

A Simulation-Based Study of Graph Algorithm Characteristics

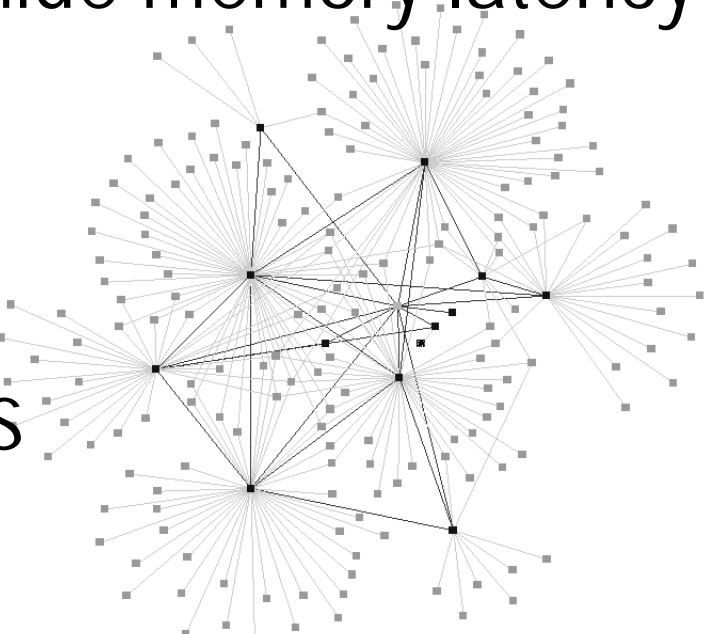
Pervasive Parallelism Laboratory, Stanford University
Nicole Rodia and Kunle Olukotun



Graph Algorithms

Problem: Graph Processing

- Memory intensive
 - Low computation-to-communication ratio
 - Limited spatial and temporal locality
 - Large memory footprint (data size)
 - Little computation to hide memory latency
- Data-level parallelism
- Expensive, e.g. $O(n)$
 - n = number of vertices
 - m = number of edges



