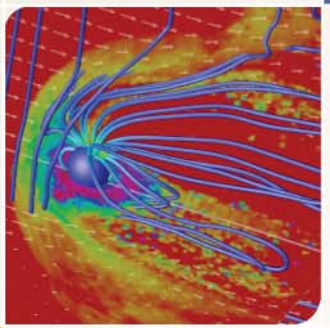
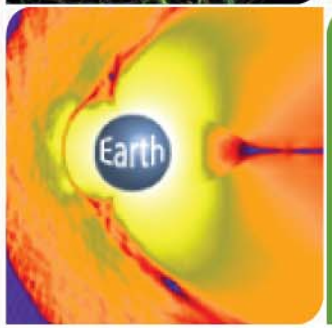
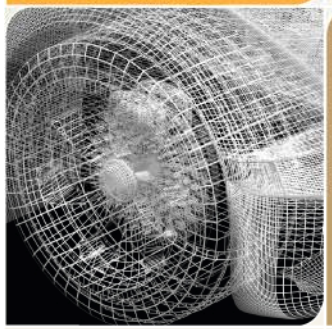



Exascale Applications: An integrative approach

Next-Generation Scalable Applications: When MPI-only is not enough

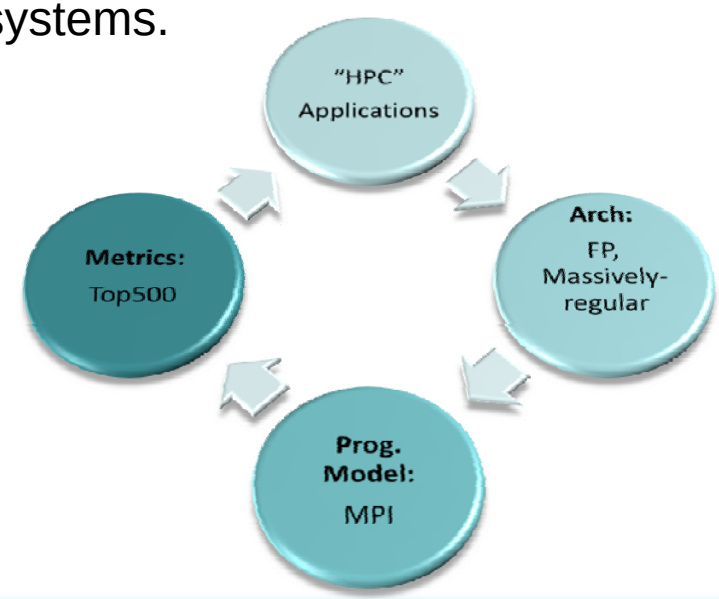
