Simulation at Extreme Scale

William Gropp
www.cs.illinois.edu/~wgropp

Why Talk About Simulation?

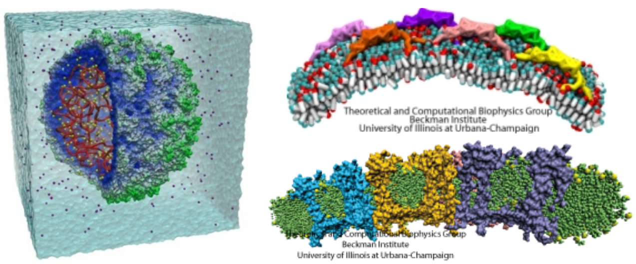
- Big Data requires Big Computing
- Simulation both a consumer and producer of data
- Individual data objects may be huge
  - ♦ Really huge, as in Petabytes
- HPC systems are uniquely capable of processing huge data viewed as a single object
  - ♦ Compared even with large cluster systems
  - ♦ Key feature of HPC systems is very fast interconnect, making HPC system one big machine in a way that clouds are not.

HPC in 2012

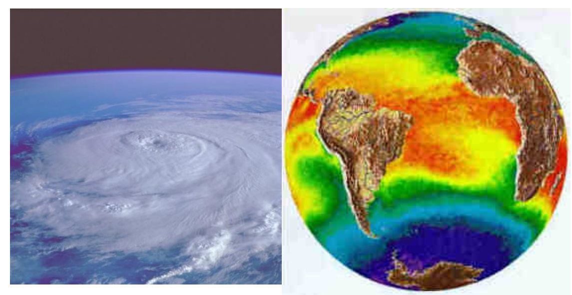
- Sustained PF systems
  - K Computer (Fujitsu) at RIKEN, Kobe, Japan (2011)
  - "Sequoia" Blue Gene/Q at LLNL
  - NSF Track 1 "Blue Waters" at Illinois
  - Undoubtedly others (China, …)
- Focus remains on FLOPS, even though systems are uniquely capable of handling big data
  - There has been a long history of ranking systems by FLOPS
  - Esp TOP500 but also HPCC, others, even Graph500
- NSF asked in 2006 for a *sustained* PetaFLOP system
  - Includes entire application, not just "the fast part"
  - Includes realistic I/O in time
  - Illinois won the award with "Blue Waters"

Sustained Petascale computing will enable advances in a broad range of science and engineering disciplines

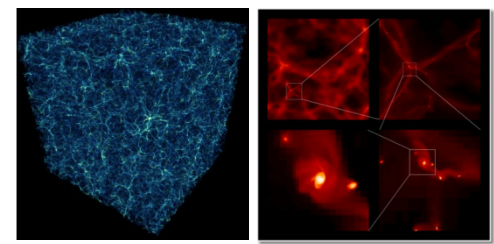
Molecular Science



Weather & Climate Forecasting

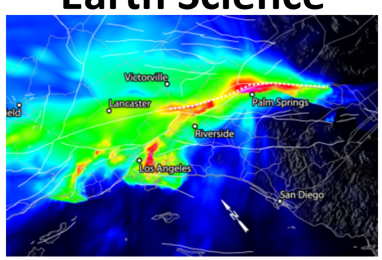


Astrophysics

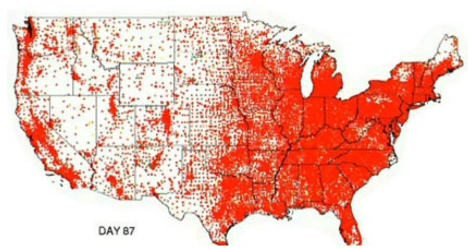


Astronomy

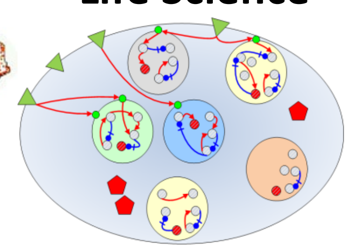
Earth Science



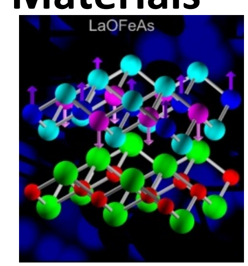
Health



Life Science



Materials



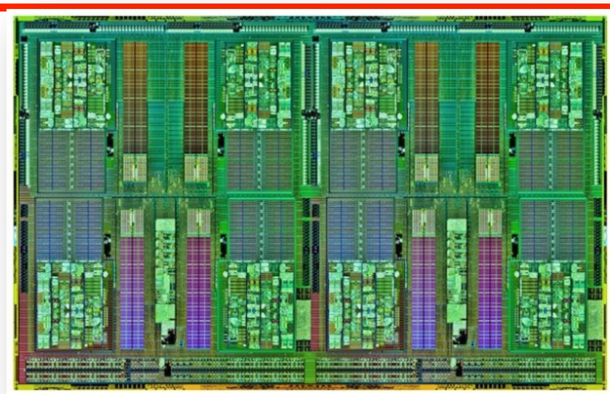
Missing are true data-centric applications
Have one? - http://www.nsf.gov/pubs/2008/nsf08529/nsf08529.htm
or search for NSF PRAC (#1 with duckduckgo)

