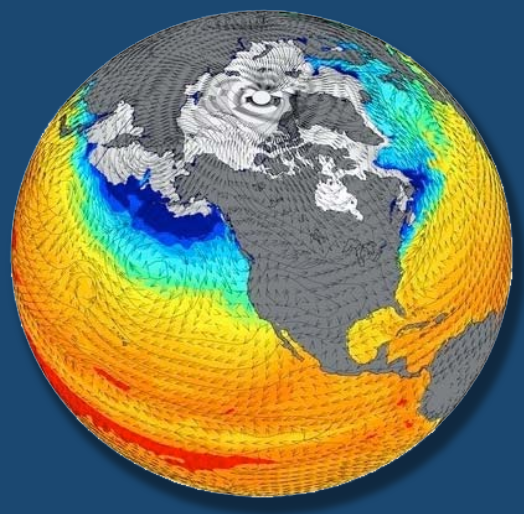
One Two Tree … Infinity:
Facts and Speculations on Exascale Software

Pete Beckman, Argonne National Laboratory

Director, Exascale Technology and Computing Institute (ETCi)

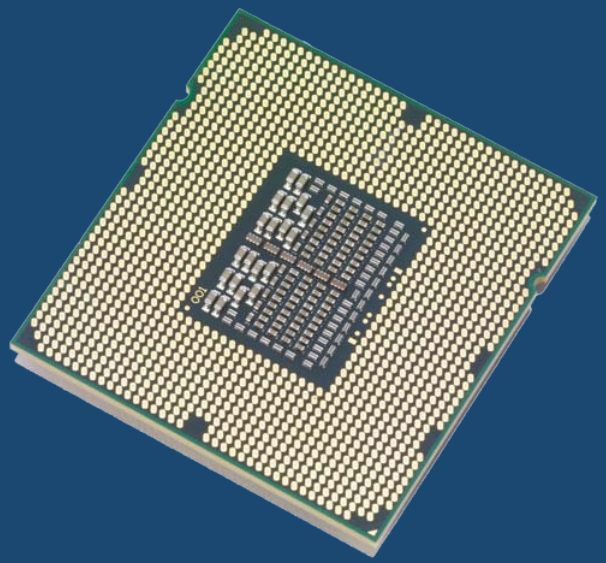
Jim Ahrens, Pavan Balaji, Pete Beckman, George Bosilca, Ron Brightwell,
Jason Budd, Shane Canon, Franck Cappello, Jonathan Carter,
Sunita Chandrasekaran, Barbara Chapman, Hank Childs, Bronis de Supinski,
Jim Demmel, Jack Dongarra, Sudip Dosanjh, Al Geist, Gary Grider,
Bill Gropp, Jeff Hammond, Paul Hargrove, Mike Heroux, Kamil Iskra,
Bill Kramer, Rob Latham, Rusty Lusk, Barney Maccabe, Al Malony,
Pat McCormick, John Mellor-Crummey, Ron Minnich, Terry Moore,
John Morrison, Jeff Nichols, Dan Quinlan, Terri Quinn, Rob Ross,
Martin Schultz, John Shalf, Sameer Shende, Galen Shipman, David Skinner,
Barry Smith, Marc Snir, Erich Strohmaier, Rick Stevens, Nathan Tallent,
Rajeev Thakur, Vinode Tipparaju, Jeff Vetter, Lee Ward, Andy White,
Kathy Yelick, Yili Zheng



```
/*
 * This is ugly, but reschedule_idle() is very timing-critical.
 * We are called with the runqueue spinlock held and we must
 * not claim the tasklist_lock.
 */
static FASTCALL(void reschedule_idle(struct task_struct * p));

static void reschedule_idle(struct task_struct * p)
{
#ifdef CONFIG_SMP
        int this_cpu = smp_processor_id();
        struct task_struct *tsk, *target_tsk;
        int cpu, best_cpu, i, max_prio;
        cycles_t oldest_idle;

        /*
         * shortcut if the woken up task's last CPU is
         * idle now.
         */
        best_cpu = p->processor;
        if (can_schedule(p, best_cpu)) {
                tsk = idle_task(best_cpu);
                if (cpu_curr(best_cpu) == tsk) {
                        int need_resched;
send_now_idle:
                        /*
                         * If need_resched == -1 t
                         * the IPI altogether, t
                         * actively watched by t
                         */
                        need_resched = tsk->need_re
                        tsk->need_resched = 1;
                        if ((best_cpu != this_cpu) &&
                                smp_send_reschedule(best_
                        return;
                }
```



