Application/Algorithm Requirements for Interconnects

Michael A. Heroux
Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Projects

- **Trilinos**:
  - Large collection of interoperable software libraries.
  - Meshing, discretization, load balancing, solvers, parallel data structures.
  - 8.0 Release 8/31/2007. 2200 downloads. 7000 since Mar ‘05. 5000 users.
  - Growing external collaborations: ORNL, LBL, INL, Boeing, XOM.
  - Trilinos 9.0: Fuller vertical SW stack, fuller support for Windows, Mac, more customers.

- **TOPS-2**:
  - DOE Office of Science SciDAC-2 Project.
  - Bringing Apps to Petascale via libraries.

- **Tramonto**:
  - Fluid Density Functional Theories code.
  - Nano-structured fluids, complex fluid structures, e.g., lipid-bilayers.
  - Tramonto 2.1: First public Release March 2007. 120 downloads.

- **Mantevo**:
  - Mantevo: Five microapps (phdMesh, HPCCG, pHPCCG, Beam, Prolego) + framework.
  - HPCCG: Publicly available. Part of Sequoia benchmark.
    - “Closest thing to an unstructured FEM/FVM code in 500 semi-colons or fewer.”
    - Ports to nVidia, Clovertown, Sun 8x8 core/threads, RedStorm, Sequoia RFP, …
    - Rewritten in BEC, Qthreads, OpenMP.
    - 25K core runs on Redstorm.
  - pHPPCG: Parametrized HPCCG - arbitrary int/float types, data structure base class.
  - phdMesh part of Trilinos…Beam exercises vertical stack in Trilinos…Prolego basic research.

Sandia National Laboratories

About MPI

- MPI will be the primary inter-node programming model.
- Very few people program in MPI: Abstractions.
- Right ingredients:
  - Portable, ubiquitous.
  - Forced alignment of work/data ownership and transfer.
- Matches architectures:
  - Interconnects of best commercial node parts.
- New languages:
  - Big fan of Co-Array Fortran (Have been for 15 years: F--).
  - Chapel looks good.
  - But tough uphill climb.

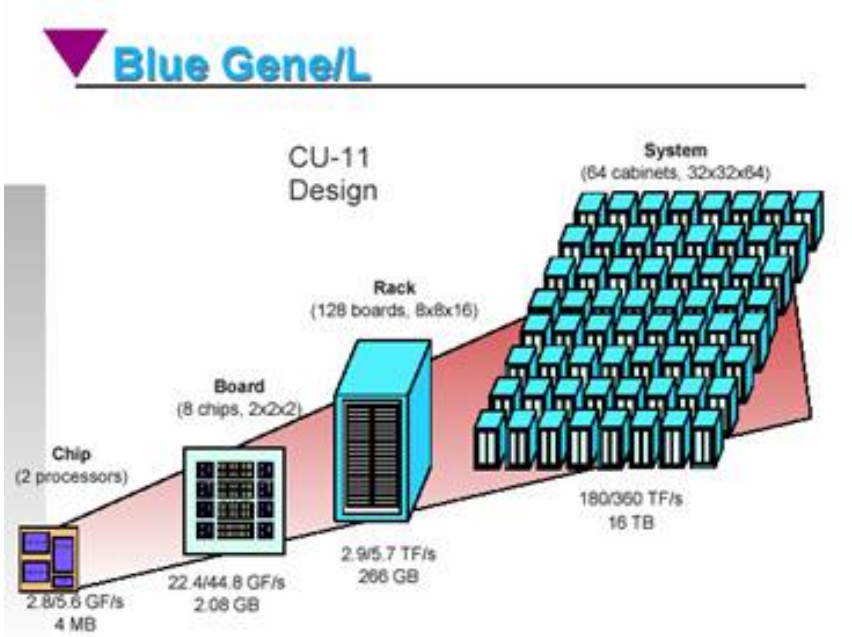
Two Views of the System Network

App Developer View

MPI Process

- Single core node.
- All other processes equi-distant.
- Simultaneous communication to many processes.