PARALLEL@ILLINOIS

Blue Waters Science Team Characteristics

Science Area	Number of Teams	Codes	Structured Grids	Unstructured Grids	Dense Matrix	Sparse Matrix	N-Body	Monte Carlo	FFT	Significant I/O
Climate and Weather	3	CESM, GCRM, CM1, HOMME	X	X		X		X		
Plasmas/ Magnetosphere	2	H3D(M), OSIRIS, Magtail/ UPIC	X				X		X	X
Stellar Atmospheres and Supernovae	2	PPM, MAESTRO, CASTRO, SEDONA	X			X		X		X
Cosmology	2	Enzo, pGADGET	X			X	X			
Combustion/ Turbulence	1	PSDNS	X						X	
General Relativity	2	Cactus, Harm3D, LazEV	X			X				
Molecular Dynamics	4	AMBER, Gromacs, NAMD, LAMMPS			X		X		X	
Quantum Chemistry	2	SIAL, GAMESS, NWChem			X	X	X	X		X
Material Science	3	NEMOS, OMEN, GW, QMCPACK			X	X	X	X		
Earthquakes/ Seismology	2	AWP-ODC, HERCULES, PLSQR, SPECFEM3D	X	X			X			X
Quantum Chromo Dynamics	1	Chroma, MILC, USQCD	X		X	X	X		X	
Social Networks	1	EPISIMDEMICS								
Evolution	1	Eve								
Computer Science	1			X	X	X			X	X

Heart of Blue Waters: Two New Chips

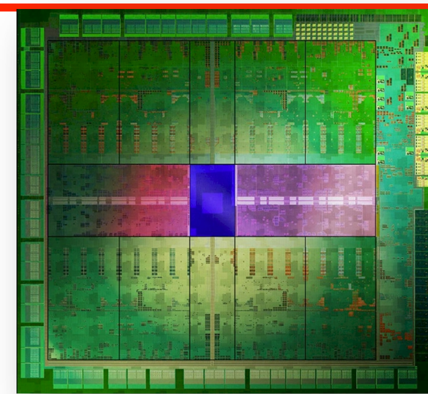


AMD Interlagos

157 GF peak performance

Features:

- 2.3-2.6 GHz
- 8 core modules, 16 threads
- On-chip Caches
  - L1 (I:8x64KB; D:16x16KB)
  - L2 (8x2MB)
- Memory Subsystem
  - Four memory channels
  - 51.2 GB/s bandwidth



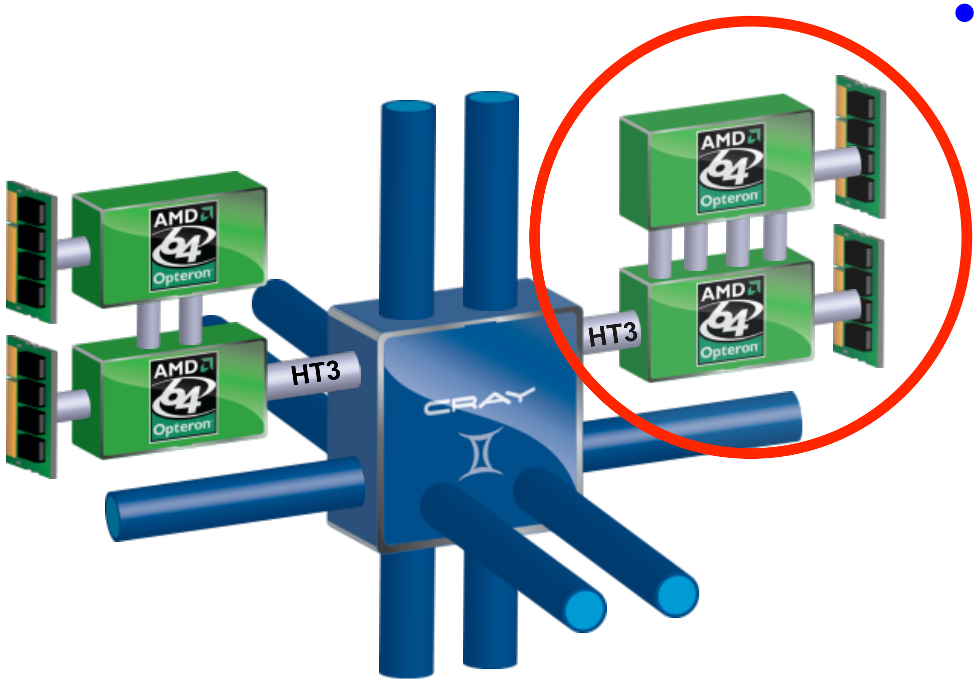
NVIDIA Kepler

1,400 GF peak performance

Features:

- 15 Streaming multiprocessors (SMX)
  - SMX: 192 sp CUDA cores, 64 dp units, 32 special function units
  - L1 caches/shared mem (64KB, 48KB)
  - L2 cache (1536KB)
- Memory subsystem
  - Six memory channels
  - 180 GB/s bandwidth