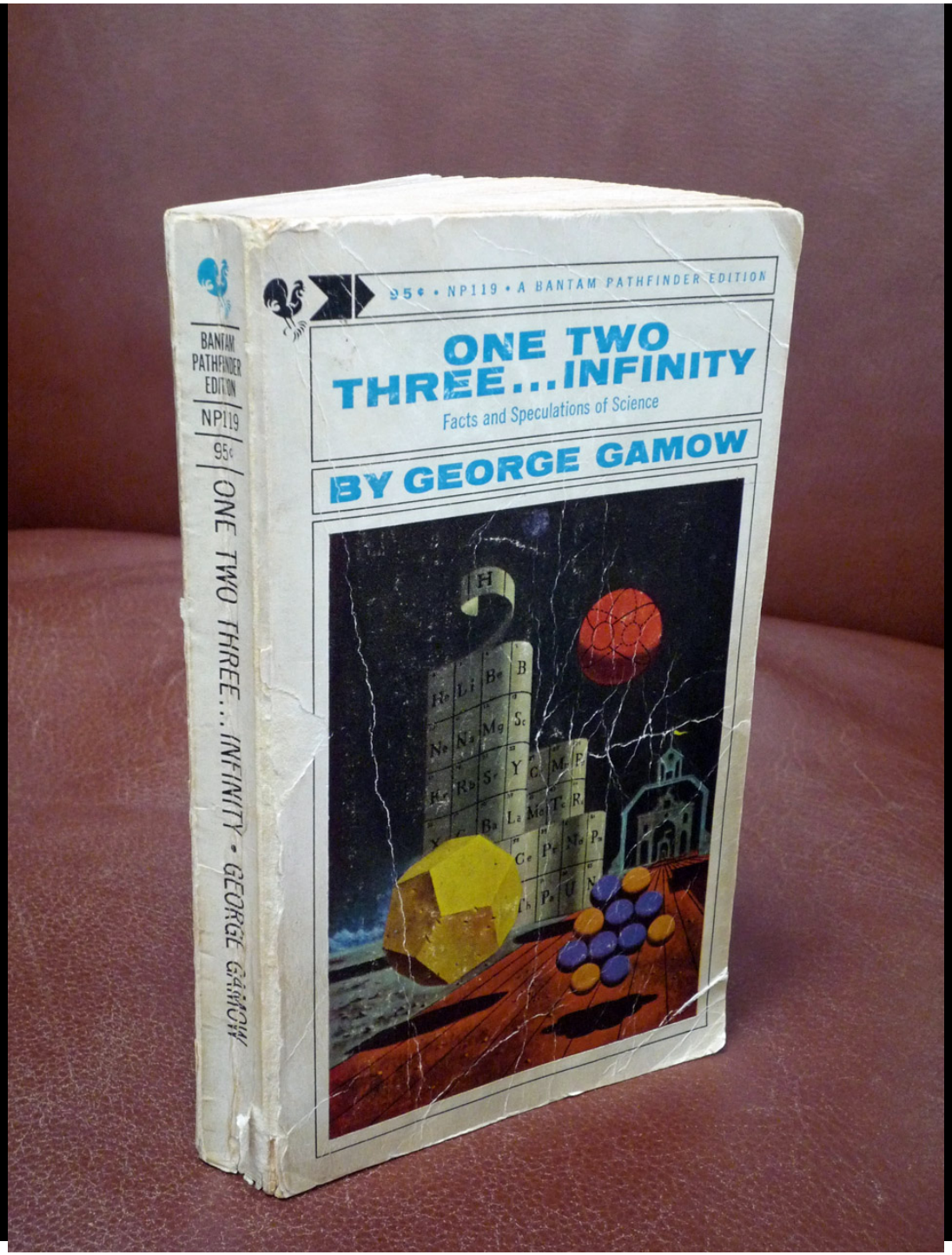

95¢ • NP119 • A BANTAM PATHFINDER EDITION
ONE TWO THREE...INFINITY
Facts and Speculations of Science
BY GEORGE GAMOW
BANTAM PATHFINDER EDITION
NP119
95¢
ONE TWO THREE...INFINITY • GEORGE GAMOW

FIGURE 2

Grand Vizier Sissa Ben Dahir, a skilled mathematician, asks his reward from King Shirham of India.

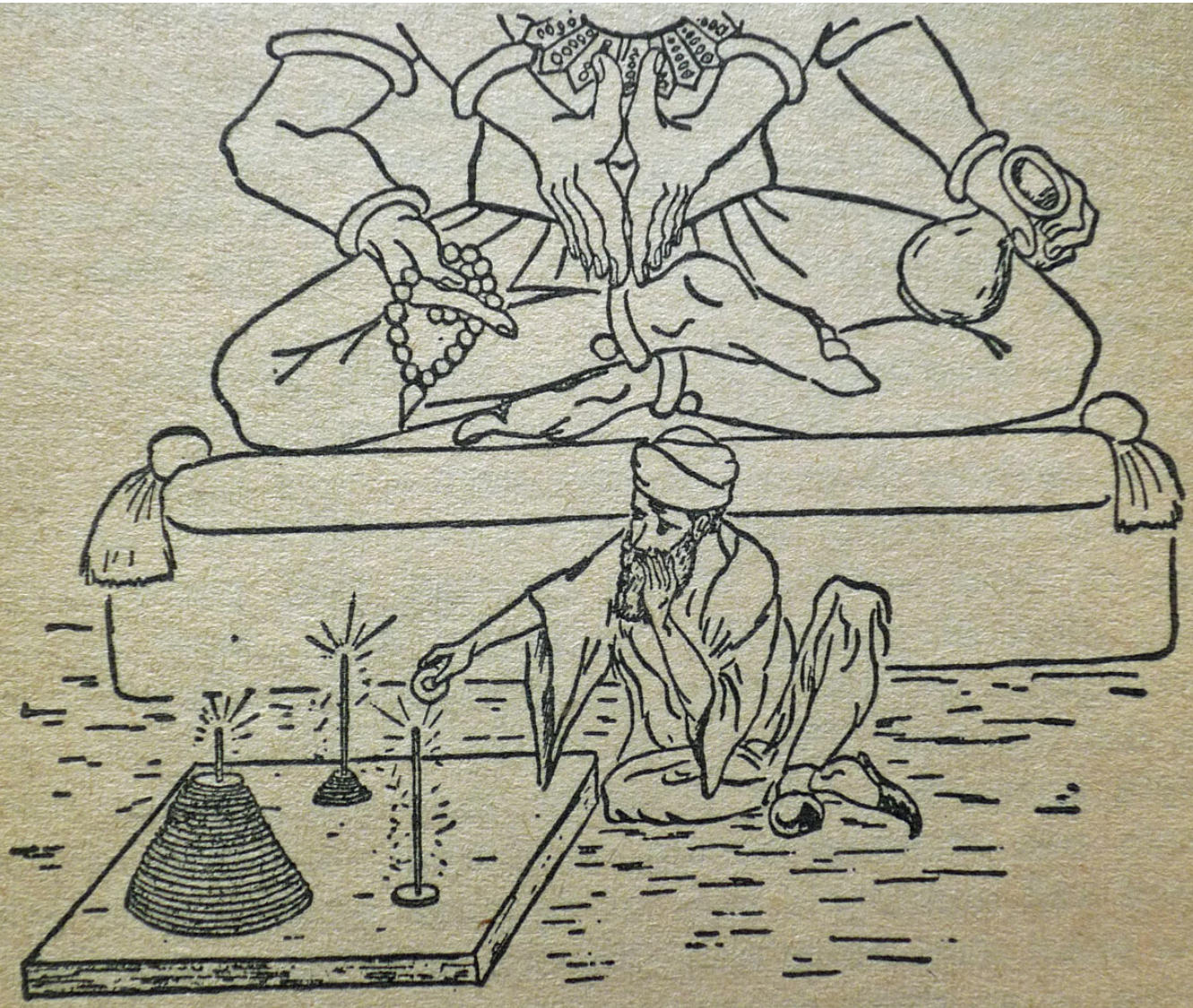
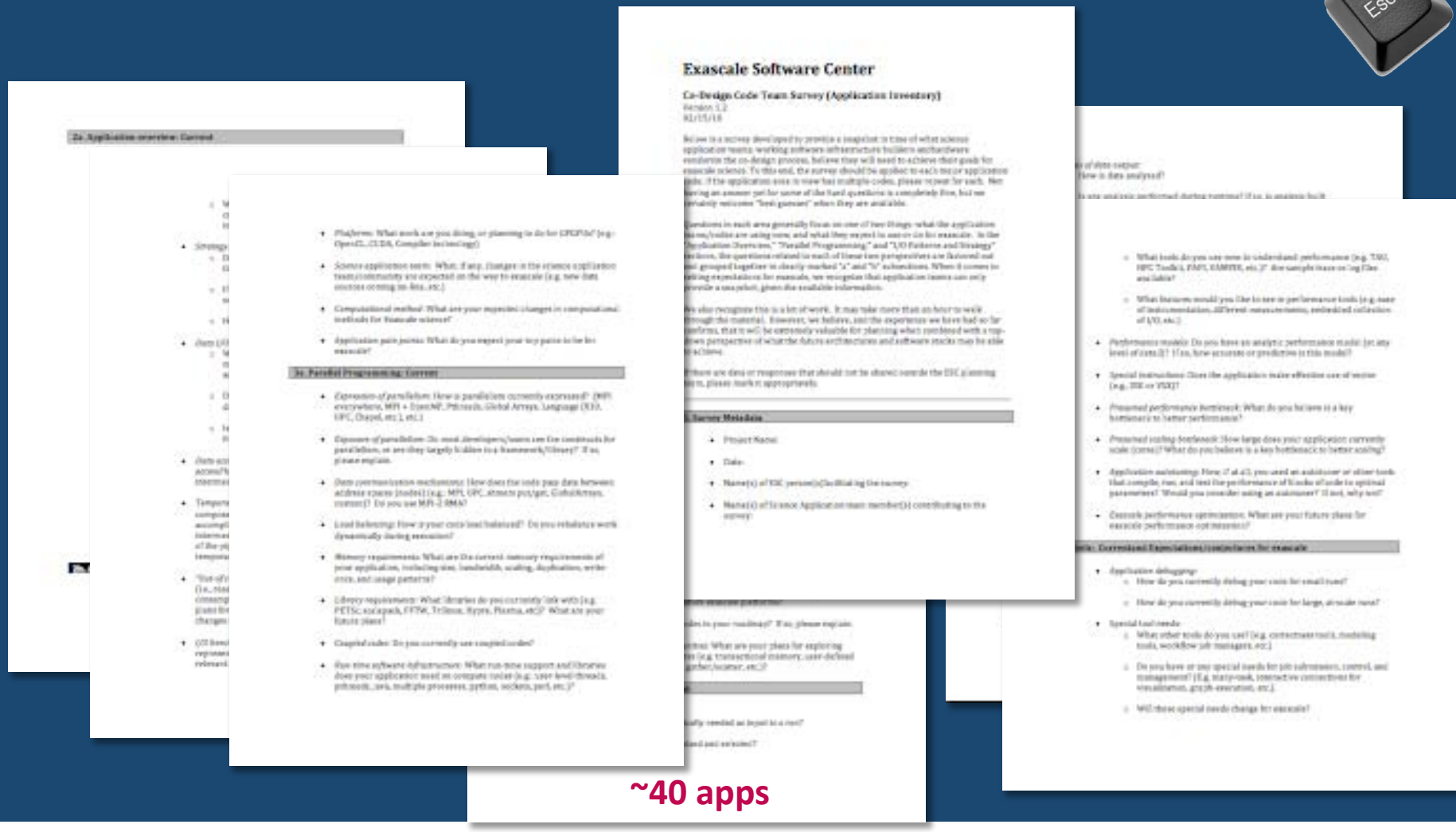