David A. Bader

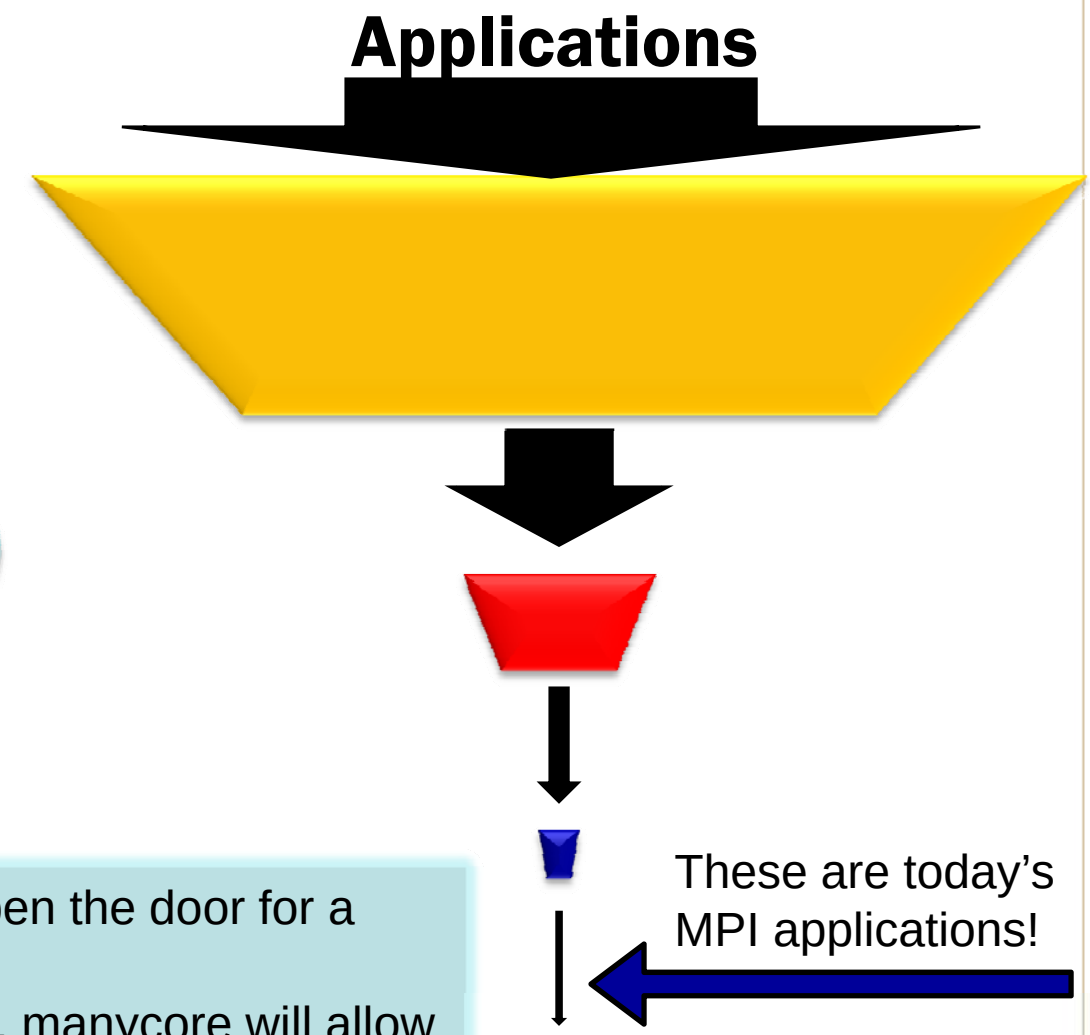


Georgia Tech
College of Computing
Computational Science and Engineering

HPC's success has been its failure!

For decades, HPC has been on a vicious cycle of enabling applications that run well on HPC systems.