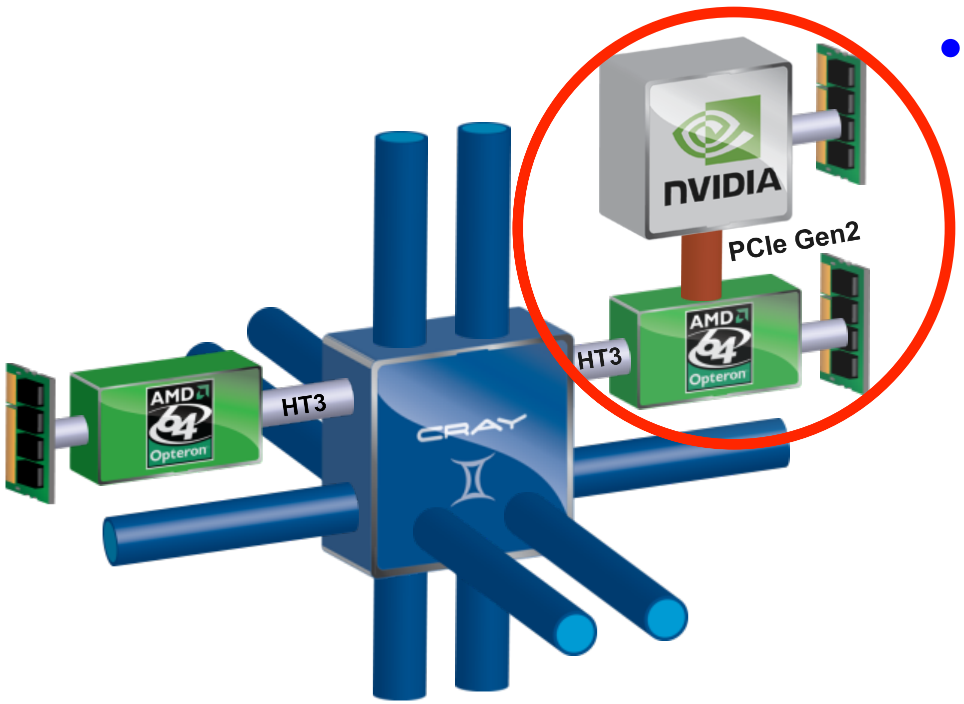
Cray XE6 Nodes



- Dual-socket Node
  - ♦ Two AMD Interlagos chips
    - 16 core modules, 64 threads
    - 313 GFs peak performance
    - 64 GBs memory
      - 102 GB/sec memory bandwidth
  - ♦ Gemini Interconnect
    - Router chip & network interface
    - Injection Bandwidth (peak)
      - 9.6 GB/sec per direction

Blue Waters contains 22,640 Cray XE6 compute nodes.