FIGURE 3

A priest working on the "End of the World" problem in front of a giant statue of Brahma. The number of golden discs is shown here smaller than 64 because it was difficult to draw so many.

What are your plans for exascale?



~40 apps

Excerpts

Do most developers/users see the constructs for parallelism […]

- "Most developers don't see the construct for the parallelism, and it is hidden in functions or library. In this way, physics developers don't have to worry about the parallelism, and physics packages are automatically parallelized. MPI is largely hidden inside wrappers right now, except where people have not followed that 'rule', for whatever reason. "

Will that be changing for exascale?

- "We cannot make this call yet."

How is your code load balanced?

- "Yes, it is dynamically load balanced, but not terribly well. […] we dynamically rebalance the workload. The balance is checked after every time step. "

"But our coarse grain concurrency is nearly exhausted"

(Bill Dally: A programming model that expresses all available parallelism and locality – hierarchical thread arrays and hierarchical storage)

Exascale Applications: A Few Selected Observations

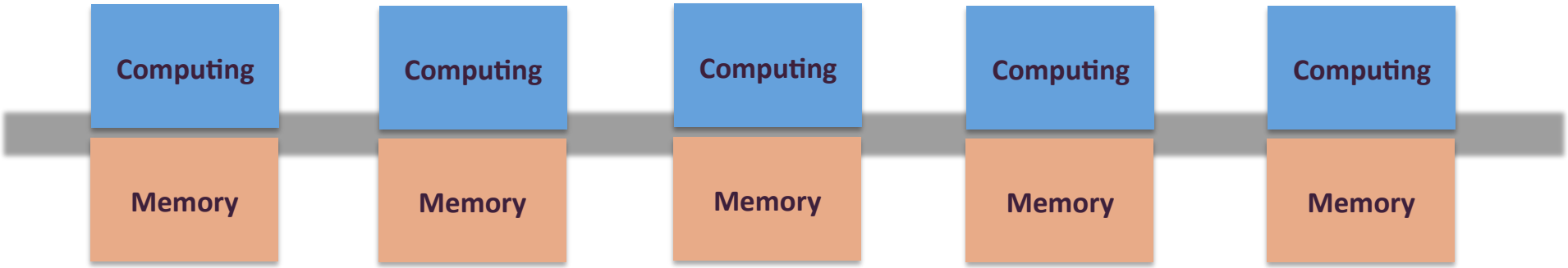
- All Codes:
  - Portability is critical
  - Large, mature community and code base
- Many codes:
  - Hide parallelism in framework
  - Have load balancing issues
- Some codes:
  - Use “advanced” toolchains:
    - Python, Ruby, DLLs, IDL, DSL code generation, autotuning, compiler transforms, etc
  - Remote invocation, real threads
- Shared worries:
  - How should I manage faults?
  - Bug or fault, can you tell me which?
  - Can I have reproducibility?
  - Given: MPI+X, solve for X

