SAND2012-4633P

Graph Analysis with High-Performance Computing

Jonathan Berry
Scalable Algorithms Department
Sandia National Laboratories

Part of 'Big Data" minisymposium, ISC 2012 (Hamburg)
June 20, 2012

Outline

- Big data vs. scientific computing
- HPC graph processing challenges
- Graph Problems
- Performance considerations

Plimpton’s Assessment: Big Data vs. Scientific Sim.

- **Programming model**
  - SS: tune same code for years; performance is driver
  - BD: adapt code to data; agile, minimize programming effort
- **Computational load**
  - SS: compute bound, in-core
  - BD: memory, I/O bound, out-of-core
- **Data distribution**
  - SS: structured; partitioning/load balancing/locality are key
  - BD: unstructured
- **Data usage and ownership**
  - SS: large datasets are output, not input (simulation) – regenerate in HPC
  - BD: data are valuable, in situ, can’t be moved

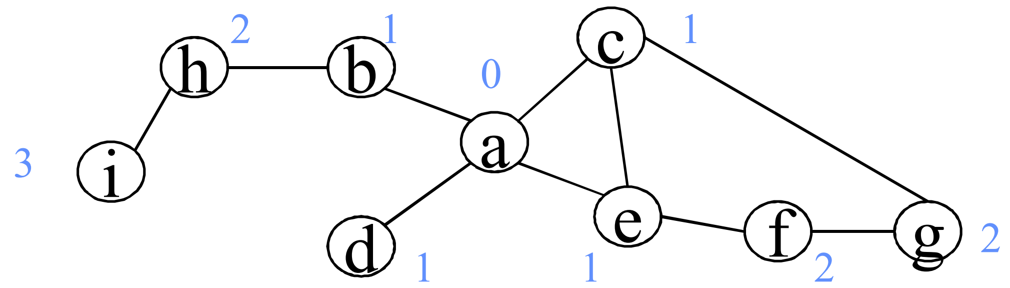
Thanks: Steve Plimpton, Sandia National Laboratories

Sandia National Laboratories

Graph Analysis: What is it?

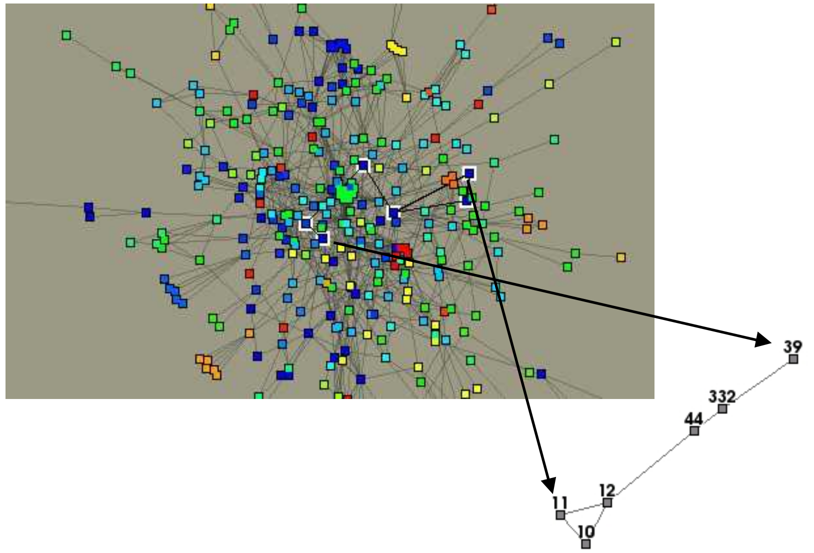
- **A *graph* is a set of vertices and a set of edges**
  - This abstraction is a powerful modeling tool
  - Edges are “meta-data” – storing this is economical and often sufficient
  - Motivations in homeland security, computer security, biology, etc.