Cray XK7 Nodes

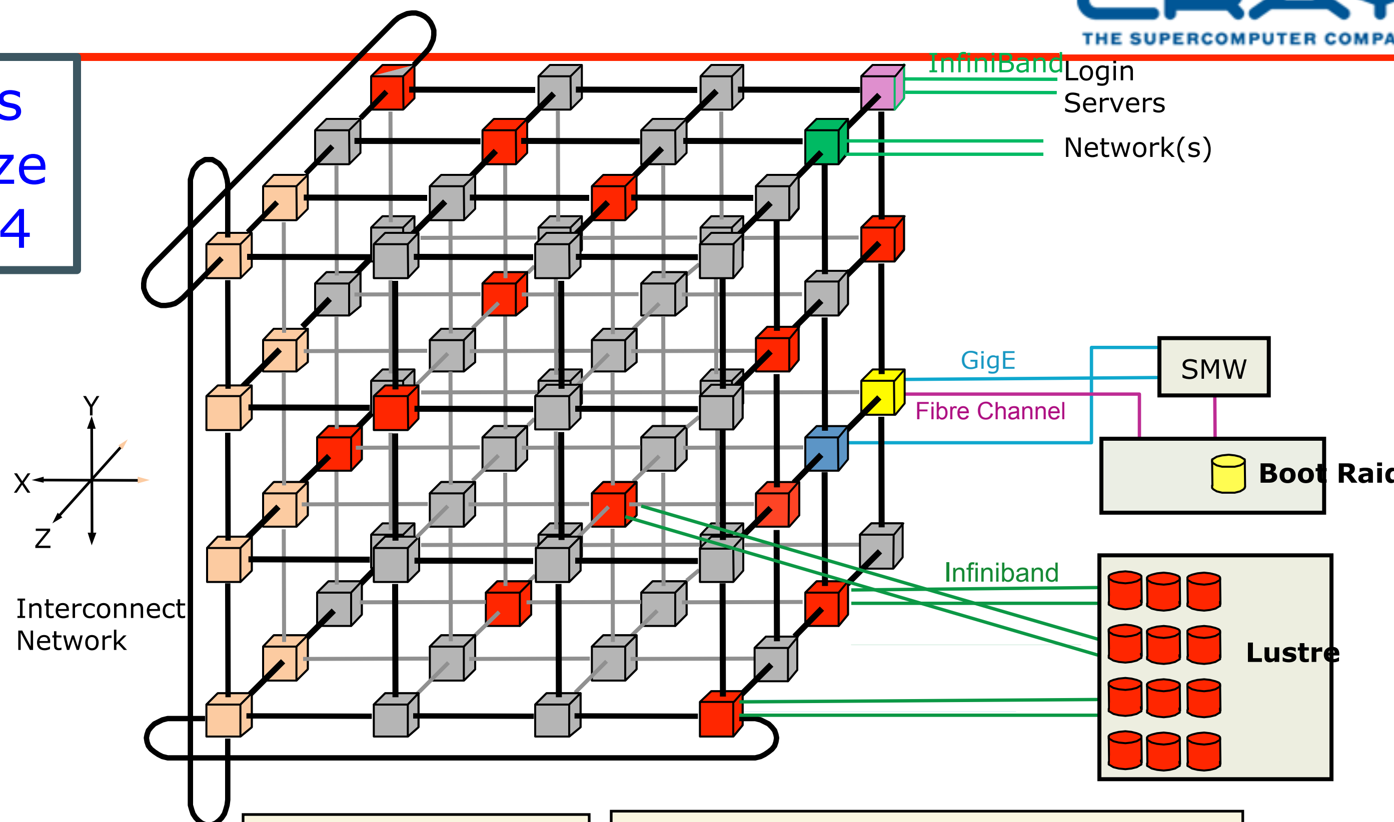


Blue Waters contains 3,072 Cray XK7 compute nodes.

- Dual-socket Node
  - ♦ One AMD Interlagos chip
    - 32 GBs memory
      - 51.2 GB/s bandwidth
  - ♦ One NVIDIA Kepler chip
    - 1.4 TFs peak performance
    - 6 GBs GDDR5 memory
      - 180 GB/sec bandwidth
  - ♦ Gemini Interconnect
    - Same as XE6 nodes

Gemini Interconnect Network

Blue Waters
3D Torus Size
23 x 24 x 24



Compute Nodes
- Cray XE6 Compute
- Cray XK7 Accelerator

Service Nodes

Operating System
- Boot
- System Database

Login/Network
- Login Gateways
- Network

Lustre File System
- LNET Routers

Service Nodes spread throughout the torus

Blue Waters Disk Subsystem

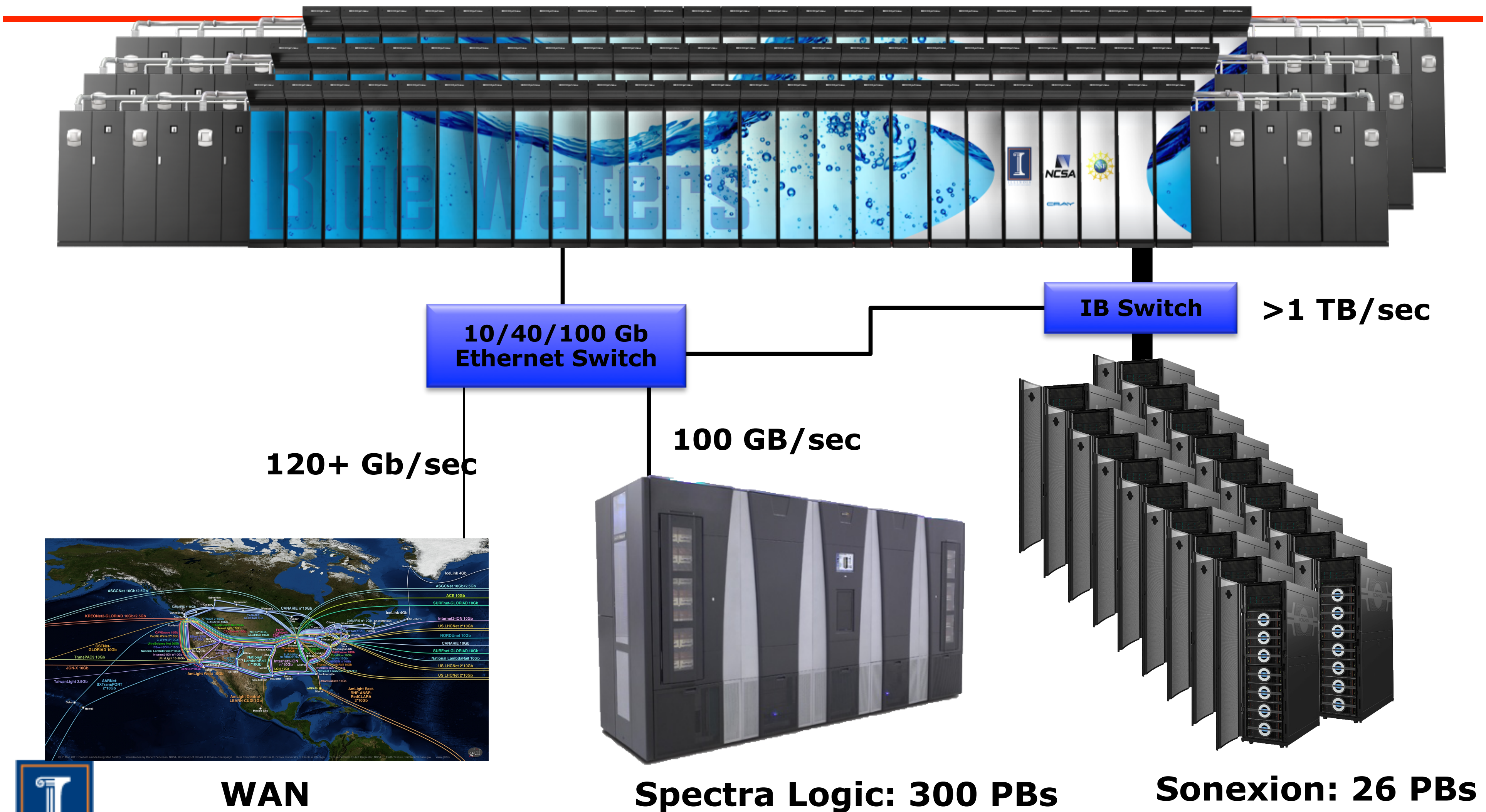
- Cray Sonexion 1600
  - ♦ Lustre file system
  - ♦ Reliable, Modular, Scalable
  - ♦ Fully integrated
    - Servers
    - Disk drives (Scalable Storage Units)
    - QDR Infiniband switches
  - ♦ Hierarchical monitoring
- Blue Waters Disk Subsystem
  - ♦ Capacity: 34.6 PBs (raw), 25.9 PBs (usable)
  - ♦ Bandwidth: >1 TB/s (sustained)