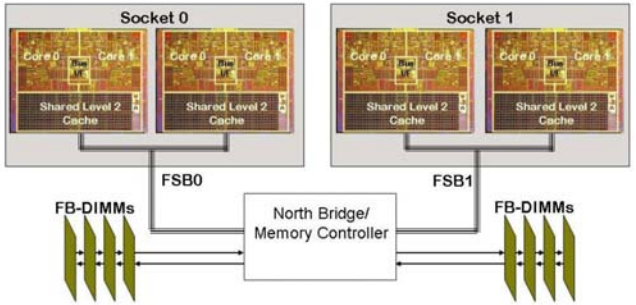
Reality



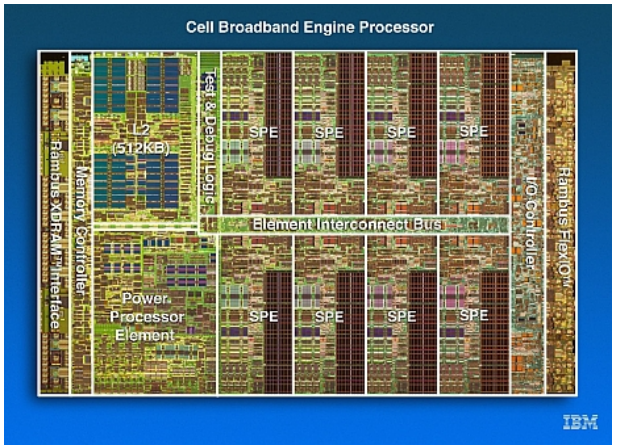
- Goal: Give app developers illusion they want.
- Problem: Harder and harder to do.
- Current focus: How to program the node?

Node Classification

- Homogeneous multicore:
  - SMP on a chip.
  - NUMA nodes.
  - Varying memory architectures.



- Heterogeneous multicore:
  - Serial/Controller processor(s).
  - Team of identical, simpler compute processors.
  - Varying memory architectures.



Sandia National Laboratories

Why Homogeneous vs. Heterogeneous?

- Homogeneous:
  - Out-of-the-box: Can attempt single-level MPI-only.
  - m nodes, n cores per node: p = m*n
  - mpirun -np p …
- Heterogeneous:
  - Must think of compute cores as “co-processors”.
  - mpirun -np m …
  - Something else on the node.
- Future:
  - Boundary may get fuzzy.
  - Heterogenous techniques can work well on homogeneous nodes.

Programming Models for Scalable Homogeneous Multicore (beyond single-level MPI-only)

Single Core Performance: Still improving for some codes

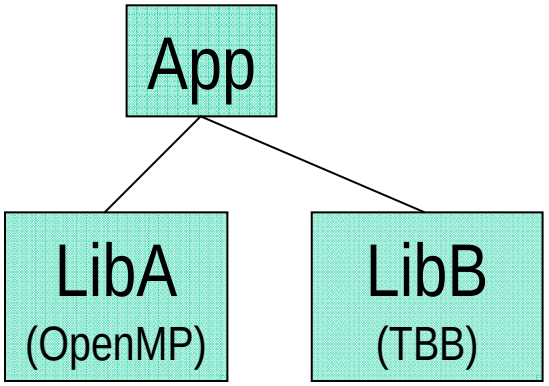
- HPCCG microapp.
- Clock speeds stable: ~ 2GHz.
- FP-friendly computations stalled.
- Memory-intensive computations still improving.

Year	Processor	Clock (GHz)	Cores per socket	MFLOPS /sec
2003	AMD Athlon	1.9	1	178
2004	AMD Opteron	1.6	1	282
2005	Intel Pentium M	2.1	1	310
2006	AMD Opteron	2.2	2	359
2007	Intel Woodcrest	1.9	4	401
2007	AMD Opteron	2.1	4	476
2007	Intel Core Duo	2.3	2	508

Sandia National Laboratories

Threading under MPI

- Default approach: Successful in many applications.
- Concerns:
  - Opaqueness of work/data pair assignment.
    - Lack of granularity control.
  - Collisions: Multiple thread models.
    - Performance issue, not correctness.



- Bright spot: Intel Thread Building Blocks (TBB).
  - Iterator (C++ language feature) model.
  - Opaque or transparent: User choice.

