Informatics & HPC:

Tomorrow's Applications Meet Yesterday's Technologies

Bruce Hendrickson

Jonathan Berry

Will McLendon

Tammy Kolda

Sandia National Labs

Future DOE HPC Needs

- The past is mostly about physical simulation
- The future will be more diverse
- Data-centric computing is on the ascendancy
  - » Experimental data
  - » Simulation data
  - » National security data
- Computing for decision support
- Future DOE HPC applications include
  - » Complex search capabilities
  - » Data mining
  - » Machine learning

Context

- **Data computing is multifaceted**
- **This talk is not about …**
  - » **data management & retrieval**
  - » **natural language processing**
  - » **metadata**
  - » **data fusion**
  - » **uncertainty**
  - » **provenance tracking, etc, etc.**
- **Instead, consider the seemingly simpler problem**
  - » **How can our vast experience in HPC be applied to data-centric computing problems?**

Data Computing is Different

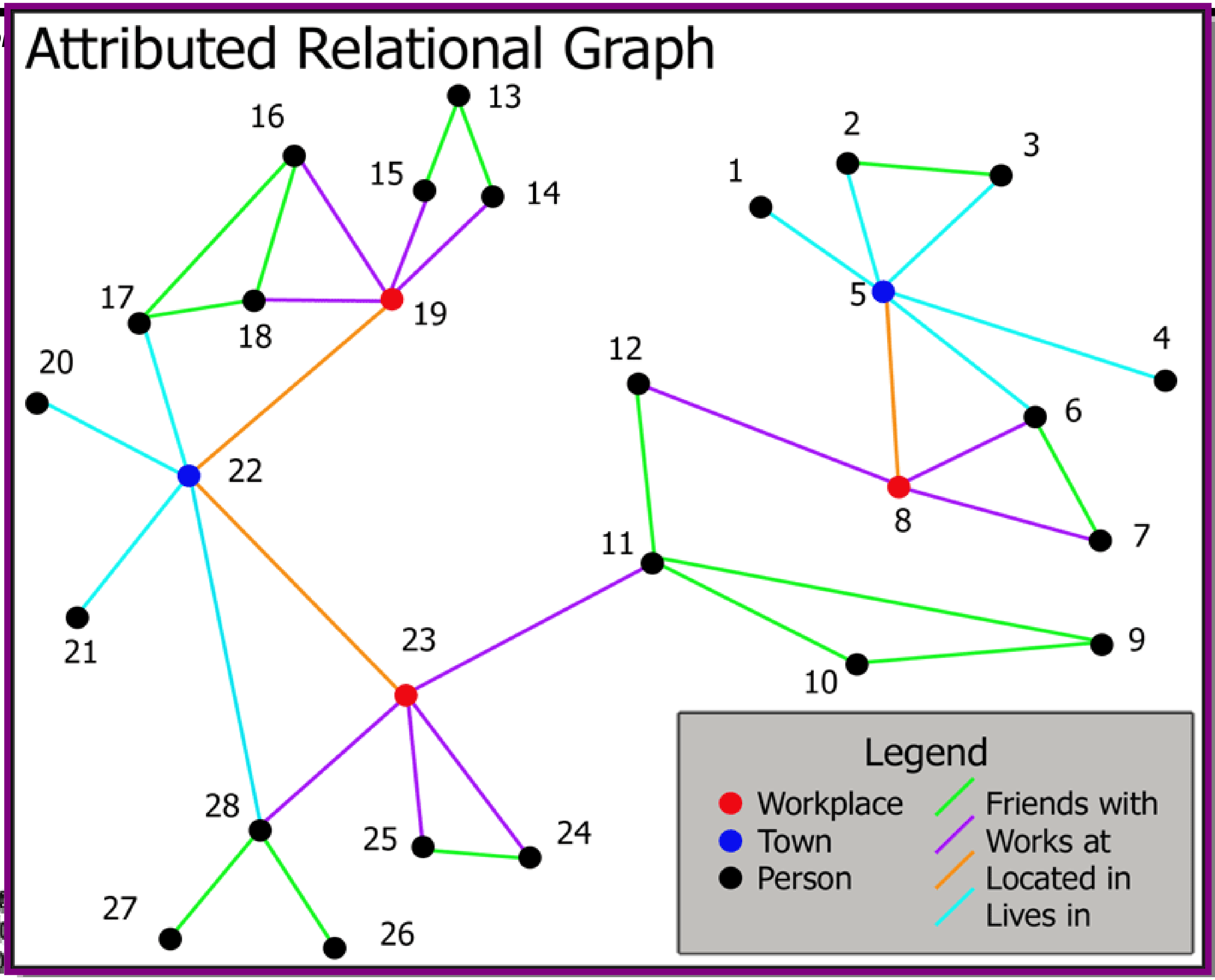
- **Even if we're in core …**
- **Minimal computation to mask access time**
  - » **Low utilization of processors**
- **Complex, unstructured access patterns**
  - » **Poor utilization of memory hierarchy**
- **Complicated data dependencies**
  - » **Difficult to partition well**
  - » **Prefetching likely to be ineffective**
- **The Anti-LINPACK!**