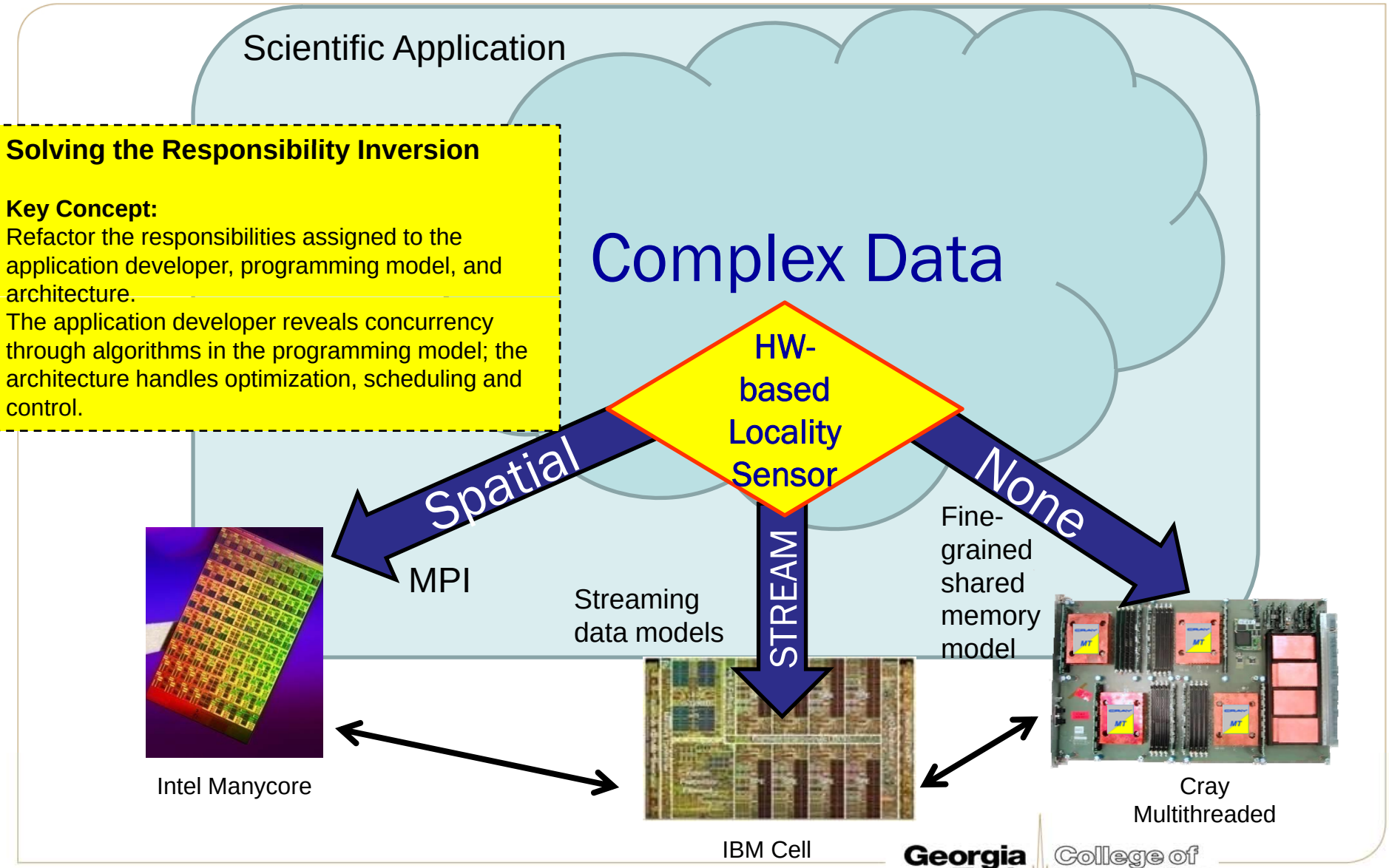
➔The manycore disruption can open the door for a revolutionary transformation!

- For the first time in **decades**, manycore will allow innovation in real-world algorithms

Looking Ahead: Grand Challenge Applications

- Driving real-world applications are not just traditional HPC:
  - health care, transportation, energy, proteomics, security, data sciences, informatics, …
- FLOPS are free, Data is the challenge!
- Fundamental abstractions are more irregular and complex than sparse/dense matrices
  - Very sparse, very high-dimensional data
  - For example, there are few (if any) efficient distributed-memory parallel implementations of even the simplest algorithm for sparse, arbitrary graphs!
- **We must integrate across algorithms, programming models, and architectures, to address the growing requirements of grand challenge applications**

Designing A Data-Centric Exascale System



Solving the Responsibility Inversion

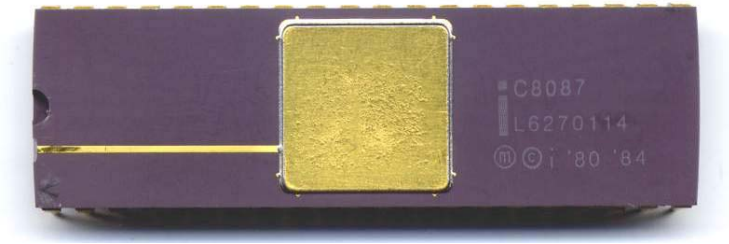
Key Concept:
Refactor the responsibilities assigned to the application developer, programming model, and architecture.
The application developer reveals concurrency through algorithms in the programming model; the architecture handles optimization, scheduling and control.