Blue Waters Archive System

- Spectra Logic T-Finity
  - ♦ Dual-arm robotic tape libraries
  - ♦ High availability and reliability, with built-in redundancy
- Blue Waters Archive
  - ♦ Capacity: 380 PBs (raw), 300 PBs (usable)
  - ♦ Bandwidth: 100 GB/sec (sustained)
  - ♦ RAIT for increased reliability

PARALLEL@ILLINOIS

Blue Waters Computing System



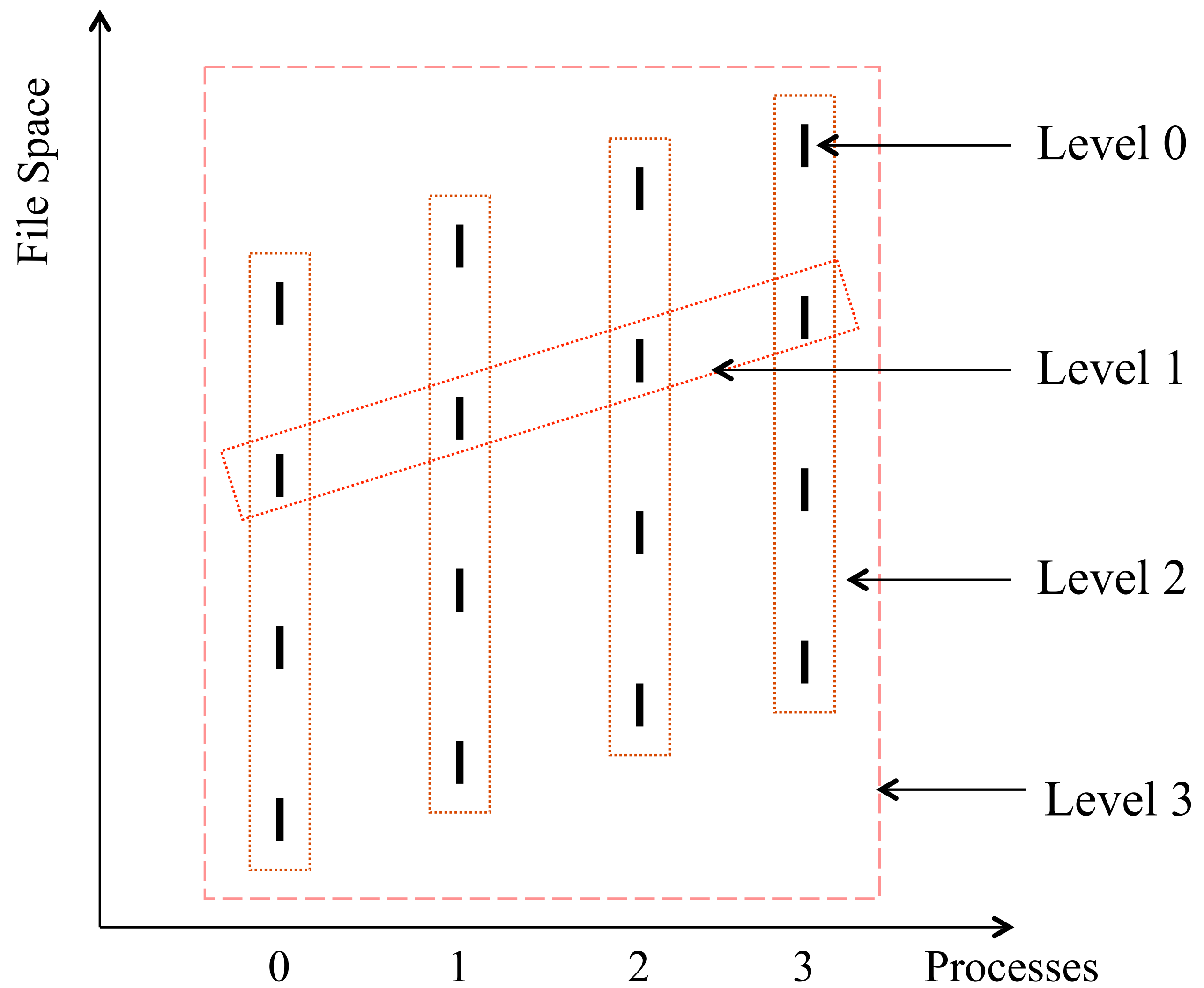
How Do We Make Effective Use of These Systems?