Are We Ready?

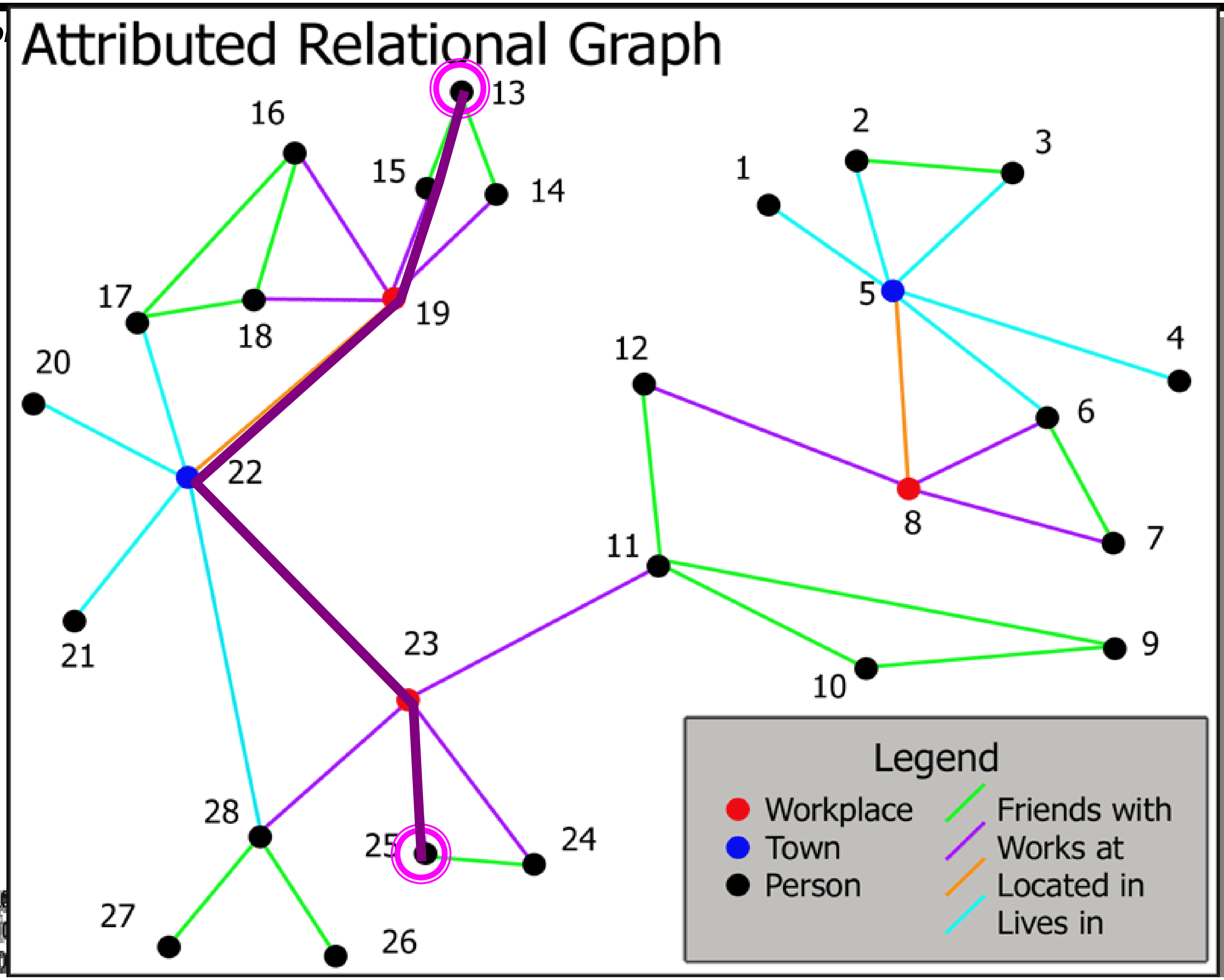
- Existing architectures
  - » Poor I/O
  - » Require cache-exploitable reference patterns
  - » Not benchmarked on data-centric applications

