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
- 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

Memory Bandwidth and Latency
- 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

Data Cache Analysis

References

Future Work

Contact Information
E-mail: nrodia@stanford.edu