- Better use of our existing systems
  - Blue Waters will provide a sustained PF, but that typically requires ~10PF peak (BW over 11PF peak)
- Improve node performance
  - Make the compiler better
  - Give better code to the compiler
  - Match algorithms/data structures to real hardware
- Improve parallel performance/scalability
- Improve productivity of applications
  - Better tools and interoperable languages, not a (single) new programming language
- Improve algorithms wrt real hardware
  - Optimize for the real issues – data movement, power, resilience, …

Common Themes

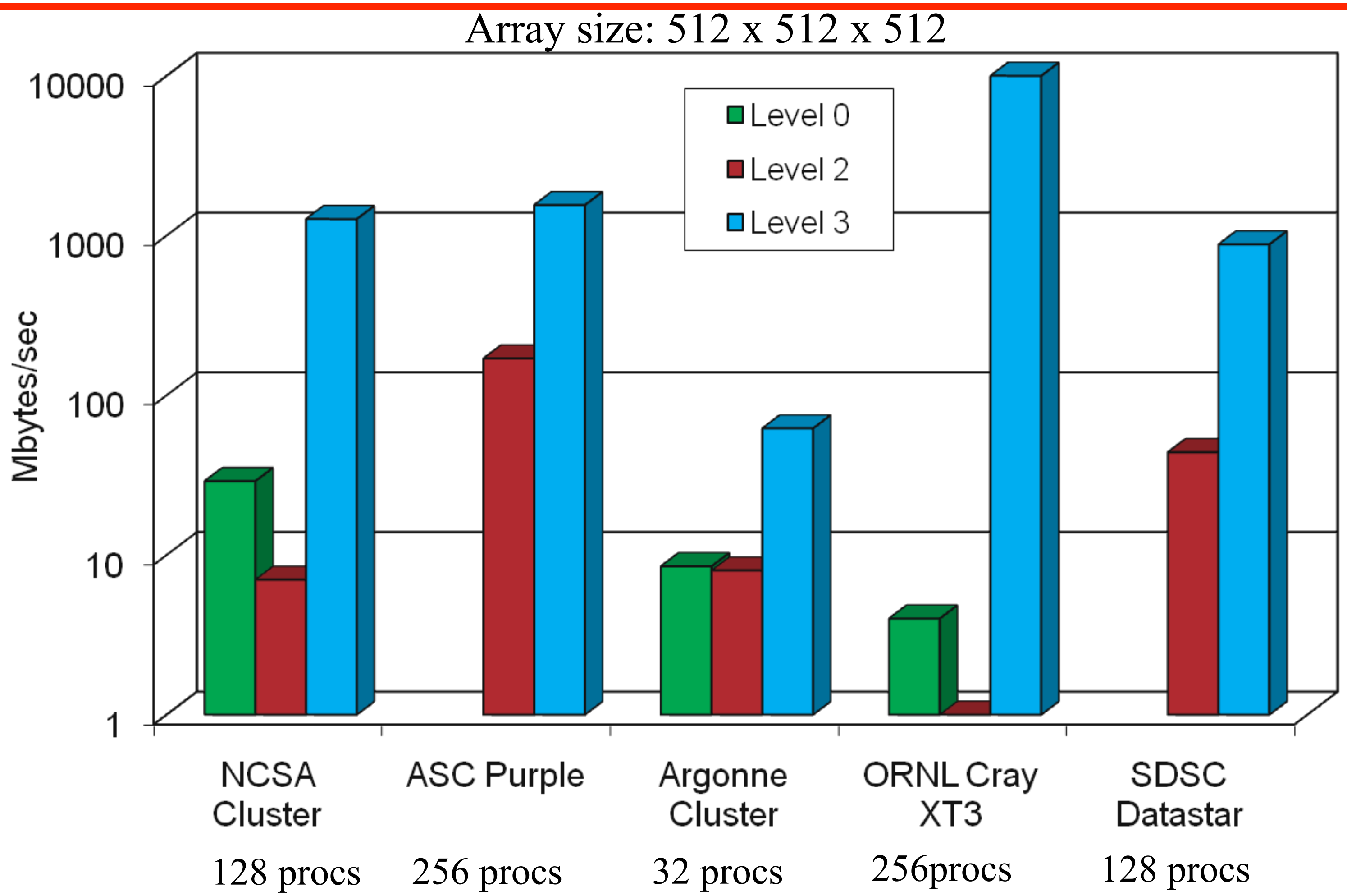
- Multiple operations must be pending at any time
  - ♦ Asynchronous I/O, communication, even computation
  - ♦ "split" computations and communication
- Complex systems require adaptive approaches
  - ♦ "Autotuning" for likely choices, runtime optimization
- Operations must be on aggregates
  - ♦ CPU: "vectors" (GPU gangs/workers/vectors)
  - ♦ I/O: Collective, parallel I/O
- Example: Parallel collective I/O for a distributed data structure
  - ♦ mesh distributed across all nodes