CESM	Global climate model (atmosphere, ocean, land, ice, etc.)
TGYRO	GYRO gyrokinetic solver and NEO neoclassical transport calculator
GTS	Global gyrokinetic PIC simulation
BOUT++	Simulation of tokamak edge plasma phenomena
M3D-C1	Simulation of global MHD instabilities in tokamaks
NIMROD	Model long-wavelength instabilities in fusion experiments
VORPAL	Simulation of RF-plasma coupling in fusion plasmas
NWChem	Multiple computational chemistry methods
MADNESS	Multiresolution Adaptive Numerical Environment for Scientific Simul
ddcMD	molecular dynamics
Ale3D	Arbitrary Lagrangian-Eulerian hydrodynamics and structural dynami
CPMD	Car-Parrinello molecular dynamics
Qbox	First principles molecular dynamics
Coop	Cooperative parallelism runtime system
LMC	Low Mach number combustion simulation
S3D	Compressible Navier-Stokes solver
GFMC	Green's function Monte Carlo nuclear structure calculation
Flash	Simulates compressible reactive flows in HEDP: supernova and cosm
FLAG	Multiphysics simulations
MCNP	Radiation transport of neutral and charged particles, mostly Monte C
PARTISN	Deterministic neutron/gamma transport solver
RAGE	Multiphysics simulations, Eulerian grid with AMR
Code 1	Multiphysics simulation using ALE techniques
Code 2	Radiation-hydrodynamics with thermal burn and laser ray tracing
Code 3	Multiphysics simulation with Lagrangian hydro, ALE remap, multiple
Code 4	Multiphysics ALE radiation-hydrodynamics on block-structured mesh
Code 5	Lattice QCD and technicolor
Code 6	Classical molecular dynamics
Code 7	Dislocation dynamics
Code 8	Library of high-performance preconditioners
Code 9	Data visualization and analysis tool
Code 10	Simulation of high-intensity laser-plasma interaction
Denovo	Reactor analysis and nuclear technology
Trident	Coupled reactor physics
Nek5000	Fluid mechanics, heat transfer
Diablo	Structural mechanics in response to temperature pressure, applied
UNIC	Neutron transport modeling in reactor cores
Sierra	Engineering analysis suite for thermal, fluid, solid, and structural m
LAMMPS	Classical molecular dynamics (large-scale atomic/molecular massive

Productivity == advanced tool chains
Understand having != Scaling
Productivity == advanced execution models

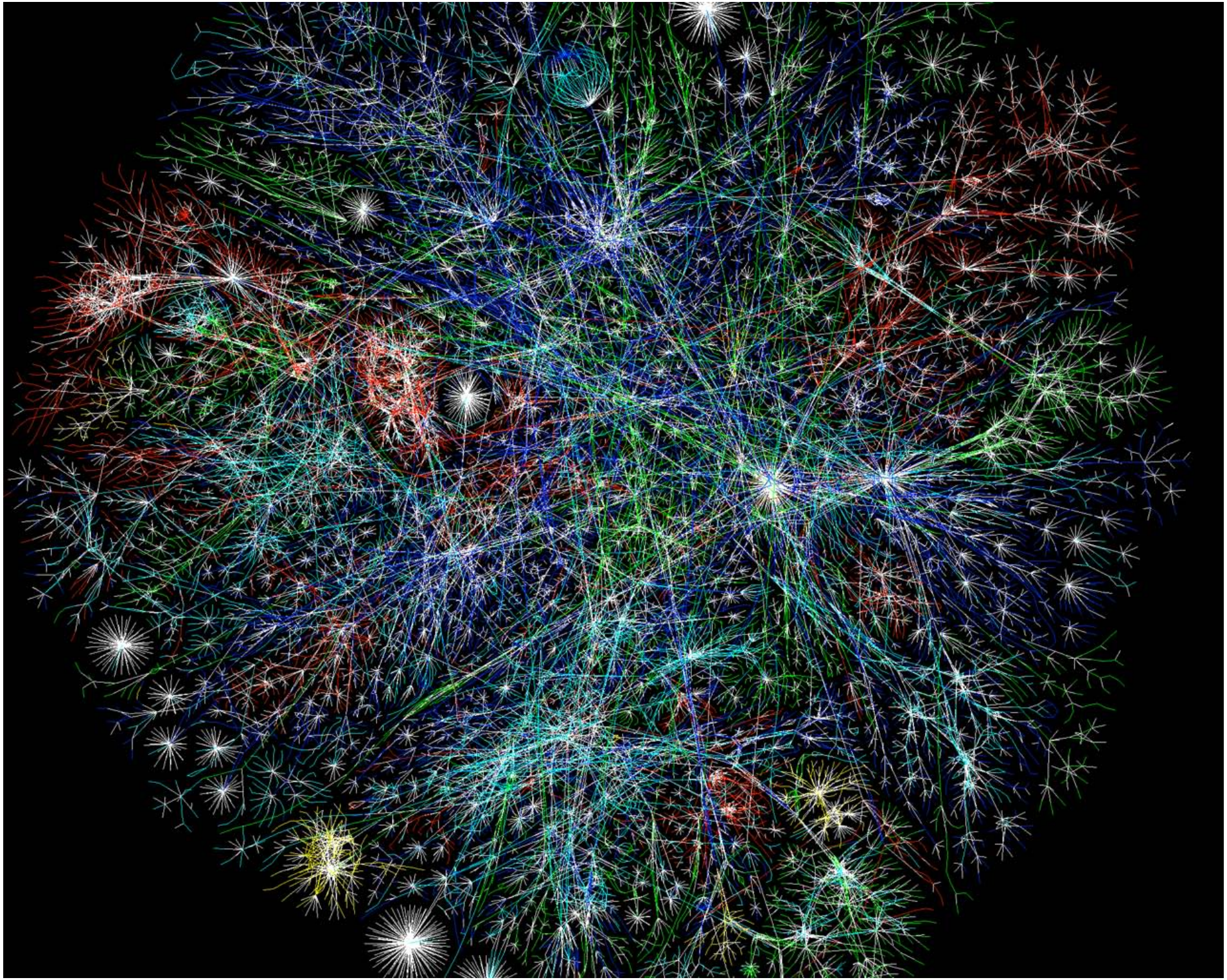
(unscalable is as unscalable does)

The Abstract Machine Model & Execution Model



- A computer language is not a programming model
  - "C++ is not scalable to exascale"
- A communication library is not a programming model
  - "MPI won't scale to exascale"
- A unique application inventory response…
  - Should we be talking "Execution Model"?
- What is a programming model?

Thought: real issue is mapping science problem to execution model and run-time system



Exascale Architectures: A Few Selected Observations

- Power will be (even more) a managed resource
  - More transistors than can be turned on
- Memory and computing packaged together
- Massive levels of in-package parallelism
- Heterogeneous multicore
- Complex fault and power behavior
- NVRAM near computing

- (selected) Conclusions
  - More hierarchy, dynamic behavior
  - More investment in compositional run-time systems
  - Plus… new I/O models, fault communication, adv. language features, etc

Exascale Application Frameworks (Co-design Centers, etc)

(selected) To Do List:

- Embrace dynamism -- convert flat fork-join to hierarchy: graphs, trees, load balancing
- Understand fault, reproducibility

Compute Node

- 16 PowerPC processing cores per node, 4w MT, 1600MHz
- SIMD floating point unit (8 flop/clock)
- Crossbar switch with optimistic routing protocol
- Speculative multithreading and transactional memory support
- 8 or 16GB DDR3 SDRAM memory per node
- Voltage scaled design point
- 17th processor core used for system functions.