- Existing Programming models & languages
  - » Don't efficiently support random global accesses
  - » Require partitionability into *P* subproblems
    - – This is true of MPI, OpenMP and PGAS Languages (UPC)

Case Study: Graph Informatics



Query Example I: Short Paths



Example II: Pattern Finding

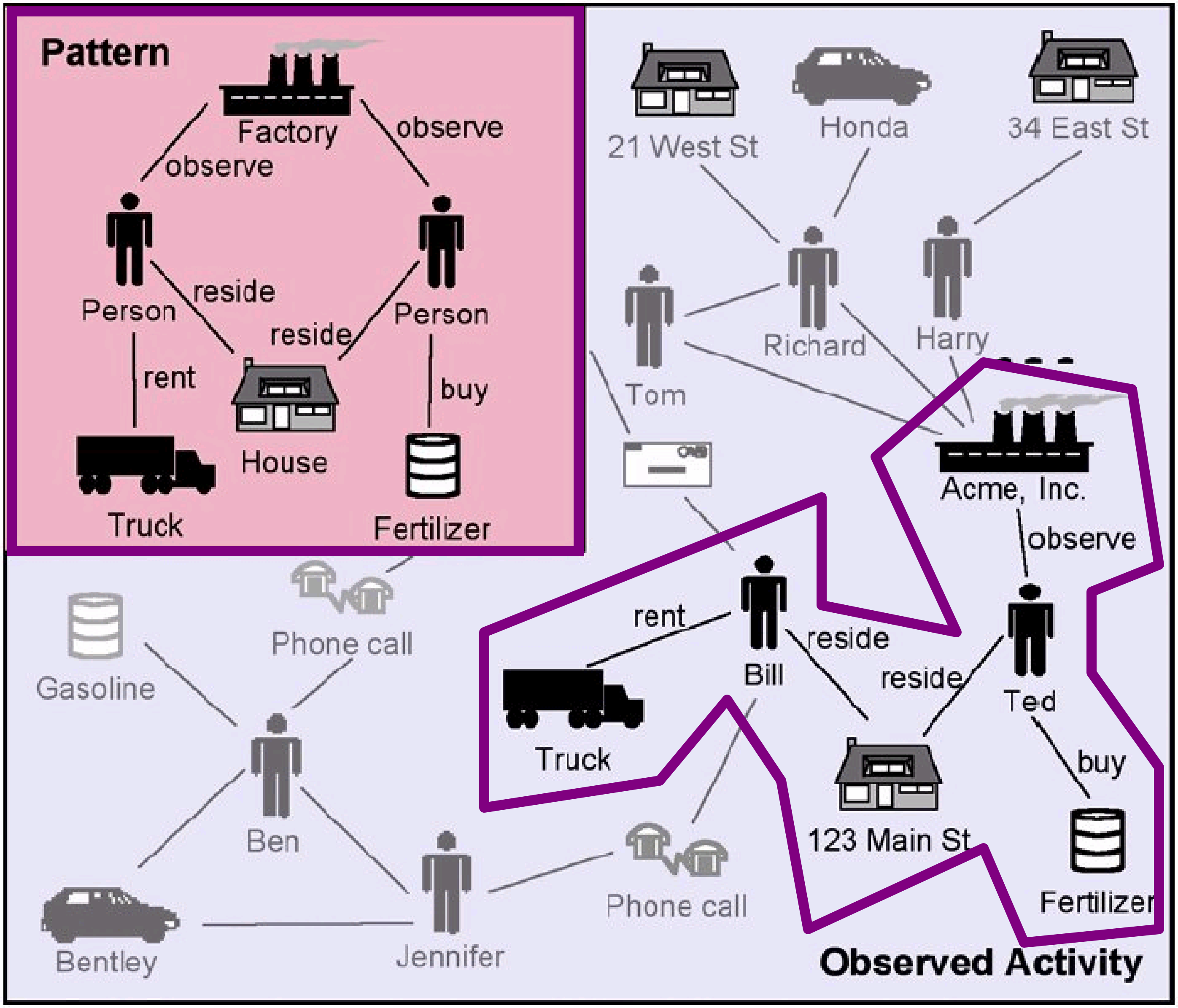


Image Source:
T. Coffman,
S. Greenblatt,
S. Marcus,
Graph-based technologies for intelligence analysis,
CACM, 47
(3, March 2004):
pp 45-47

Graph-Based Informatics: Data

- **Datasets can be enormous**
- **Graphs are highly unstructured**
  - » **High variance in number of neighbors**
  - » **Little or no locality – Not partitionable**
  - » **Experience with scientific computing graphs of limited utility**
- **Queries touch unpredictable subsets of data**

Architectural Challenges

- **Runtime is dominated by latency**