Four Levels of Collective I/O



PARALLEL@ILLINOIS

Distributed Array Access: Write Bandwidth



Note: Log Scale!

1GB data

Thanks to Weikuan Yu, Wei-keng Liao, Bill Loewe, and Anthony Chan for these results.

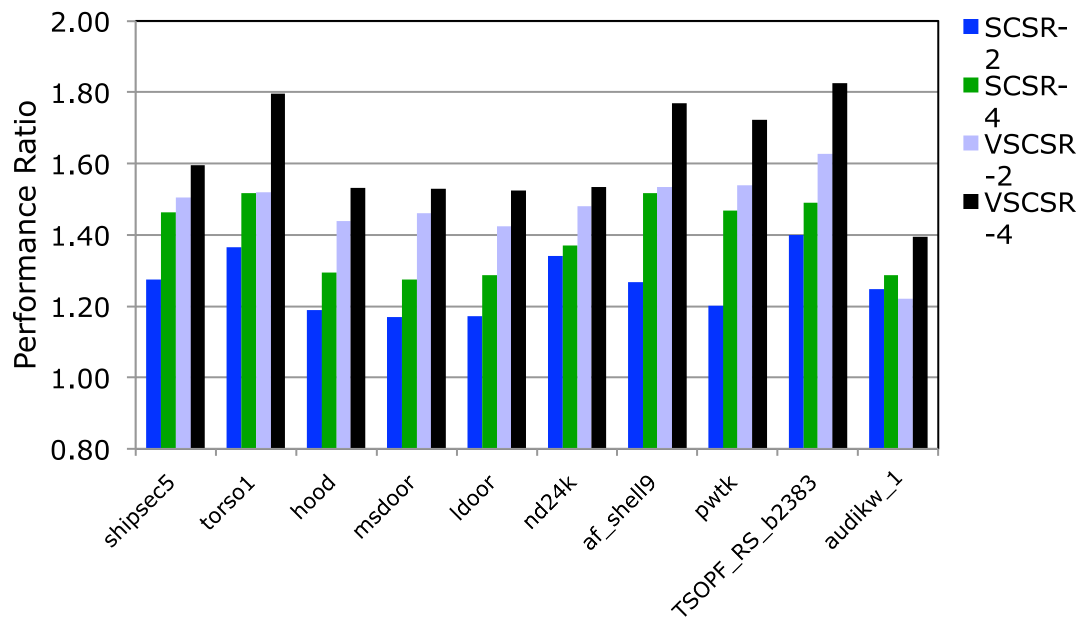
Better Algorithms and Data Structures

- Relying on compilers or other optimization tools (including autotuning) only offers the best performance with the given data structure and algorithm
  - That's a big constraint
- Processors include hardware to address performance challenges
  - "Vector" function units
  - Memory latency hiding/prefetch
  - Atomic update features for shared memory
  - Etc.

Sparse Matrix-Vector Multiply

Barriers to faster code

- "Standard" formats such as CSR do not meet requirements for prefetch or vectorization
- Modest changes to data structure enable both vectorization, prefetch, for 20-80% improvement on P7



Prefetch results in *Optimizing Sparse Data Structures for Matrix Vector Multiply* http://hpc.sagepub.com/content/25/1/115

PARALLEL@ILLINOIS

What Does This Mean For You?

- It is time to rethink data structures and algorithms to match the realities of memory architecture at all levels
  - Better match of algorithms to prefetch hardware is necessary to overcome memory performance barriers
- Similar issues come up with heterogeneous processing elements (someone needs to *design* for memory motion and concurrent and nonblocking data motion) and for file/data operations