- **Typical graph algorithm behavior:**
  - Nodes visit their neighbors
  - Algorithms traverse and compute
  - There’s a large amount of communication relative to computation
  - Often this is *asynchronous*

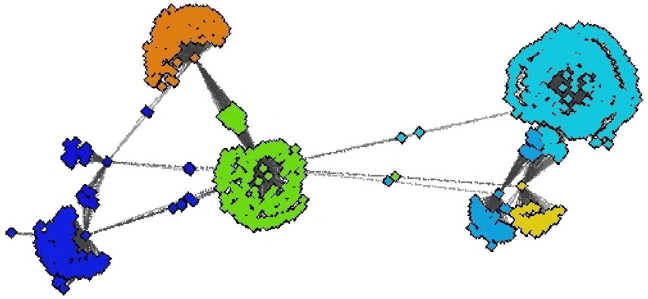


Some Non-Trivial Graph Analysis

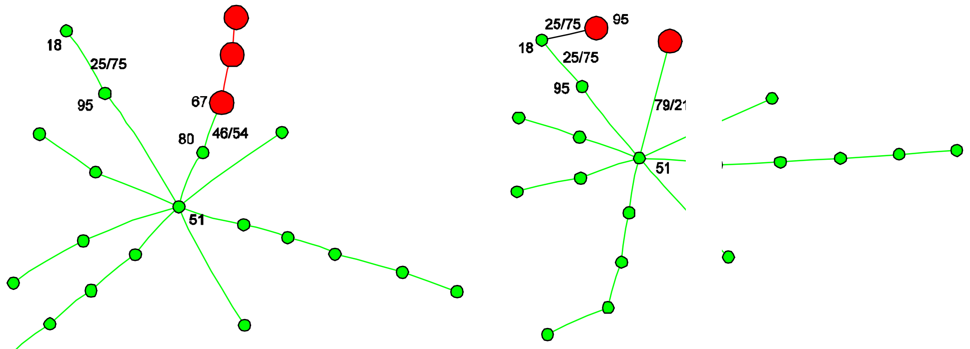
- Connection subgraphs (Faloutsos, et al., KDD 2004)



- Community detection (100's of papers)



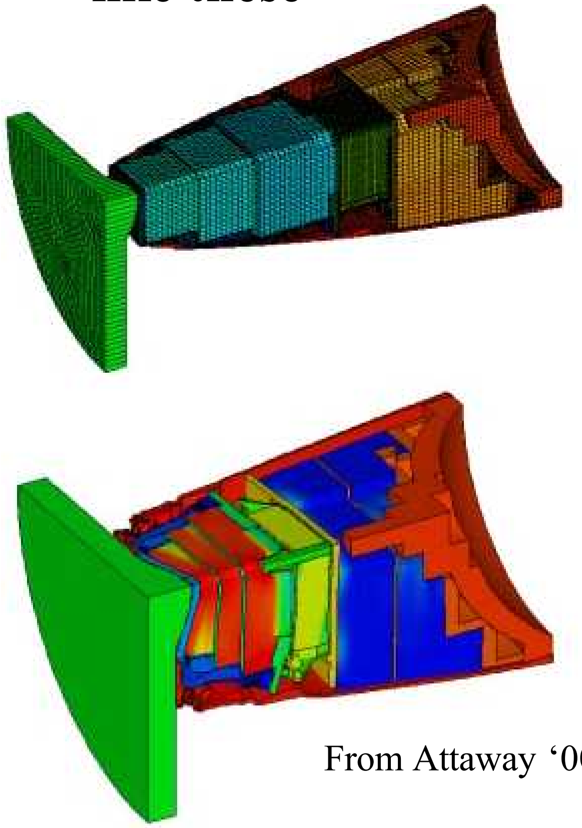
- Inexact subgraph isomorphism (several)