  - » **Random accesses to global address space**
  - » **Perhaps many at once – fine-grained parallelism**
- **Essentially no computation to hide access time**
- **Access pattern is data dependent**
  - » **Prefetching unlikely to help**
  - » **Usually only want small part of cache line**
- **Potentially abysmal locality at all levels of memory hierarchy**

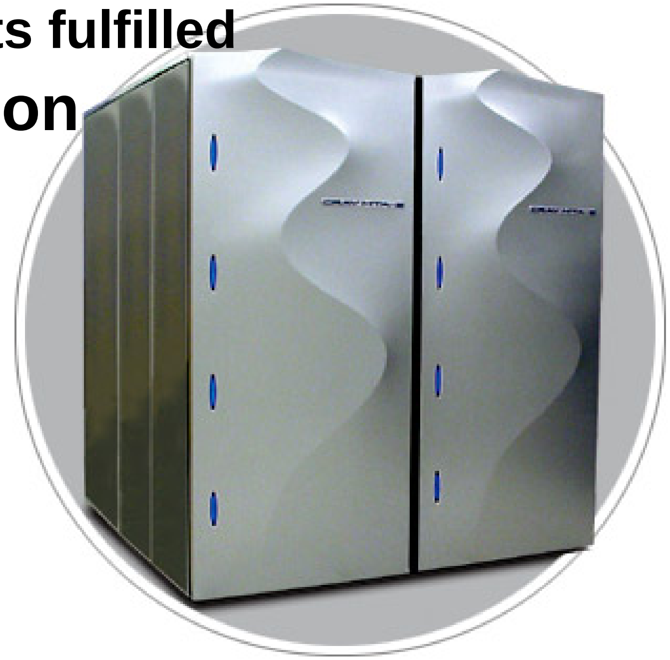
Desirable Architectural Features

- **Low latency / high bandwidth**
  - » **For small messages!**
- **Latency tolerant**
- **Light-weight synchronization mechanisms**
- **Global address space**
  - » **No graph partitioning required**
  - » **Avoid memory-consuming profusion of ghost-nodes**
  - » **No local/global numbering conversions**

- **One machine with these properties is the Cray MTA-2**
  - » **And successor *Eldorado***

How Does the MTA Work?