Methodology

Workloads

- Social & information network analysis algos

Analysis Environments

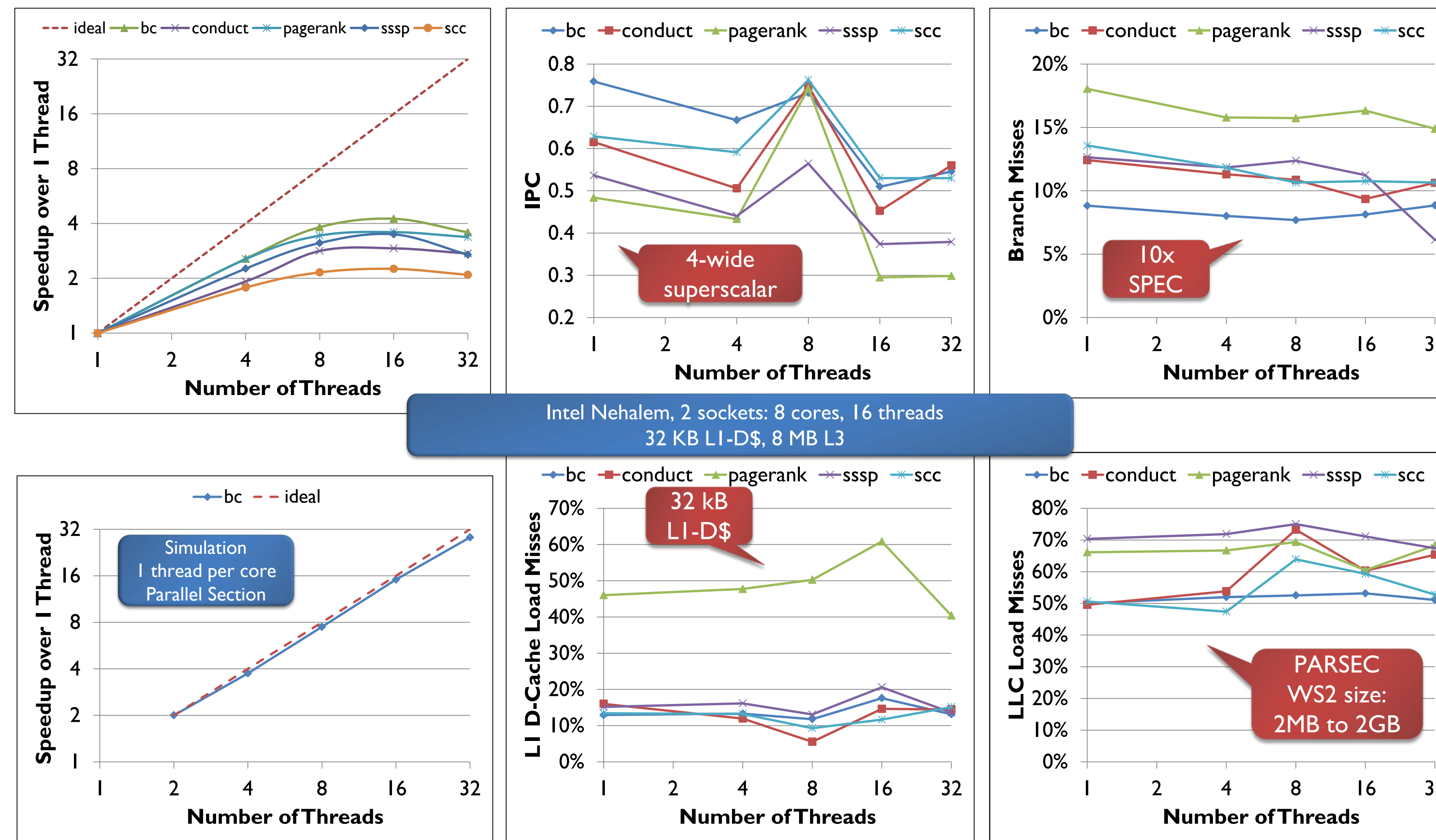
- Execution-driven multiprocessor simulation with multi-level memory subsystem (zsim) [3]
 - Nominal Simulator Configuration:
 - 32 cores, 1 thread per core, IPC = 1
 - Private L1 cache per core
 - Shared L2 cache, inclusive
 - Directory-based MESI coherence
- Performance counter measurements on x86 Intel Nehalem-based machine
 - 2.66 GHz, 2 sockets: 8 cores, 16 threads
 - 32 KB L1 D\$ and I\$ per core
 - 256 KB L2 cache per core, inclusive
 - 8 MB L3 cache per chip, inclusive
 - MESIF coherence