Application Characteristics Simplified

The HPC community is used to datasets with geometry,

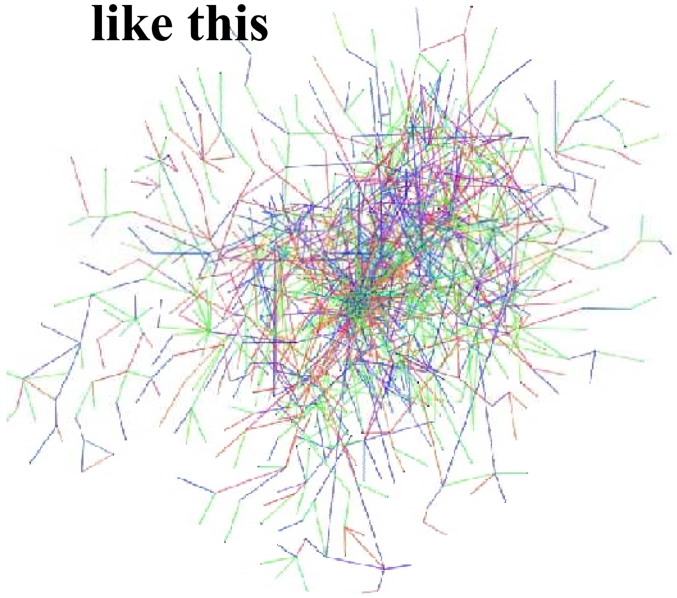
like these



From Attaway '00

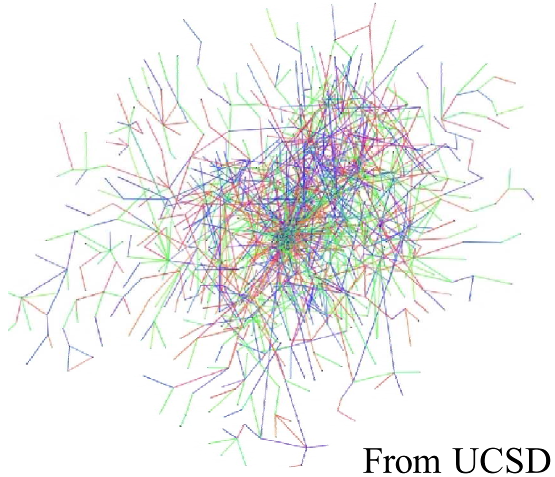
But datasets in graph analysis often look

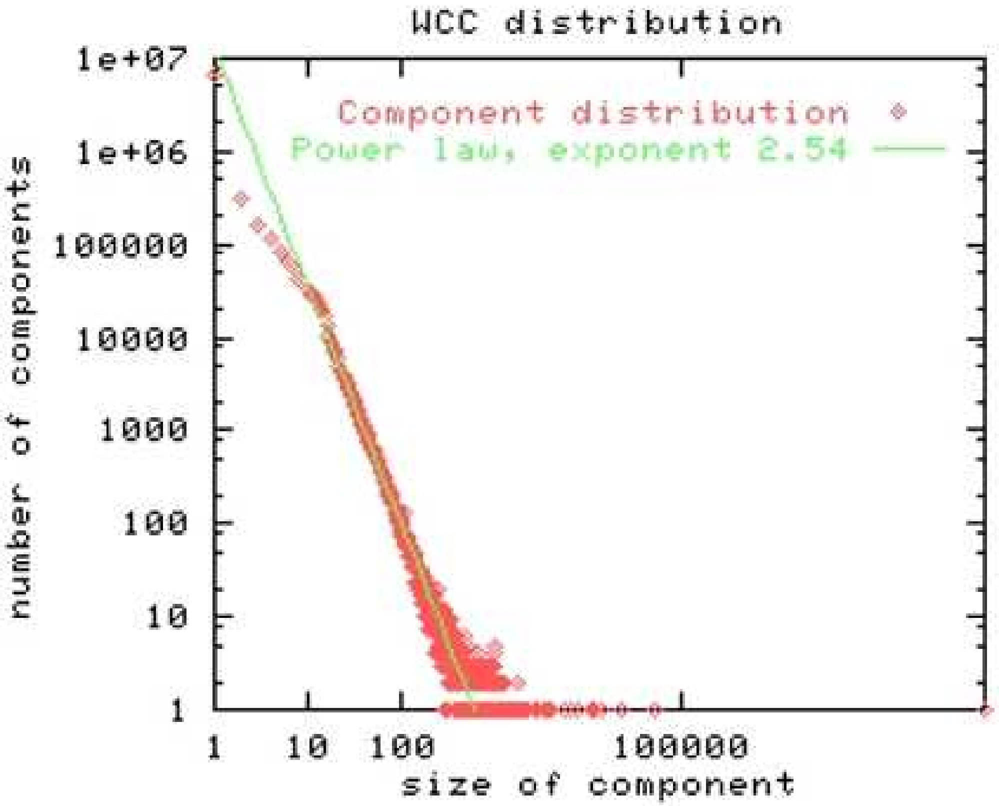
like this



From UCSD '08

Graph Datasets Are Different


From UCSD ‘08



“*One of the interesting ramifications of the fact that the PageRank calculation converges rapidly is that the web is an expander-like graph***”**