- Latency tolerance via massive multi-threading
  - » Context switch in a single tick
  - » Global address space, hashed to reduce hot-spots
  - » No cache or local memory.
  - » Multiple outstanding loads
- Remote memory request doesn't stall processor
  - » Other streams work while your request gets fulfilled
- Light-weight, word-level synchronization
  - » Minimizes conflicts, enables parallelism
- Flexible dynamic load balancing
- Notes:
  - » 220 MHz clock
  - » Largest machine is 40 processors



Case Study: MTA-2 vs. BlueGene

- **With LLNL, implemented S-T shortest paths in MPI**
- **Ran on IBM/LLNL BlueGene/L, world's fastest computer**
- **Finalist for 2005 Gordon Bell Prize**
  - » **4B vertex, 20B edge, Erdös-Renyi random graph**
  - » **Analysis: touches about 200K vertices**
  - » **Time: 1.5 seconds on 32K processors**

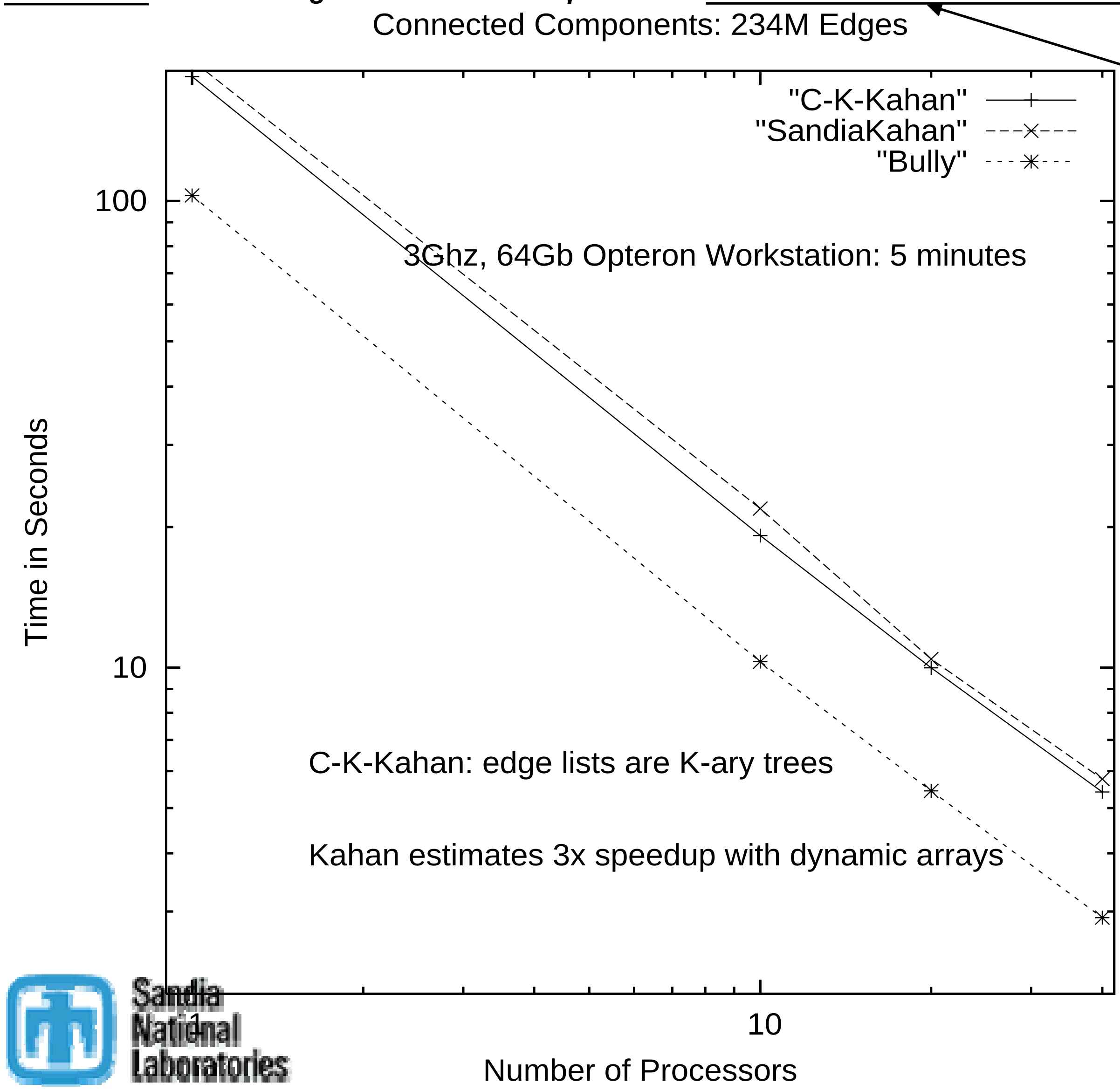
- **Ran similar problem on MTA-2**
  - » **32 million vertices, 128 million edges**
  - » **Measured: touches about 23K vertices**
  - » **Time: .7 seconds on one processor, .09 seconds on 10 processors**

