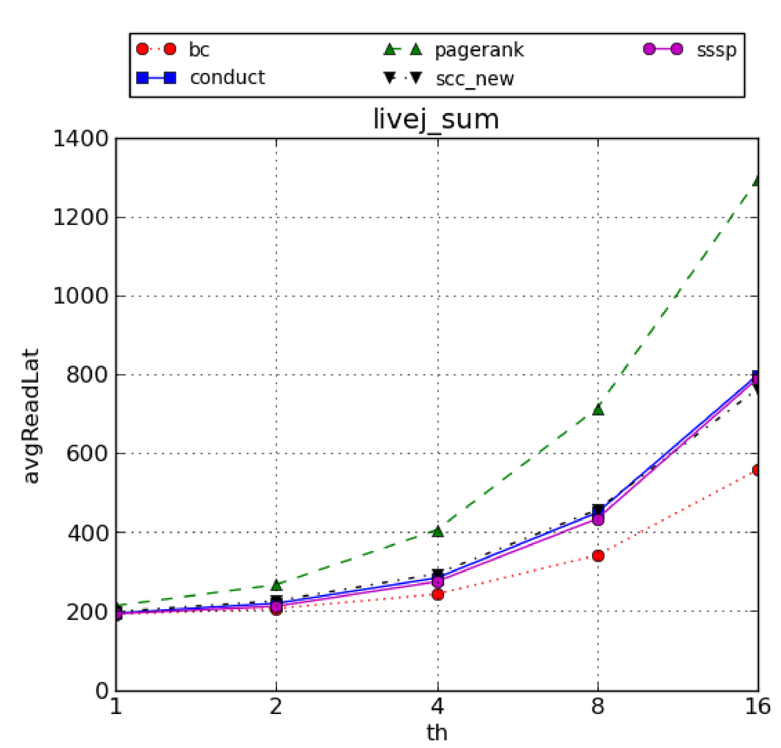


Execution breakdown for 16-core simulated system for LiveJournal dataset

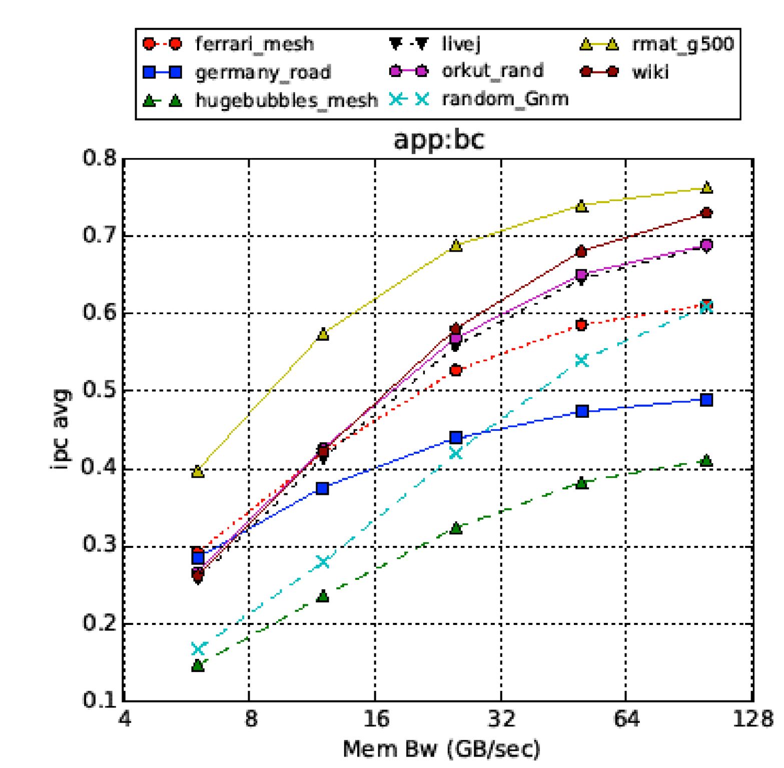
Memory Bandwidth and Latency



Simulated main memory average load factor as a function of thread count for 16-core system