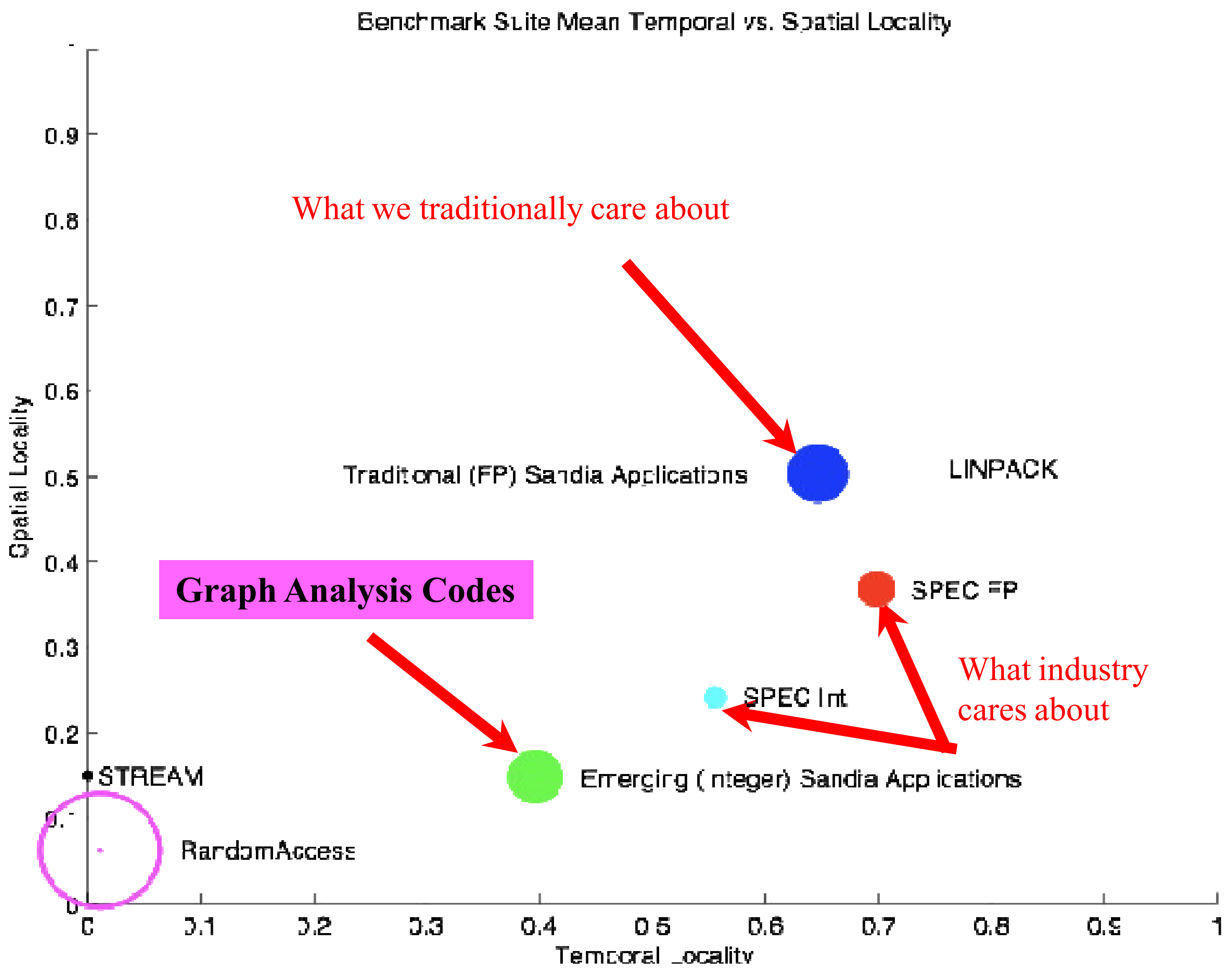
Page, Brin, Motwani,Winograd 1999

Broder, et al. ‘00

Primary HPC Implications: 1) Any partitioning is “bad”