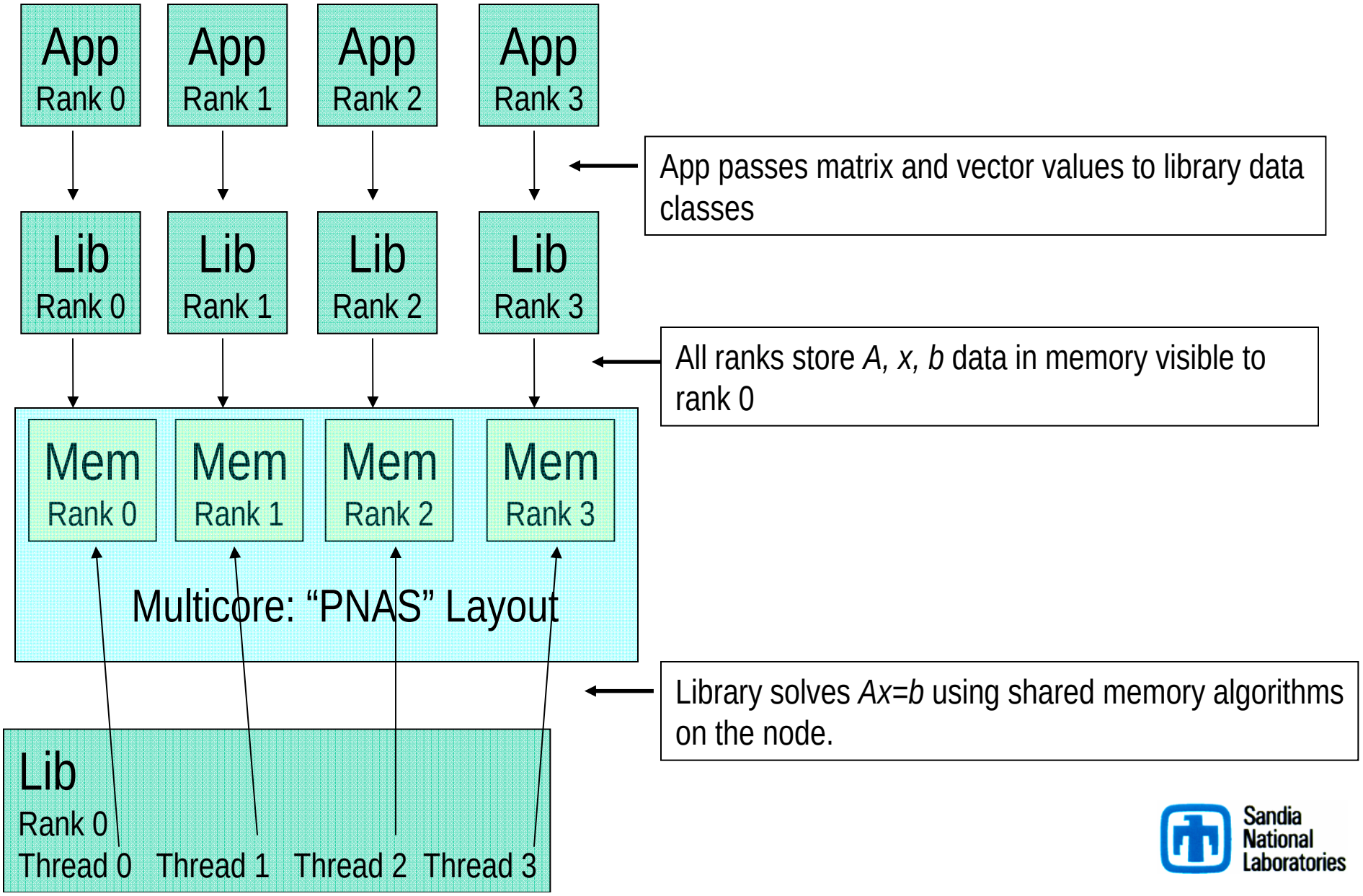
MPI Under MPI

- Scalable multicores:
  - Two different MPI architectures.
  - Machines within a machine.
- Exploited in single-level MPI:
  - Short-circuited messages.
  - Reduce network B/W.
  - Missing some potential.
- Nested algorithms.
- Already possible.
- Real attraction: No new node programming model.
- Can even implement shared memory algorithms (with some enhancements to MPI).

"Ping-pong" test	**Latency (microsec)**	**Bandwidth (MB/sec)**
Inter-node machine	0.71	1082
Intra-node machine	47.5	114

Sandia National Laboratories

MPI-Only + MPI/Threading: *Ax=b*



Sandia National Laboratories

Heterogeneous Multicore Issues

Excited about multimedia processors

- Inclusion of native double precision.
- Large consumer market.
- Qualitative performance improvement over standard microprocessors…
- If your computation matches the architecture.
- Many of our computations do match well.
- But a long road ahead…

Sandia National Laboratories

APIs for Heterogeneous Nodes (A Mess)

Processor	API
NVIDIA	CUDA
AMD/ATI	Brook+
STI Cell	ALF
Intel Larrabee	Ct
Most/All?	Sequoia
Most	RapidMind (Proprietary)
Apple/All	OpenCL

Commonality: Fine-grain functional programming.

Our Response: A Library Node Abstraction Layer

Sandia National Laboratories

Going Forward: Changing the Atomic Unit

- Now:
  Single-level MPI-only OK for many apps.
- Future:
  Hiding network heterogeneity beneath single MPI level too hard.
- Philosophical approach:
  Node becomes the new atomic unit.
- Key Requirement:
  Portable standard node API.
- Hard work:
  Changes are ubiquitous (unlike MPI).

Some Algorithm Trends

- Ensembles:
  - Increasing feasibility and importance.
  - UQ, QMU, stability analyses.
  - Tend to increase computation:communication ratio.
- Data-driven algorithms:
  - SPMD unfriendly.
  - Multithreading friendly.

Summary

- **Exciting times:**
  For architecture and software design.
- **Keep the illusion alive:**
  Flat, uniform, single core per node.
- **Multimedia processors:**
  Right mix for next qualitative performance improvement?
- **Possible scenario for some apps/libs**:
  - Heterogeneous API superior on homogeneous nodes.
  - Go directly from single-level MPI-only to MPI+heterogenous node?
- **A common, standard API for multicore:**
  Most critical need.