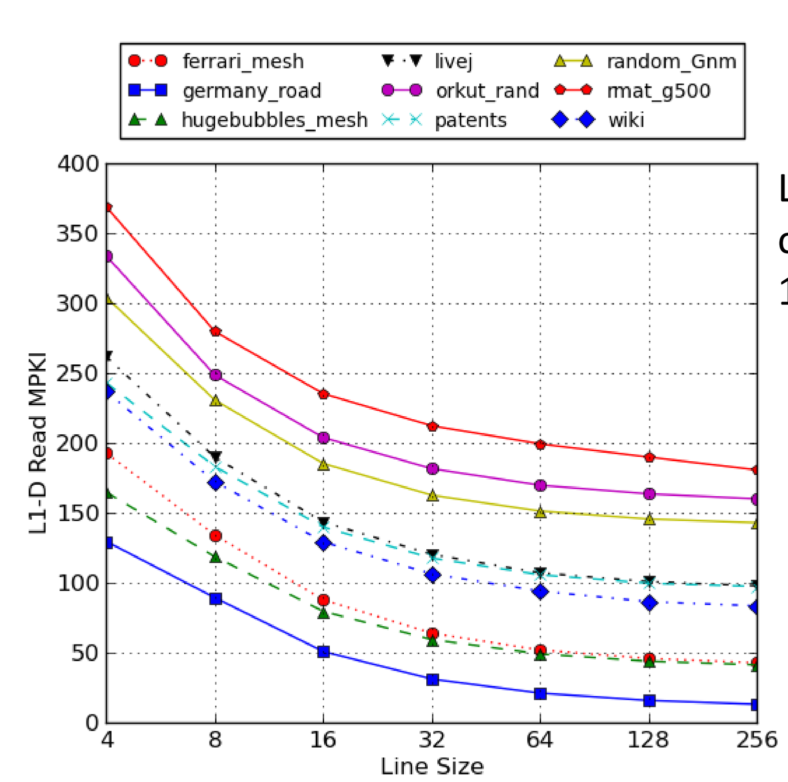


Simulated main memory average read latency as a function of thread count for 16-core system



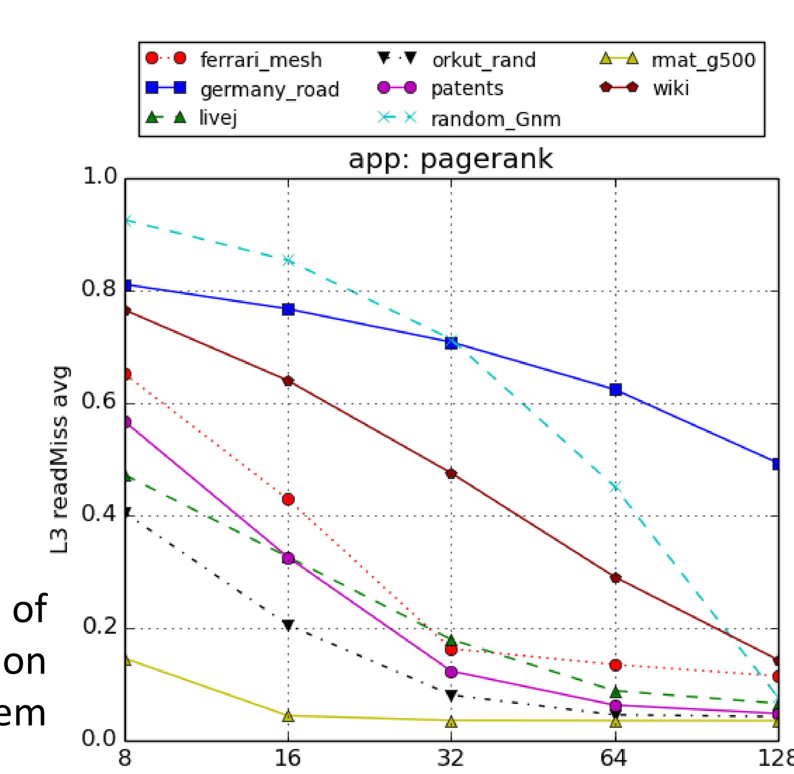
IPC as a function of maximum main memory bandwidth on 16-core simulated system

Data Cache Analysis



L1 data cache read MPKI as a function of cache line size for PageRank algorithm on 16-core simulated system

L3 cache read miss rate as a function of L3 cache size for PageRank algorithm on 16-core simulated system



Conclusions

- Significant available data parallelism
- Bottlenecks
  - Memory bandwidth
  - Sequential code sections
  - Parallel load imbalance
  - Graph size and structure
- Data cache analysis
  - Compulsory and capacity are main sources of cache misses

Future Work

- Implement performance improvements in simulator
  - Data structure-specific selective caching
  - Fine-grained memory access
  - Application-specific prefetching
  - Graph analytics-specific hardware co-located with memory

    Ex. pointer dereference, reduction, BFS, DFS
- Investigate algorithmic improvements
  - Refactor to improve locality and parallel scaling
  - Mitigate parallel load imbalance

References

[1] S. Hong, H. Chafi, E. Sedlar, and K. Olukotun. "Green-Marl: A DSL for Easy and Efficient Graph Analysis," in ASPLOS '12.

[2] D. Sanchez and C. Kozyrakis. "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in ISCA '13.

Contact Information

E-mail: nrodia@stanford.edu