- **Conclusion: 4 MTA-2 processors = 32K BlueGene/L processors**

But Speed Isn't Everything

- **Unlike MTA code, MPI code limited to Erdös-Renyi graphs**
  - » **Can't support power-law graphs; pervasive in informatics**
- **MPI code is 3 times larger than MTA code**
  - » **Took considerably longer to develop**
- **MPI code can only solve this very special problem**
  - » **MTA code is part of general and flexible infrastructure**
- **MTA easily supports multiple, simultaneous users**
- **But … MPI code runs everywhere**
  - » **MTA code runs only on MTA/Eldorado and on serial machines**

MTA-2: Connected Components



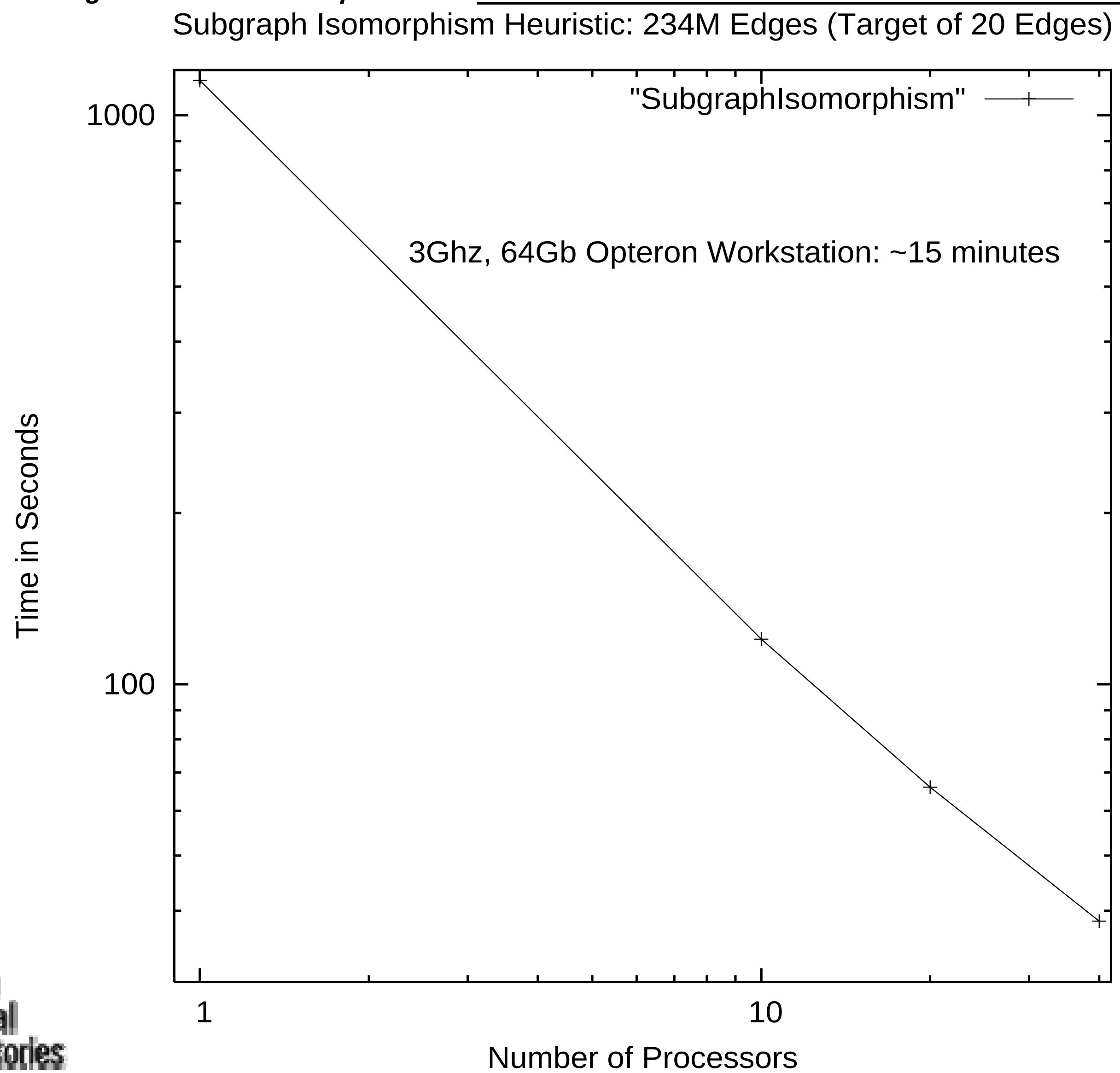
Power Law Graph (highly unstructured)

5.41s

2.91s

MTA-2 Results: Subgraph Isomorphism

Discrete Algorithms & Math Department



Algorithmic Approach

- **Very many small threads (>>*P*)**
  - » **Runtime manages them as a virtual task pool**
  - » **Runtime does virtual-to-physical assignment dynamically**
  - » **Programmer needn't worry about load balancing**
  - » **Dynamic & recursive creation of parallelism**
- **Asynchronous, no global control**
  - » **Thread coordination via word-level locking**
  - » **Fine-granularity enables high degree of parallelism**
- **Serial-looking code**
  - » **But subtle & challenging to get right**

Existing Programming Models

- **Most MPI programs use *Bulk Synchronous Processing approach***
  - » **Independent computation then collective communication**
  - » **Latencies amortized by bundling communication**
- **This doesn't work for graph informatics**
  - » **Parallelism is too fine-grained & asynchronous**
- **Data and computation not easily partitionable**
  - » **Profusion of ghost nodes or expensive rendezvous in MPI**
- **Want large number of small, *virtual* threads**
  - » **No major language currently supports this**

Conclusions

- **Scientific simulation is from Mars,**
  Data-centric computing is from Venus

- **We'll need to revisit all our HPC assumptions**
  - » **Architectures**
  - » **Computing models**
  - » **Languages, etc.**

- **What an exciting time to be in HPC!**

Acknowledgements

- **Thanks to Keith Underwood, Scott Kohn, Mike Merrill, Candy Culhane, Simon Kahan, David Bader, Bill Carlson, Richard Murphy.**

- **bah@sandia.gov**
- **www.cs.sandia.gov/~bahendr**

- **Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the US DOE under contract DE-AC-94AL85000. This work was funded by Sandia's LDRD Program.**
- **SAND 2006-2368C**