System Software

Linear Algebra & Math Software
- Dense / Sparse / LA
- Autotuning
- Comm. Reducing Algs
- Energy Eff. Methods
- Resilient Methods

(PETSc, Trilinos, SuperLU, Hypre, MAGMA, PLASMA)

Advanced Programming Language Features
- Hybrid parallelism
- Threads/ tasks
- Collective data and comm

(Berkeley UPC, Houston OpenMP, Rice CAF)

Execution Model Support
- Programming Model Composability
- Fault tolerance
- Active messages / 1-sided comm

(MPICH, Plum, CIFTS)

I/O Libraries & Analysis Support
- Efficient checkpoint libraries
- End-to-end integrity
- Improved data formats
- In-situ analysis support

(SCR, HDF5, PLFS, VTK)

Storage & Metadata
- I/O Forwarding
- Autonomics, topology awareness
- Tiered storage, large file cnts,

(Giga+, IOFSL)

Tools
- Correctness introspe
- Problem-focused meas ent
- Measurement data mgmt
- Execution control

(PAPI, TAU, HPCtoolkit, OpenSpeedshop, mpiP, Marmot, MUST, memcheck, etc)

Operating and Run-time System
- Asymmetric kernel for heterogeneous CPU
- Power management
- Lightweight threads & scheduling ctrl
- Partitioning cores
- System simulation

(Linux, SST, Qthreads, IOFSL)

(selected) To Do List:

- Parallel composition run-time support
- Async multithreadedness
- Lightweight concurrency support
- Load balancing?
- Fault coordination

Examples

Linear Algebra

Tools

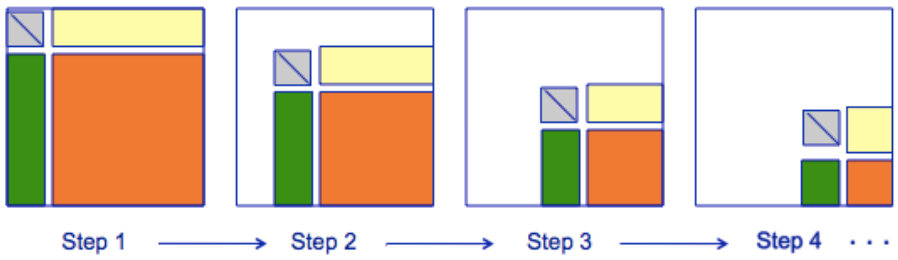
Message-driven execution

Language

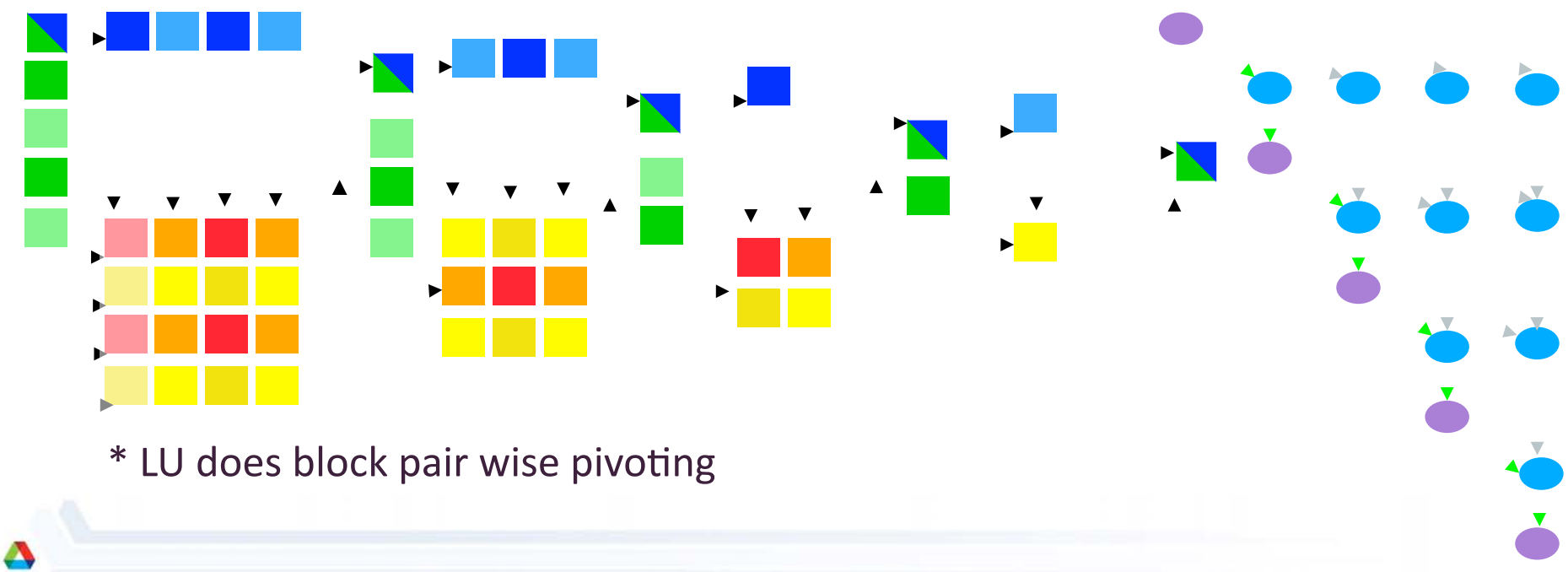
Worker / Task

Courtesy Jack Dongarra:

Parallel Tasks in LU/LL^T/QR



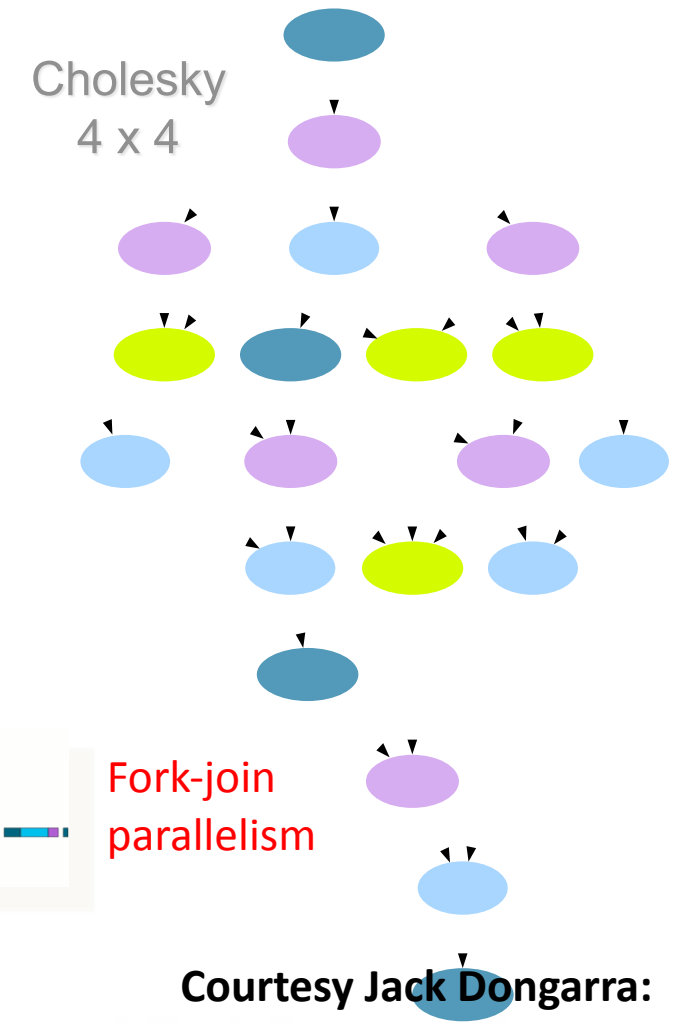
- Break into smaller tasks and remove dependencies



* LU does block pair wise pivoting

PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

- Objectives
  - High utilization of each core
  - Scaling to large number of cores
  - Shared or distributed memory
- Methodology
  - Dynamic DAG scheduling
  - Explicit parallelism
  - Implicit communication
  - Fine granularity / block data layout
- Arbitrary DAG with dynamic scheduling

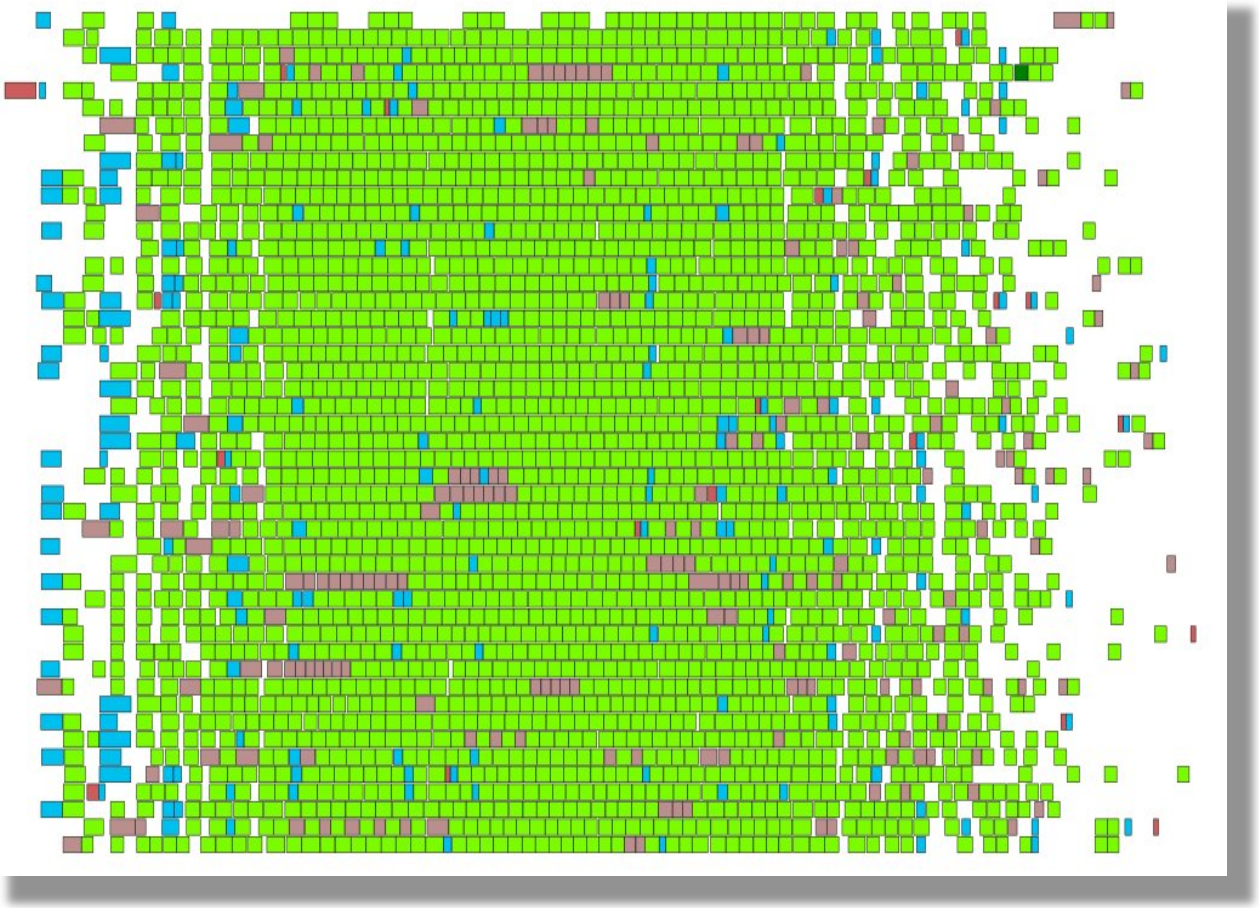


Courtesy Jack Dongarra:

Courtesy Jack Dongarra:

Synchronization Reducing Algorithms

- Regular trace
- Factorization steps pipelined
- Stalling only due to natural load imbalance
- Reduce ideal time
- Dynamic
- Out of order execution
- Fine grain tasks
- Independent block operations

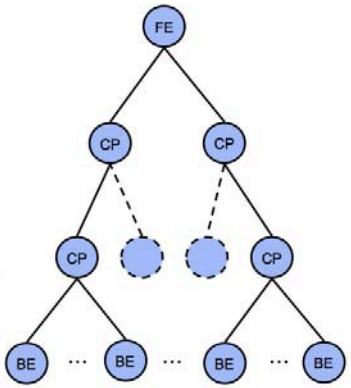


Tile LU factorization; Matrix size 4000x4000, Tile size 200
8-socket, 6-core (48 cores total) AMD Istanbul 2.8 GHz

Courtesy: Barton Miller

Tree-Based Overlay Network (TBŌN) Model

- Tree of communication processes used for:
  - Scalable multicast
  - Scalable gather
  - Scalable data reduction
- Tree topology can be matched to required communication and computation
- Communication processes may be co-located with back-ends or run on separate hosts