Accelerators are not a new phenomenon

- 1980-1: Intel introduced a co-processor

8088 CPU could invoke the 8087 accelerator for fast fp operation

  - Intel C8087
  - 60 new instructions
  - 50 KF/s



... 80287 co-processor, 80387 co-processor

- 1989: Intel 80486DX moves FP on-chip
- We should embrace **accelerator-augmented systems:**
  - Several solutions: Standardize an API for offloading work to accelerator-augmented systems (e.g. Cell, GPUs, ClearSpeed), augment the ISA, Leverage tuned libraries

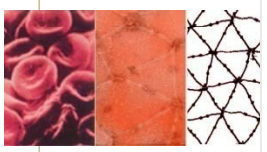
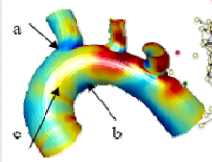
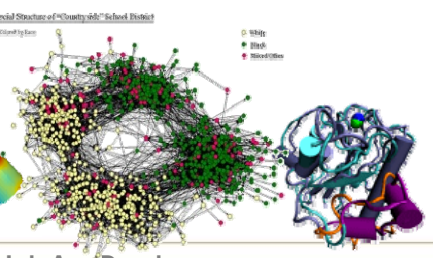
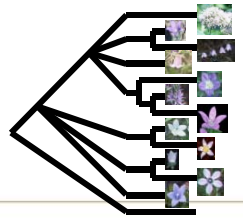
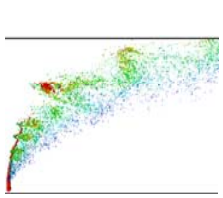
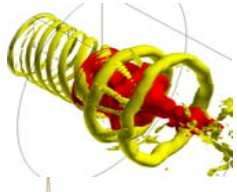
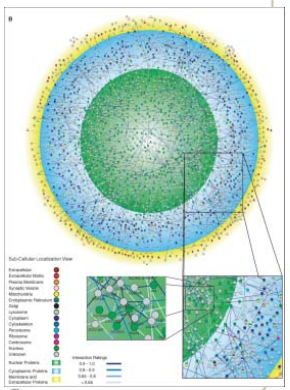
Enabling 21st Century Applications

(slightly modified slide presented at SC2001 panel)

- Current HPC systems are designed for physics-based simulations that use
  - Floating-point, linear algebra
    - Top 500 List measures Linpack!
  - Regular operations (high-degrees of locality)
    - e.g., Matrices, FFT, CG
  - Low-order polynomial-time algorithms

- Focus of current HPC algorithms:
  - Dense linear algebra
  - Sparse linear algebra
  - FFT or multi-grid
  - Global scatter-gather operations
  - Dynamically evolving coordinate grids
  - Dynamic load-balancing
  - Particle-based or lattice-gas algorithms
  - Continuum equation solvers

- [Newer scientific] apps will require
  - Integer performance
    - Strings, lists, trees, graphs
  - Combinatorics
    - Optimization
    - Computational geometry
  - Irregular data accesses
  - Dynamic programming, backtracking
  - Heuristics and solutions to NP-hard problems