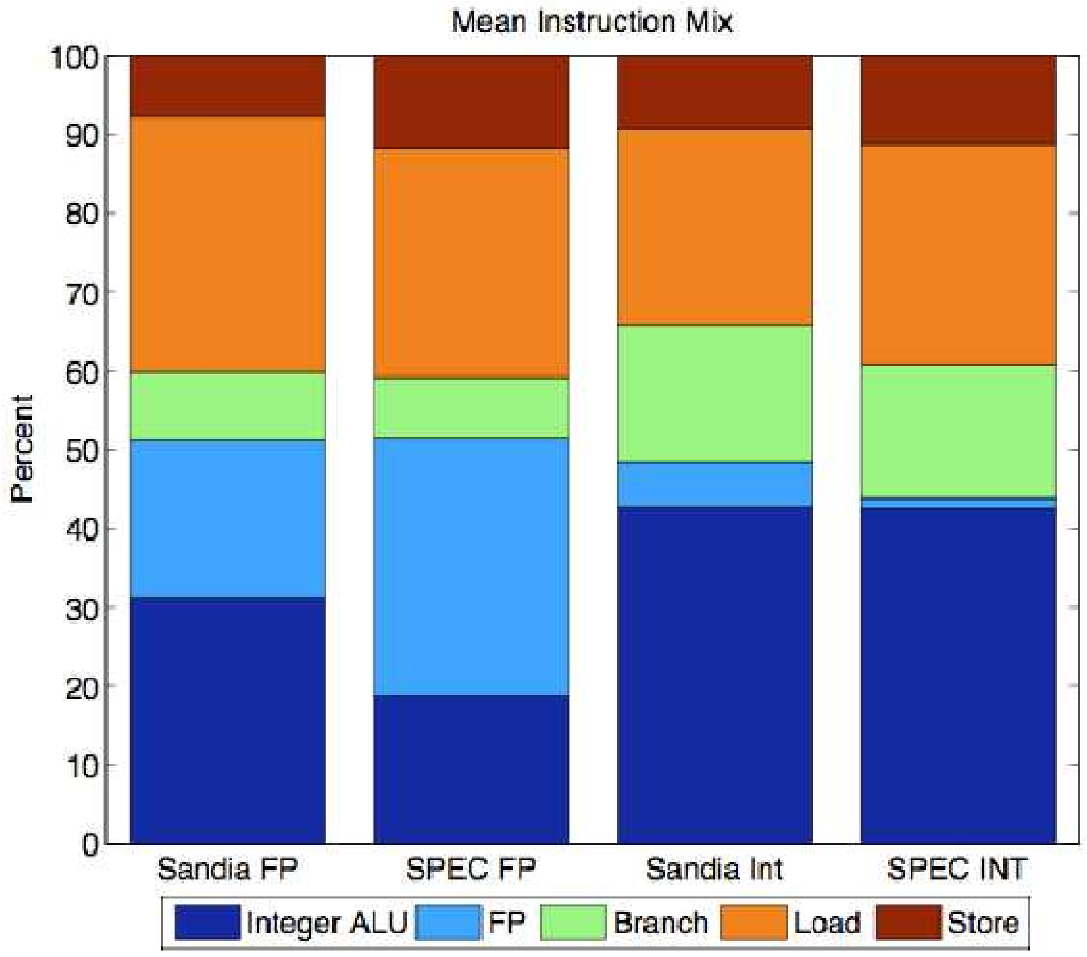
2) Data must inform the algorithm

HPC Applications: Locality Challenges



Slide Credit: Richard Murphy (SNL) and Peter Kogge (Notre Dame)