Contact Information

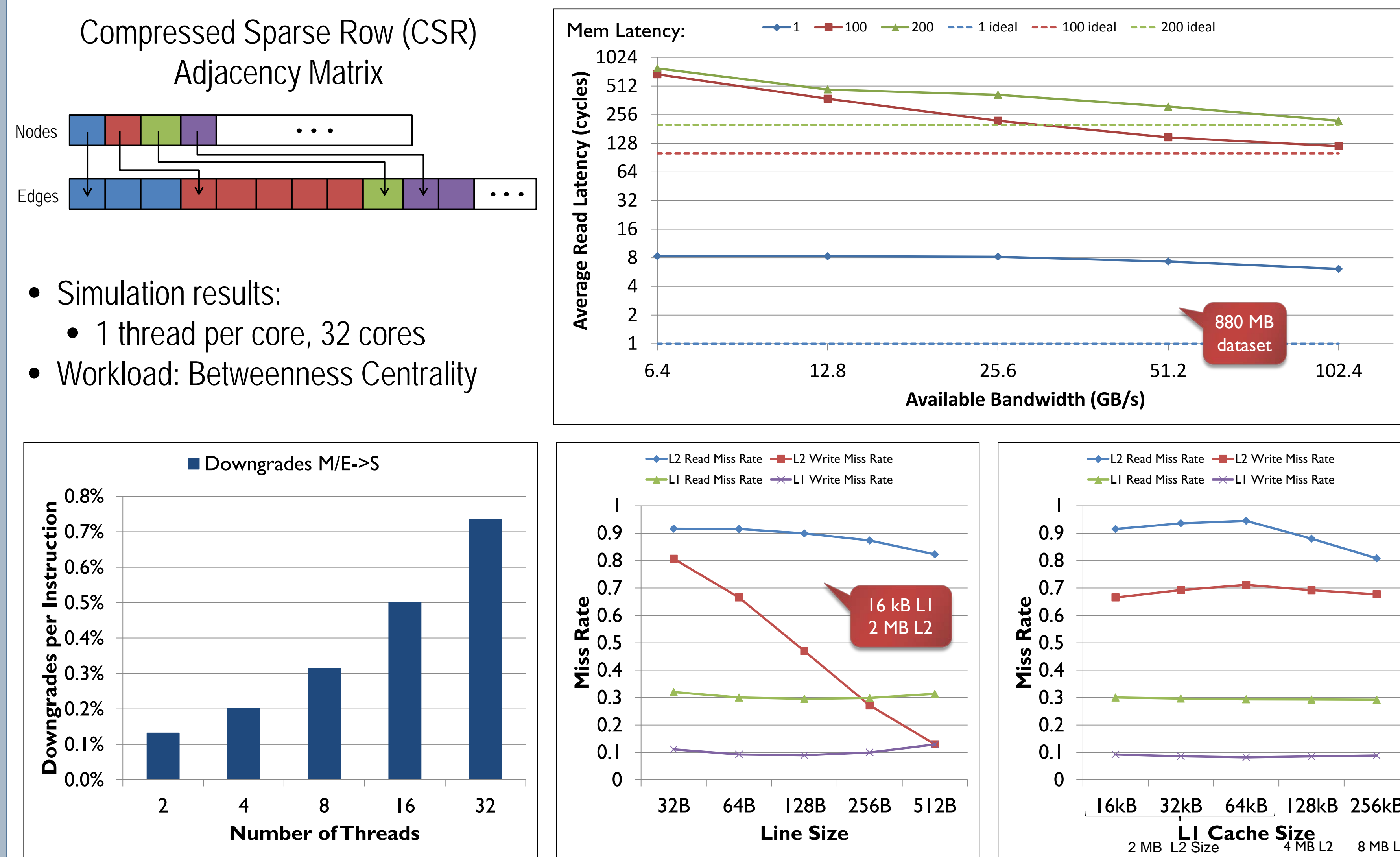
E-mail

• nrodia@stanford.edu

Scaling Behavior and Existing Multiprocessor Performance



Memory Bandwidth, Latency, and Access Patterns



Conclusions

- Scaling potential enables a highly parallel implementation
- L1 data cache useful for small working set
- Memory latency and ineffective caching are obstacles to high performance
- Trade off LLC area for more thread state
- Inter-thread communication increases with thread count
- If latency is decreased, can take advantage of more bandwidth

Future Work

- Investigate architectural improvements in simulation
 - Intelligent data prefetching and cache eviction/partitioning (leverage DSL)
 - Fast multithreading
 - Thread migration
 - Graph-specific instructions and hardware
 - e.g. BFS, reduction operations
 - Fine-grained memory accesses
- Investigate algorithmic improvements
 - Refactor for better branch prediction and locality
 - Improve parallel scaling

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] S. Hong, T. Oguntebi, and K. Olukotun. "Efficient parallel graph exploration on multi-core CPU and GPU," in PACT, '11.
- [3] D. Sanchez and C. Kozyrakis. "The ZCache: Decoupling Ways and Associativity," in MICRO-44, '10.