Understanding Dynamic Spatio-Temporal Complex Networks

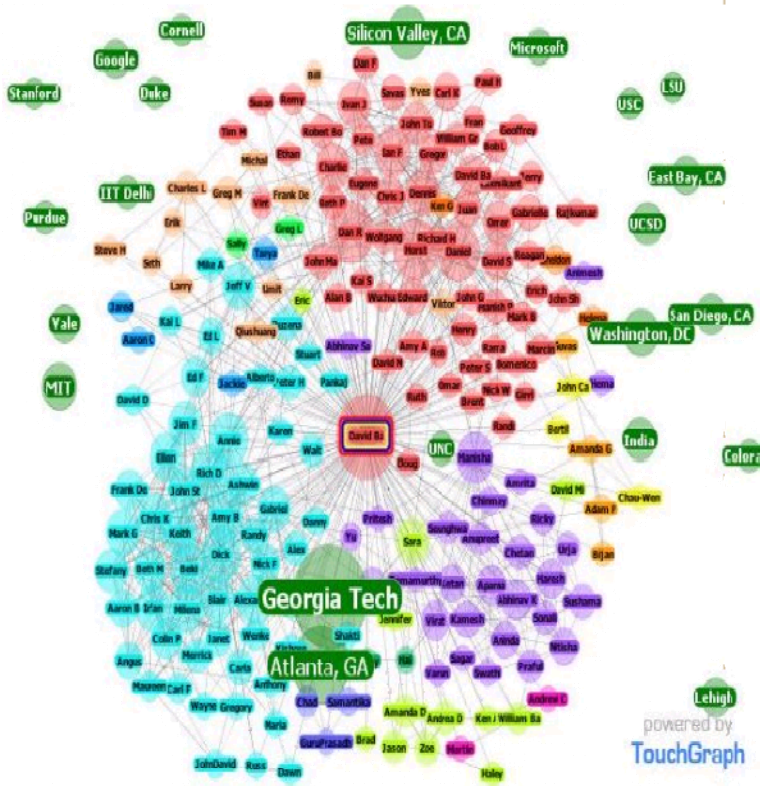
- Sample Applications
  - Health care → disease spread, detection of epidemics,
  - Massive social networks → energy conservation requires social change, modeling pandemic spread, transportation and evacuation
  - Intelligence → business analytics, anomaly detection, security, knowledge discovery from massive data sets
  - Systems Biology → understanding complex life systems, drug design, microbial research
- New data-centric algorithms are required for the understanding the dynamic structure of spatio-temporal data. (These are fundamental and have no static analogues.)
  - Genesis and dissipation of communities
  - Allegiance switching
  - Phase change in the network structure

Scenario 1: Online Social Networking

Friendship networks:

(MySpace, Facebook, orkut)

- large-scale
- evolving
- data-rich



An online interaction network snapshot

Scenario 2: Intelligence

- [Krebs ’04] Post 9/11 Terrorist Network Analysis from public domain information
- Plot masterminds correctly identified from centrality of interaction patterns

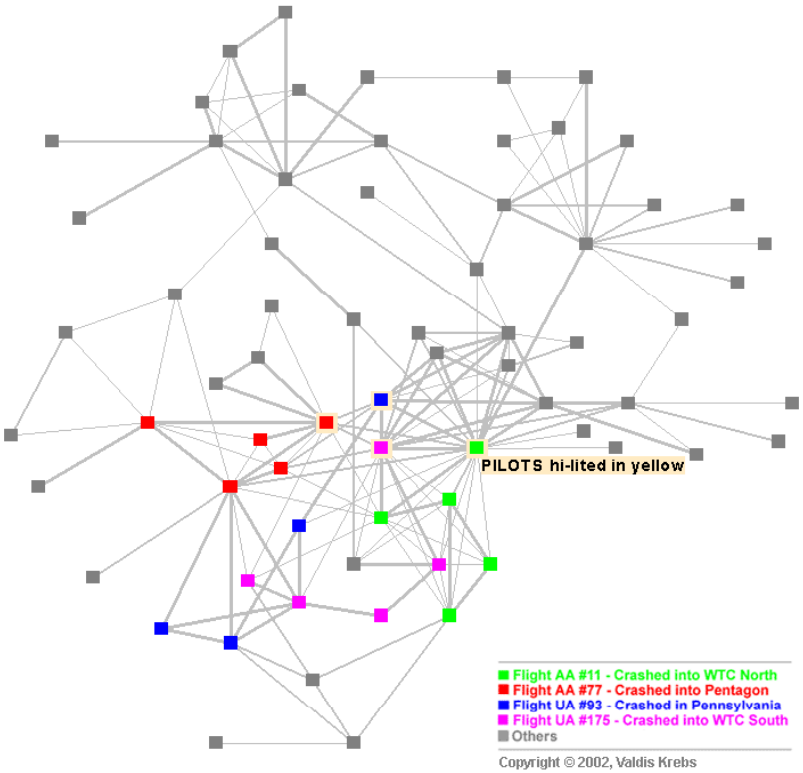


Image Source: http://www.orgnet.com/hijackers.html

Scenario 3: Intelligence

- A global view of entities is often more insightful
- Detect anomalous activities by exact/approximate graph matching