HPC Applications: Locality and Instruction Mix



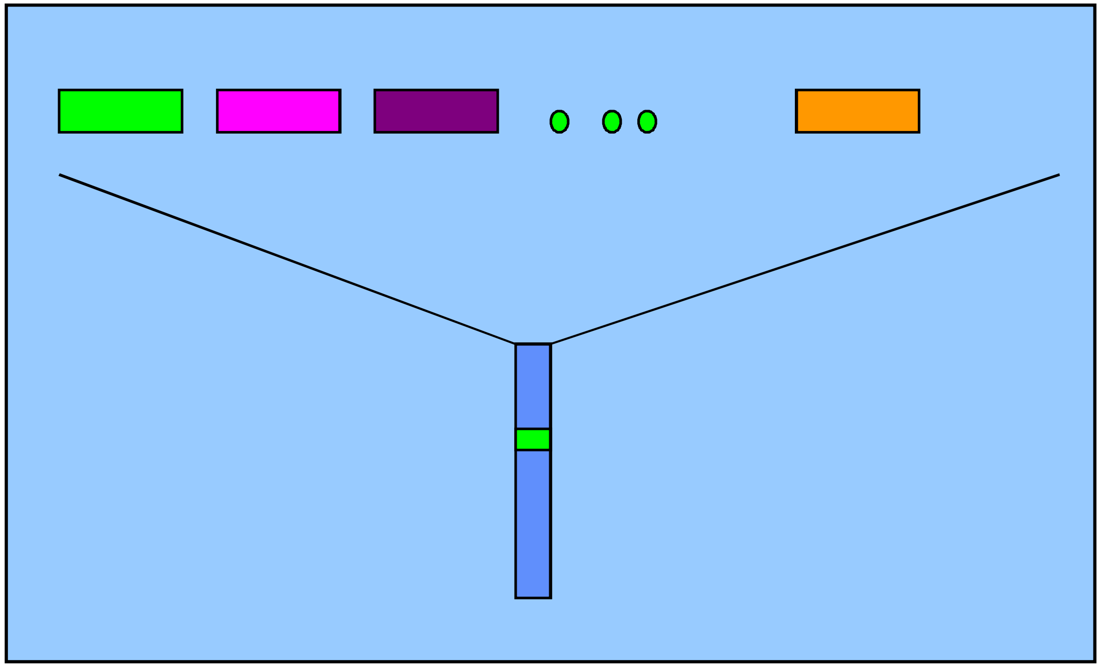
Slide credit: Arun Rodrigues (SNL)

Graph 500

- **Currently: simple challenge: Breadth-First Search (BFS)**
  - Not representative of typical graph analysis
  - Complex filtering at each level, early stops, fine-grain operations would make it more realistic
- **Bulk-synchronous parallel (BSP) operation: compute, then communicate**
  - No special architecture needed
  - SciComp machines win, but nobody does particularly "well"
- **Biggest development so far: data exploited to produce 10X algorithmic improvement**
- **More representative algorithm kernel benchmarks on the way**

Sandia National Laboratories

The Burton Smith/Cray "Threadstorm" Processor



No Processor Cache;
Chip real estate goes to thread contexts

Latency Tolerant:

important for Graph Algorithms

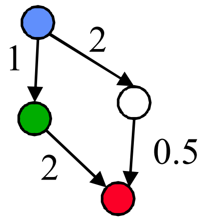
The **Cray XMT** combines up to 8192 Threadstorm processors with the Red Storm (Cray XT4/5) network.

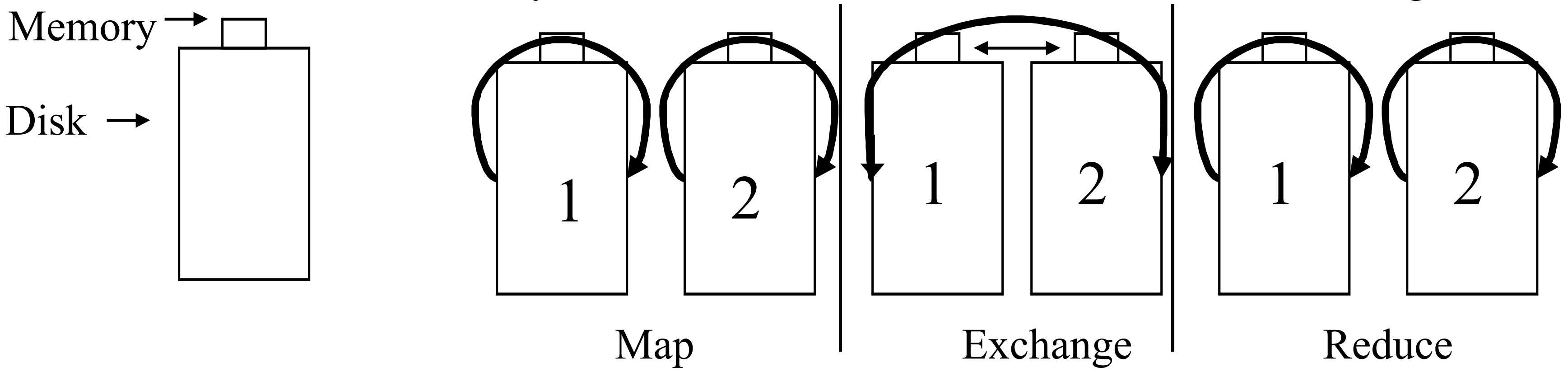
Hashed Memory

Single-Source Shortest Paths (SSSP)

Google demonstrates Map/Reduce on SSSP; what is the cost?