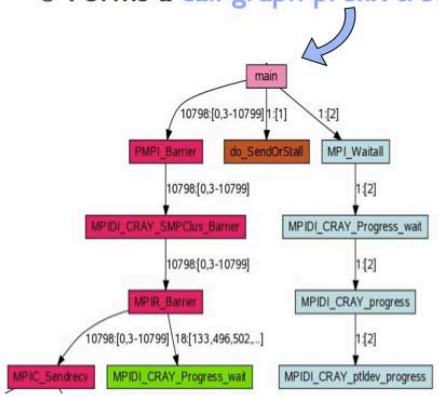
MRNet: An Easy-to-Use TBŌN

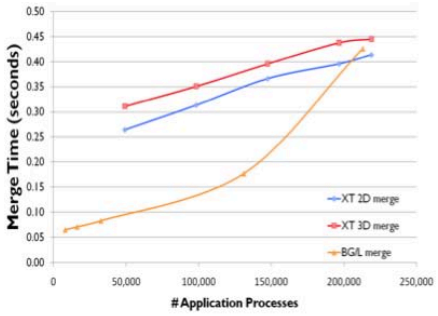
- Key Abstractions
  - Stream: logical data flow with custom filtering
    - Provides multicast/reduction over a set of back-ends
    - Tools can use many simultaneous streams
  - Packet: encapsulated data
    - Collection of singleton or array data types
- Efficiency
  - Zero-copy paths
  - Binary data representation
- Fault-Tolerance
  - TBŌN self-heals when a communication process fails
  - Streams can survive failures if tool provides fault-tolerant filters
- http://www.paradyn.org/mrnet

© Barton P. Miller 2011

STAT: An MRNet-based Tool

- Stack-Trace Analysis Tool
  - Collects stack traces from parallel application processes
  - Forms a call-graph prefix tree that shows equivalent behavior





© Barton P. Miller 2011

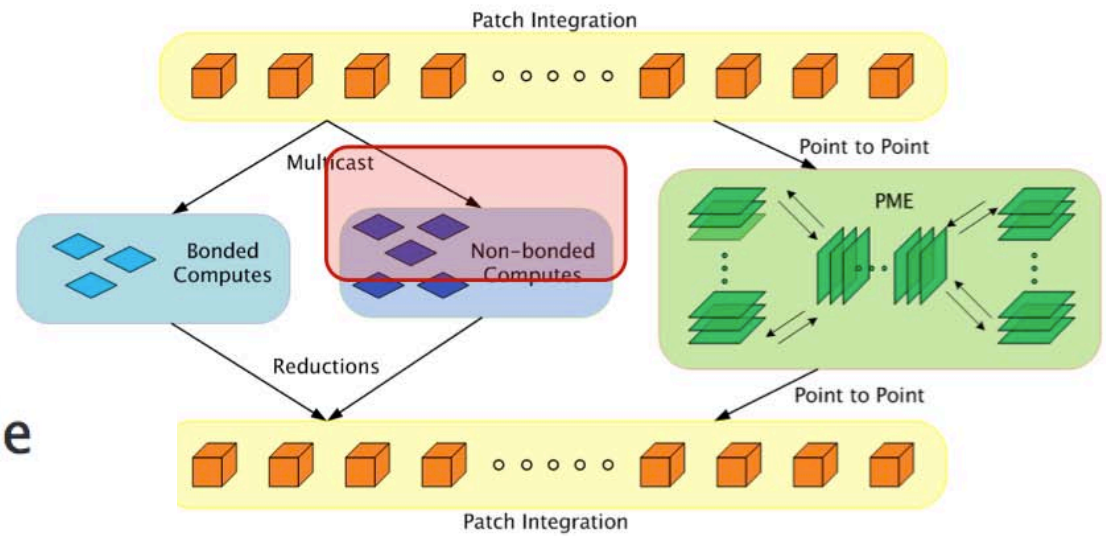
I/O Forwarding (IOFSL) and MRNET can share components and code, and are discussing ways to factor and share data movement and control

Charm++
(the run-time and execution model for NAMD)

Courtesy: Laxmikant Kale

Parallelization Using Charm++

The computation is decomposed into "natural" objects of the application, which are assigned to processors by Charm++ RTS



- Charm++/AMPI style "virtual processors"
  - Decompose into natural objects of the application
  - Let the runtime map them to processors
  - Decouple decomposition from load balancing

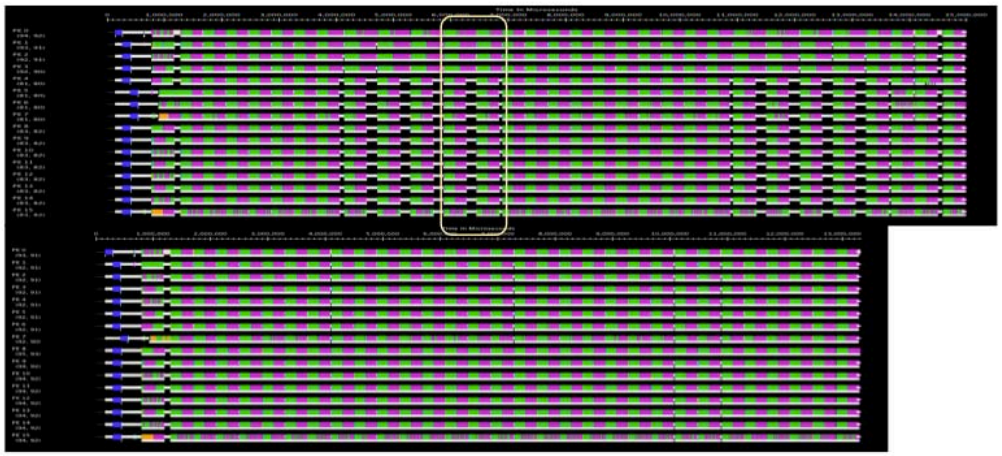
Courtesy: Laxmikant Kale

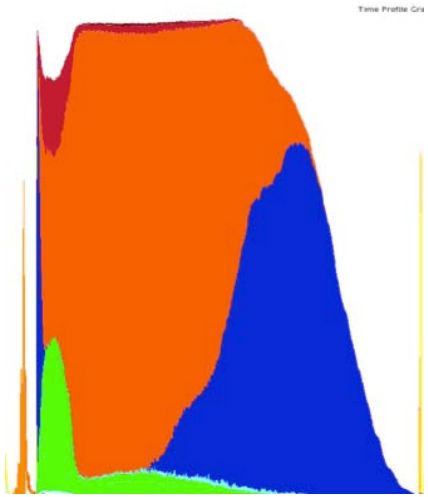
Powerful constructs for load and fault management