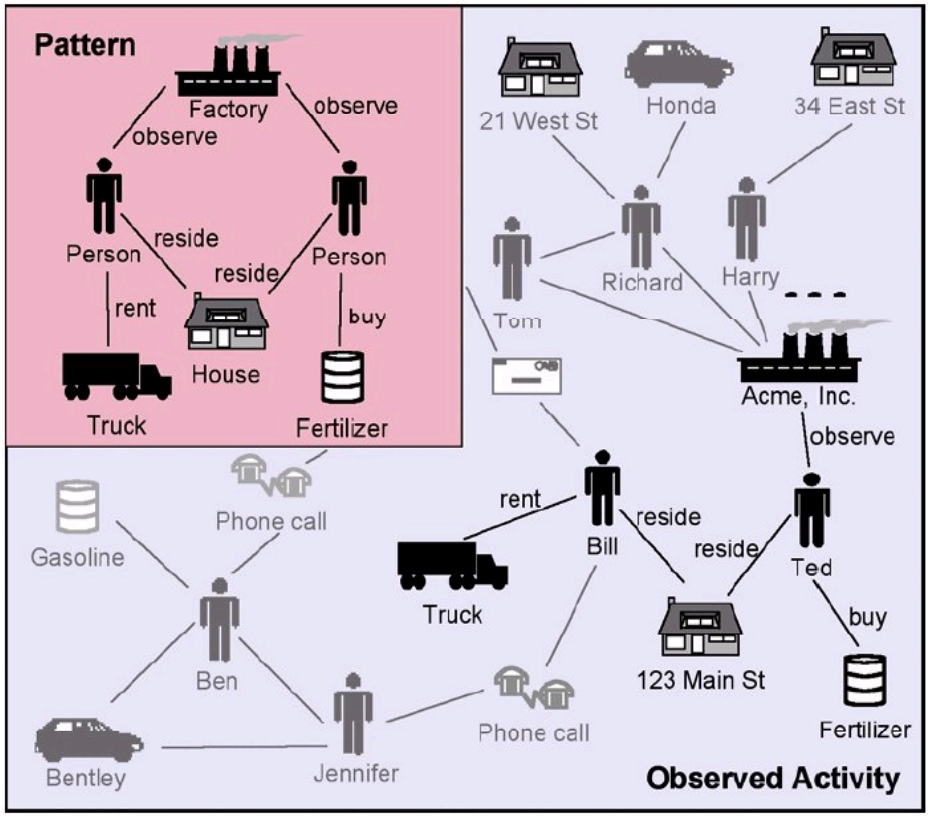
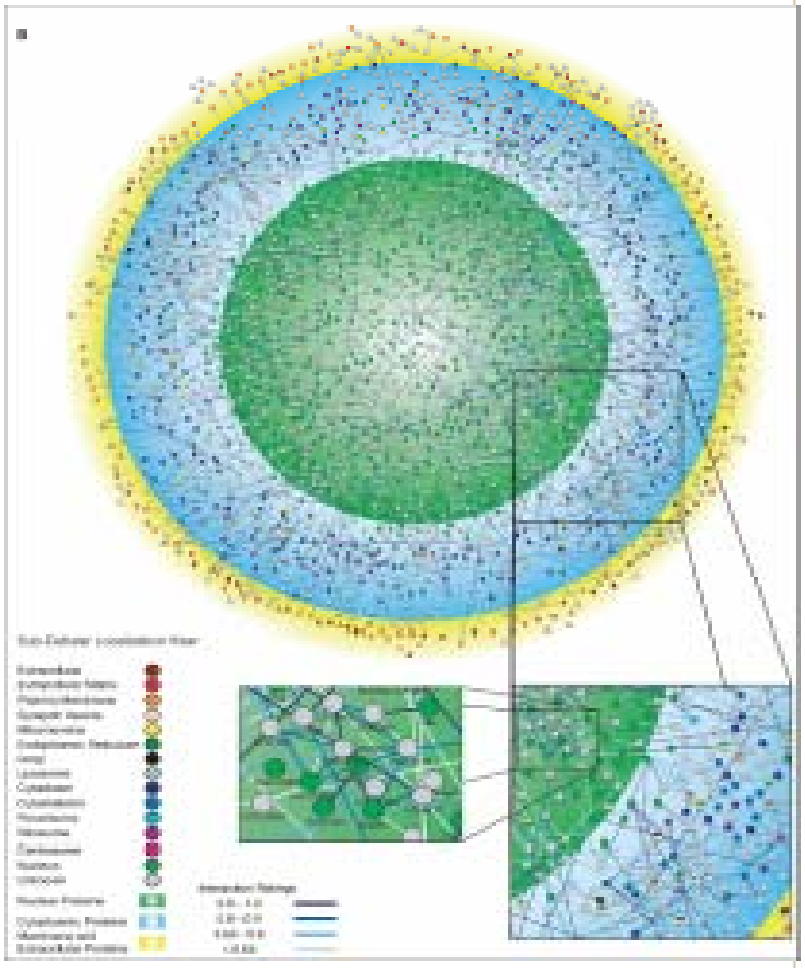