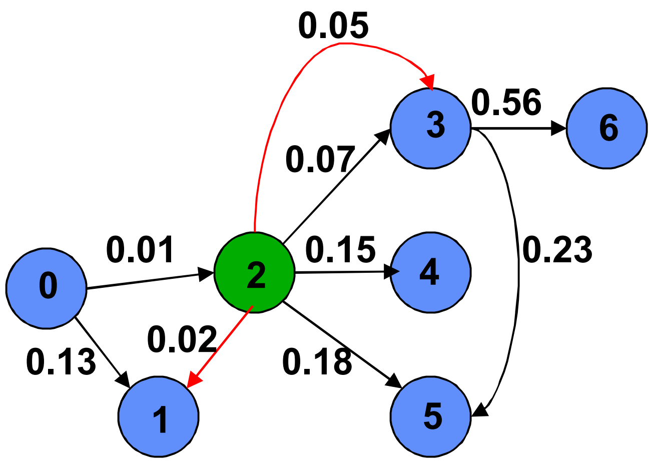
– Case study: Dijkstra's single-source shortest paths algorithm
  - Basic algorithm (much detail omitted)
    1. Start with the source and "settle" it
    2. "relax" edges leading out of most recently settled vertex
    3. Find next closest vertex and settle it, goto 2.
  - Map/Reduce algorithm (much detail omitted)
    1. Do a Map/Reduce pass to relax all edges (near the source or not)
    2. If any node's distance from the source decreased, goto 1.





Δ - stepping algorithm [Meyer, et al. 2003]

- *Label-correcting* algorithm: Can relax edges from unsettled vertices
- Δ - stepping: "approximate bucket implementation of Dijkstra's algorithm"
- Δ: bucket width
- Vertices are ordered using buckets representing priority range of size Δ
- Each bucket may be processed in parallel