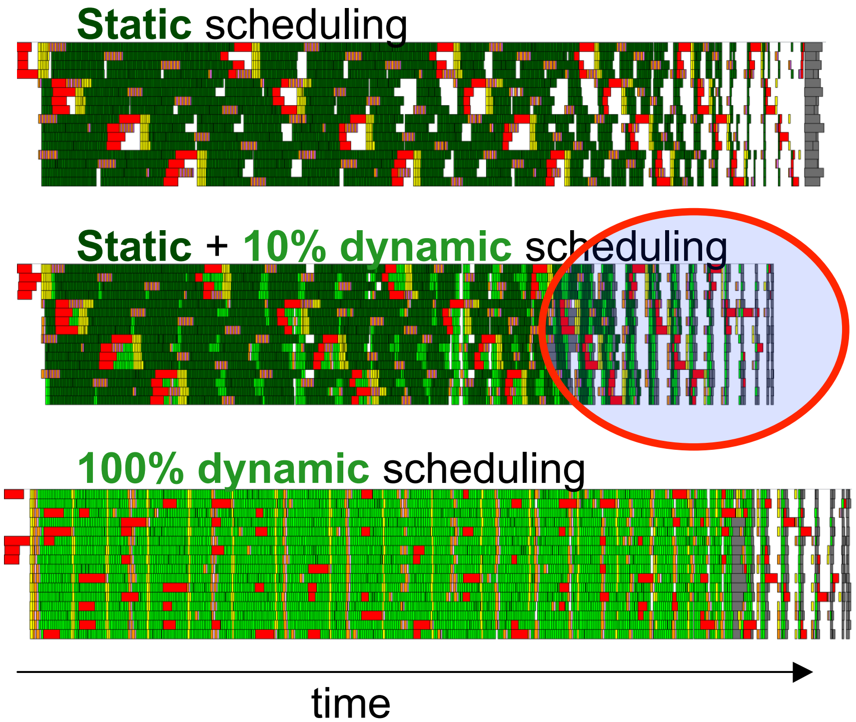
Processes and SMP nodes

- HPC users typically believe that their code "owns" all of the cores all of the time
  - The reality is that was never true, but they did have all of the cores the same fraction of time when there was one core /node
- We can use a simple performance model to check the assertion and then use measurements to identify the problem and suggest fixes.
- Based on this, we can tune a state-of-the-art LU factorization….

1867

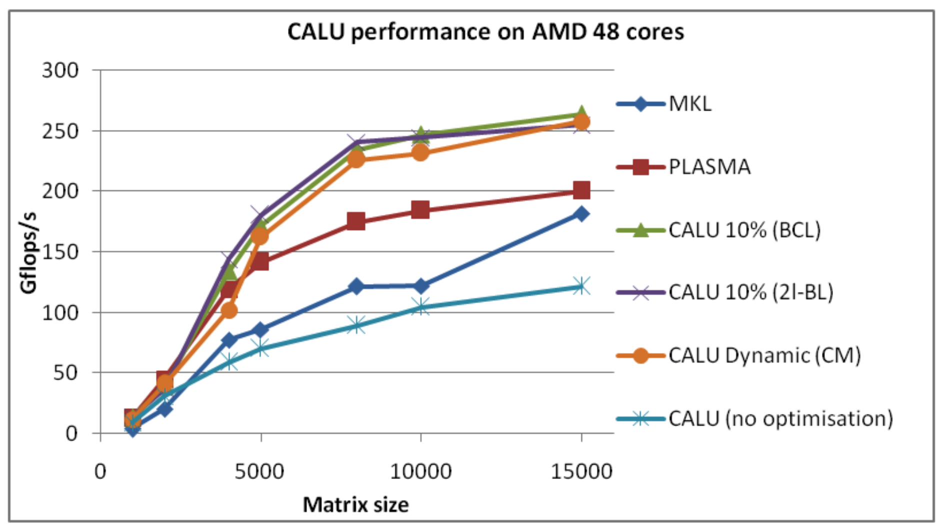
PARALLEL@ILLINOIS

Happy Medium Scheduling



Performance irregularities introduce load-imbalance.
Pure dynamic has significant overhead; pure static too much imbalance.
Solution: combined static and dynamic scheduling

Communication Avoiding LU factorization (CALU) algorithm, S. Donfack, L .Grigori, V. Kale, WG, IPDPS '12



Scary Consequence: Static data decompositions *will not work at scale.*

Corollary: programming models with static task models *will not work at scale*

Needs for Big Data and Extreme Scale Simulation

- Better use of existing resources
  - Performance-oriented programming
  - Dynamic management of resources at all levels
  - Embrace hybrid programming models (you have already if you use SSE/VSX/OpenMP/OpenAcc/…)
- Focus on results (end-to-end)
  - Adapt to available network bandwidth and latency
  - Exploit I/O capability (available space grew faster than processor performance!)
- Prepare for the future
  - Latency tolerant algorithms
  - Data-driven systems
  - Hybrid processor architectures
  - Fault tolerance

Thanks

- Torsten Hoefler
  - Performance modeling lead, Blue Waters; MPI datatype
- David Padua, Maria Garzaran, Saeed Maleki
  - Compiler vectorization
- Dahai Guo
  - Streamed format exploiting prefetch, vectorization, GPU
- Vivek Kale
  - SMP work partitioning
- Hormozd Gahvari
  - AMG application modeling
- Marc Snir and William Kramer
  - Performance model advocates
- Abhinav Bhatele
  - Process/node mapping
- Van Bui
  - Performance model-based evaluation of programming models
- Funding provided by:
  - Blue Waters project (State of Illinois and the University of Illinois)
  - Department of Energy, Office of Science
  - Sandia National Laboratories
  - National Science Foundation

SC12

Salt Lake City, Utah

November 10-16, 2012

SC13

Denver, CO

2013

sighpc

ACM Special Interest Group on High Performance Computing