Image Source: T. Coffman, S. Greenblatt, S. Marcus, Graph-based technologies for intelligence analysis, CACM, 47 (3, March 2004): pp 45-47

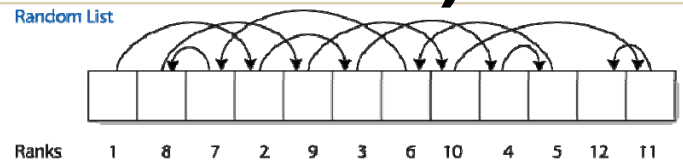
Scenario 4: Systems Biology

- Study of the interactions between various components in a biological system
- Graph-theoretic formulations are pervasive:
  - Predicting new interactions
  - Functional annotation of novel proteins
  - Identifying metabolic pathways
  - Identifying new protein complexes

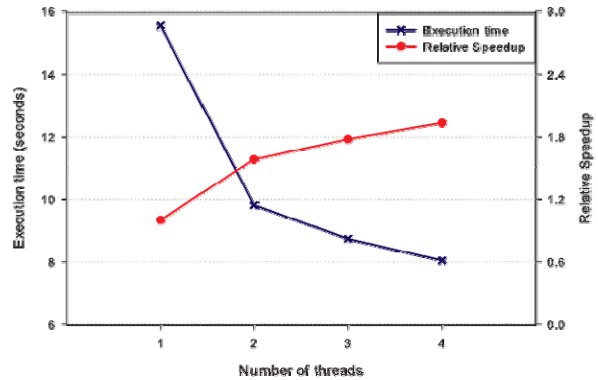


Manycore Challenge: List Ranking (or a challenge for the MPI enthusiasts….)

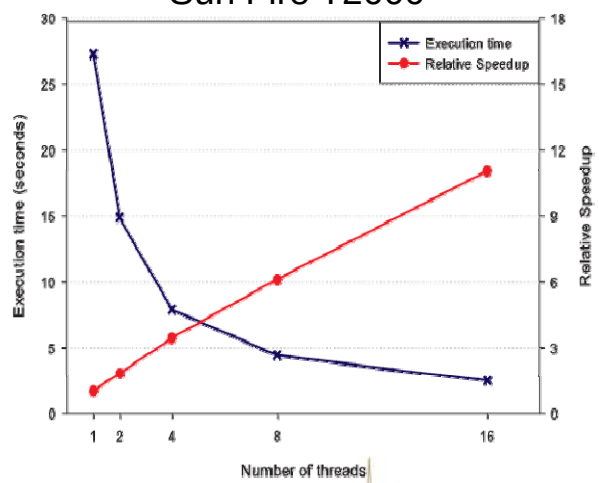
- Challenge: Given a linked list stored in memory, find the distance from each node to the head
  - Sequential approach is trivial (2 lines of code, linear time)
  - Linear speedup with the number of processors in theory (PRAM)
  - No speedup has **ever** been reported using MPI
- Rationale: List ranking is the basis for many irregular parallel algorithms, and is representative of many client applications.