Challenge: Compositionality

- It is important to support compositionality
  - For multi-module, multi-physics, multi-paradigm applications..
- What I mean by parallel composition
  - B || C where B and C are independently developed modules
  - B is parallel module by itself, and so is C
  - Programmers who wrote B were unaware of C

Benefits of Temperature Aware LB



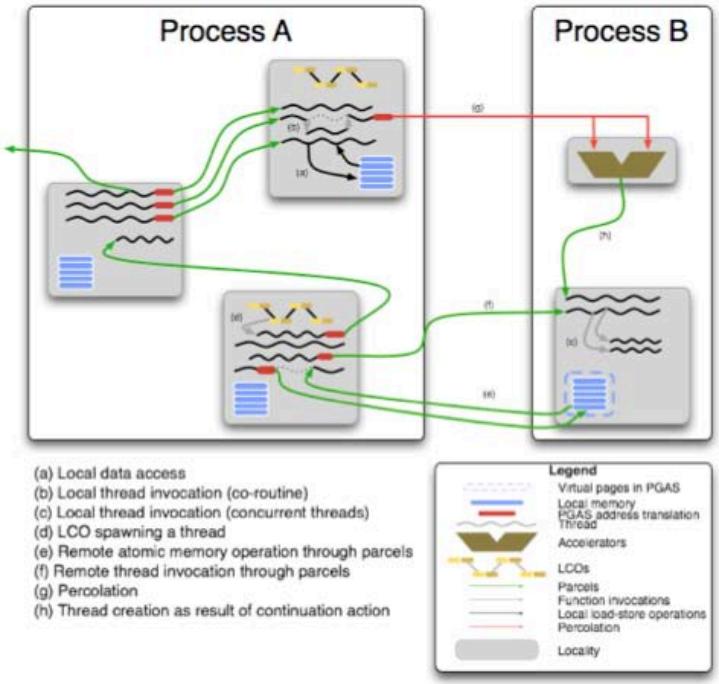


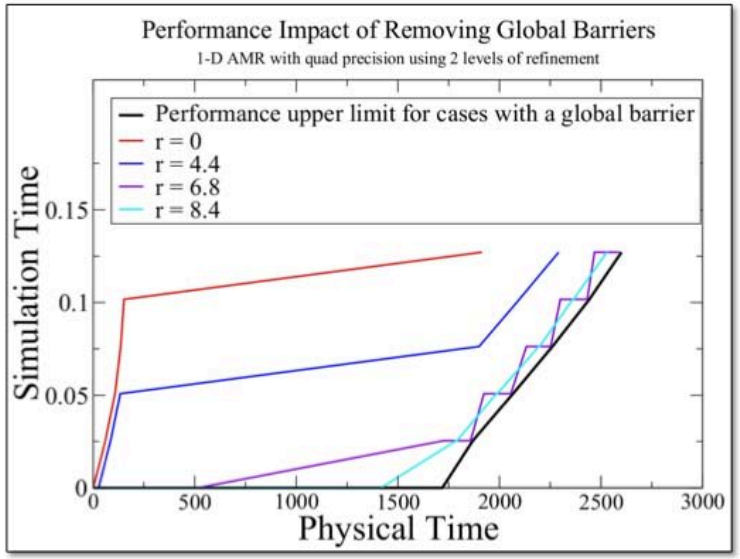
Courtesy: Thomas Sterling

ParalleX Programming Language

Application: Adaptive Mesh Refinement (AMR) for Astrophysics simulations

PX Model Components





- ParalleX based AMR removes all global computation barriers, including the timestep barrier (so not all points have to reach the same timestep in order to proceed computing)

Green's Function Monte Carlo - A Complex Application

Courtesy: Rusty Lusk

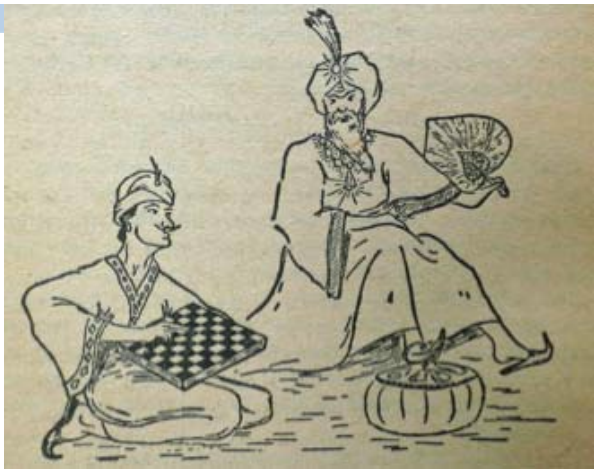
- Green's Function Monte Carlo -- the "gold standard" for *ab initio* calculations in nuclear physics at Argonne (Steve Pieper, PHY)
- A non-trivial master/worker algorithm, with assorted work types and priorities; multiple processes create work dynamically; large work units
- Had scaled to 2000 processors on BG/L a little over four years ago, then hit scalability wall.
- Need to get to 10's of thousands of processors at least, in order to carry out calculations on ^{12}C, an explicit goal of the UNEDF SciDAC project.
- The algorithm threatened to become even more complex, with more types and dependencies among work units, together with smaller work units
- Wanted to maintain master/worker structure of physics code
- This situation brought forth ADLB
- Achieving scalability has been a multi-step process
  - balancing processing
  - balancing memory
  - balancing communication

1	2				9			7
		3				6	1	
				7		8		
					5	3		
7		9	1		8	2		6
		5	6					
		1		9				
	6	7				1		
2			5				3	8

API for a Simple Programming Model

- Basic calls
  - ADLB_Init(num_servers, am_server, app_comm)
  - ADLB_Server()
  - ADLB_Put(type, priority, len, buf, target_rank, answer_dest)
  - ADLB_Reserve(req_types, handle, len, type, prio, answer_dest)
  - ADLB_Ireserve(…)
  - ADLB_Get_Reserved(handle, buffer)
  - ADLB_Set_Done()
  - ADLB_Finalize()
- A few others, for tuning and debugging
  - ADLB_{Begin,End}_Batch_Put()
  - Getting performance statistics with ADLB_Get_info(key)

Summary



- To reach exascale, applications must embrace advanced execution models
  - Over decompose, generating parallelism
  - Hierarchy! Tasks, trees, graphs, load balancing, remote invocation
  - We don't have many common reusable patterns…
    - That's the real "X"
  - No whiners
- Exascale System Software:
  - Support composition, advanced run-time, threads, tasks, etc
- Can exascale be reached without trees and tasks?
  - Other benefits:
    - Fault resistant
    - Node performance tolerant
    - Latency hiding