Process "light" (red) edges in parallel, then "heavy" edges in parallel

Slide credit: Kamesh Madduri, Lawrence Berkeley Laboratories

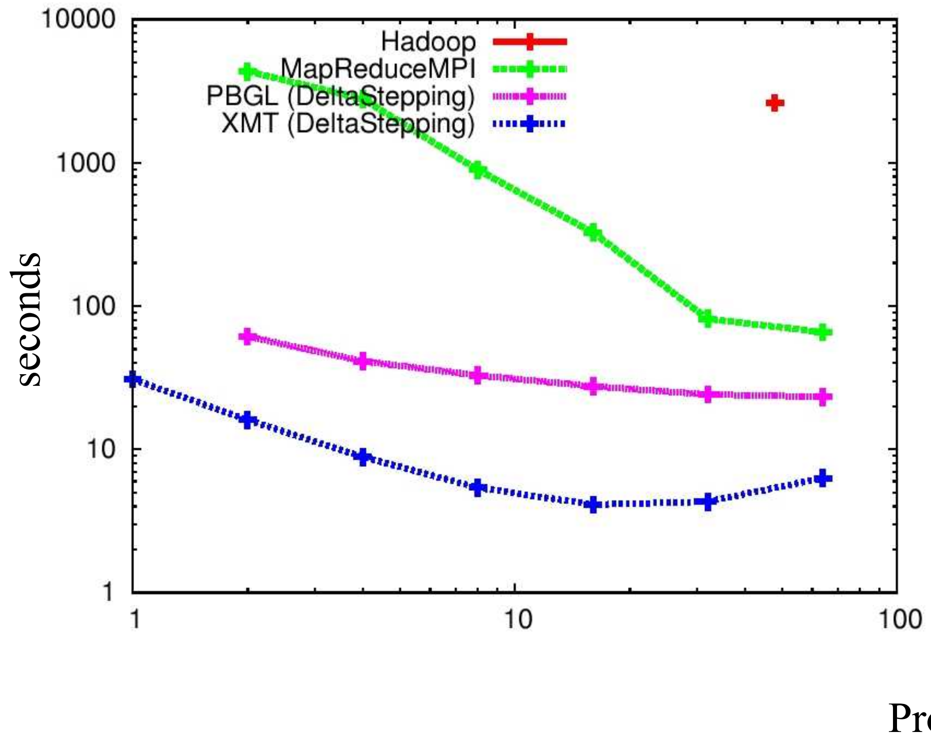
SSSP Discussion

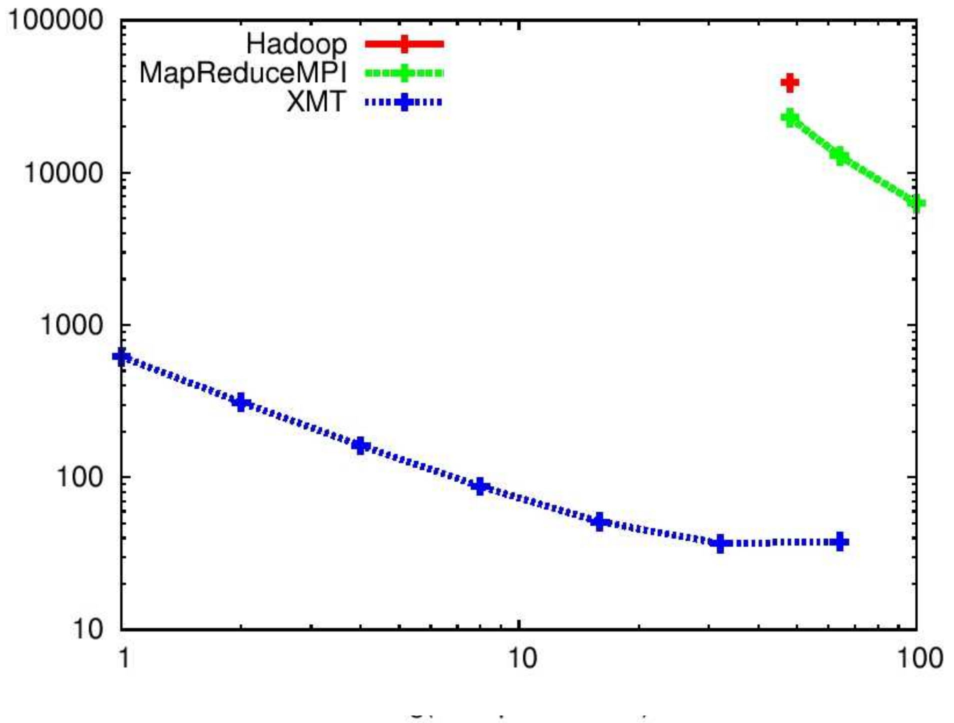
- Hadoop is open-source software for cloud computations
- MapReduceMPI keeps problem mostly in memory
- The "Parallel Boost Graph Library" implements Delta-Stepping on Traditional (distributed-memory) HPC
- The XMT version of delta stepping (a C code by Kamesh Madduri) is the fastest known parallel implementation of SSSP.

K. Madduri, D.A. Bader, J.W. Berry, and J.R. Crobak, "An Experimental Study of A Parallel Shortest Path Algorithm for Solving Large-Scale Graph Instances," *Workshop on Algorithm Engineering and Experiments* (ALENEX), New Orleans, LA, January 6, 2007.

- Our experiments here are on realistic data with power-law degree distributions

SSSP Results





~30 million edges, power-law

~0.5 billion edges, power-law

Plimpton’s “Olympic Medal” Metaphor

- **Olympic Models: Gold, Silver, and Bronze**

- **Computing Solutions: Gold, Aluminum, Plywood**
  - This conference: Gold
  - Lower top-500: Aluminum
  - Google, Facebook Big Data: Plywood (racks of cheap compute)

Model	$/PByte	TB/core	Pb/Pflop	IOPs/PB
Gp;d	$10M	0.044	5	~100K
Aluminum	$2.5M	0.25	40	~500K
Plywood	$0.3M	1+	100+	~200K

Thanks: Steve Plimpton, Sandia National Laboratories

Sandia National Laboratories

Summary

- **“Big Data” often takes the form of relational data (graphs)**
  - Different context has different priorities, constraints
  - If HPC used, alternative architectures may or may not be necessary

- **BSP vs. non-BSP is a key issue for HPC graph analysis**

- **“Plywood” computing platforms suffice for Big Data for now**
  - What might change this?

jberry@sandia.gov