Intel dual-core Xeon 5150



Sun Fire T2000



Input: Random List of 2^{26} elements

SWARM: SoftWare and Algorithms for Running on Manycore

- ➢Supported by Microsoft Research Faculty Award in Parallelism and Concurrency
- ➢Library of efficient implementations of parallel programming primitives and example kernels
  - Prefix-sums, pointer-jumping, list ranking, divide and conquer, pipelining, graph algorithms, symmetry breaking, graph algorithms

- Computational model for analyzing algorithms on multi&manycore systems
- Portable
  - Microsoft Visual Studio, Linux, AIX, Solaris
  - Intel Xeon, AMD Opteron, IBM Power6, Sun US T1
- Shared Source under the Microsoft Permissive License (Ms-PL)
  - http://multicore-swarm.sourceforge.net/

List Ranking

```
i ← head ; val ← 0
while (i ≠ EOL)
   i.rank ← val++; i ← i.next
```

- Efficient parallel source code (for SMP and MTA/XMT architectures) is available from http://www.cc.gatech.edu/~bader
- No known efficient MPI implementations.

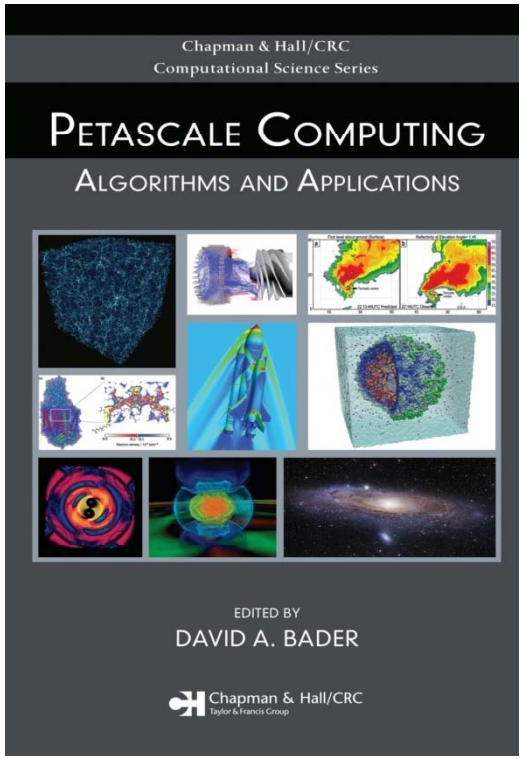
Final Thoughts

- MPI will continue to be an important model for traditional scientific applications. To address the needs of the new grand challenges, however:
  - We've had a "**responsibility inversion**" ! We need to refactor the responsibilities given to the application developer, programming models, OS and runtime systems, and architectures.
- Raise the level of abstraction. Users should specify desired goals (i.e. the high-level problem) rather than how to perform/schedule/coordinate a task (i.e., sequential or MPI/parallel source code)
  - Application developers should specify problems that need to be solved, and annotate high-level algorithms. The compiler/system/architecture should be responsible for selecting, tuning, and optimizing the implementation that takes the best advantage on that architecture.

Acknowledgment of Support

Petascale Computing: Algorithms and Applications (David A. Bader)

- Provides the first collection of articles on petascale algorithms and applications for computational science and engineering
- Covers a breadth of topics in petascale computing, including architectures, software, programming methodologies, tools, scalable algorithms, performance evaluation, and application development
- Discusses expected breakthroughs in the field for computational science and engineering
- Includes contributions from international researchers who are pioneers in designing applications for petascale computing systems



Chapman & Hall/CRC
Computational Science Series, © 2007