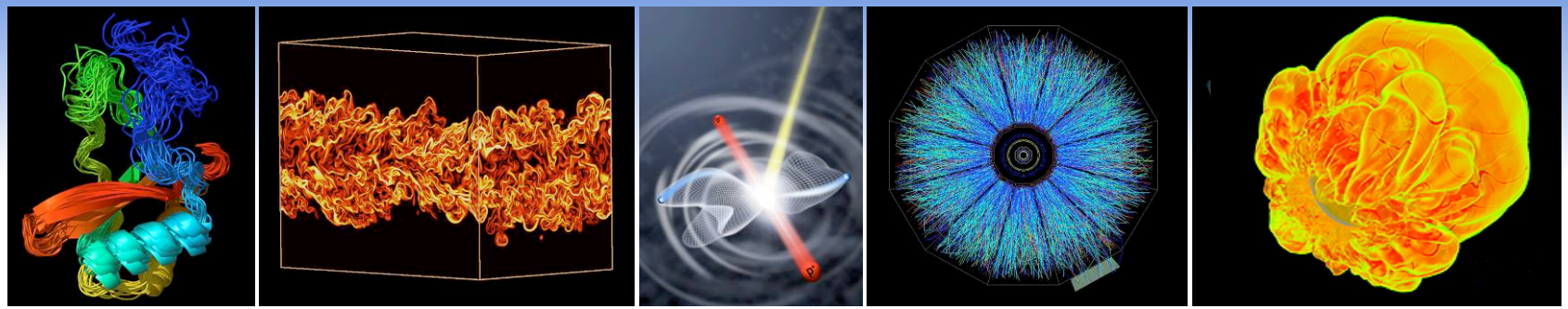


Multicore Programming Preparing for Hopper

Alice Koniges, Berkeley Lab/NERSC
With input from:

John Shalf, Berkeley Lab/NERSC

Rusty Lusk, Argonne National Laboratory (ANL)

Rolf Rabenseifner, HLRS, University of Stuttgart, Germany

Gabriele Jost, Texas Advanced Computing Center

U.S. DEPARTMENT OF ENERGY

NERSC

BERKELEY LAB

Despite continued "packing" of transistors, performance is flatlining

- **New Constraints**
  - 15 years of *exponential* clock rate growth has ended
- **But Moore's Law continues!**
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!

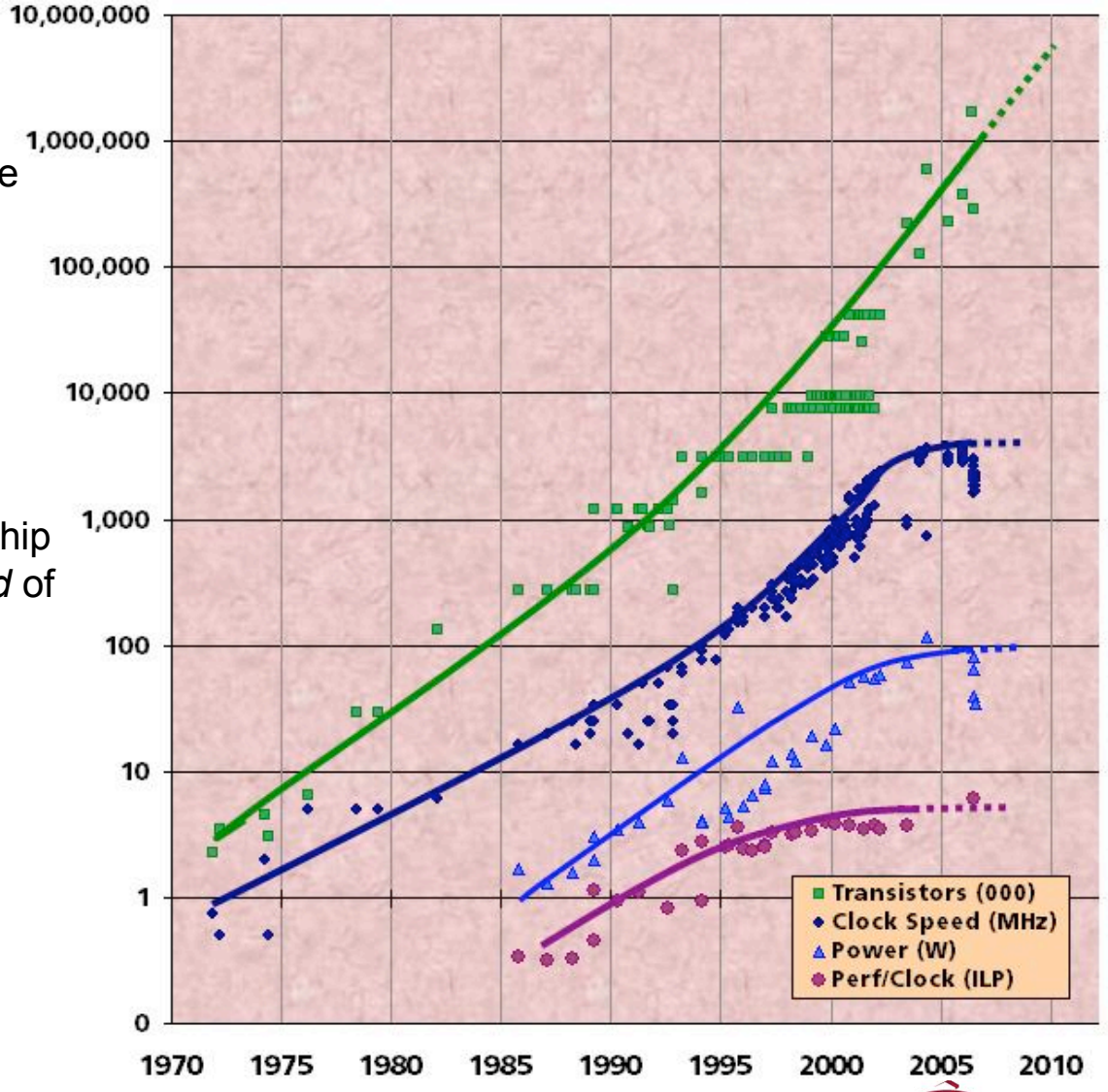


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Computer Centers and Vendors are Responding with New Designs

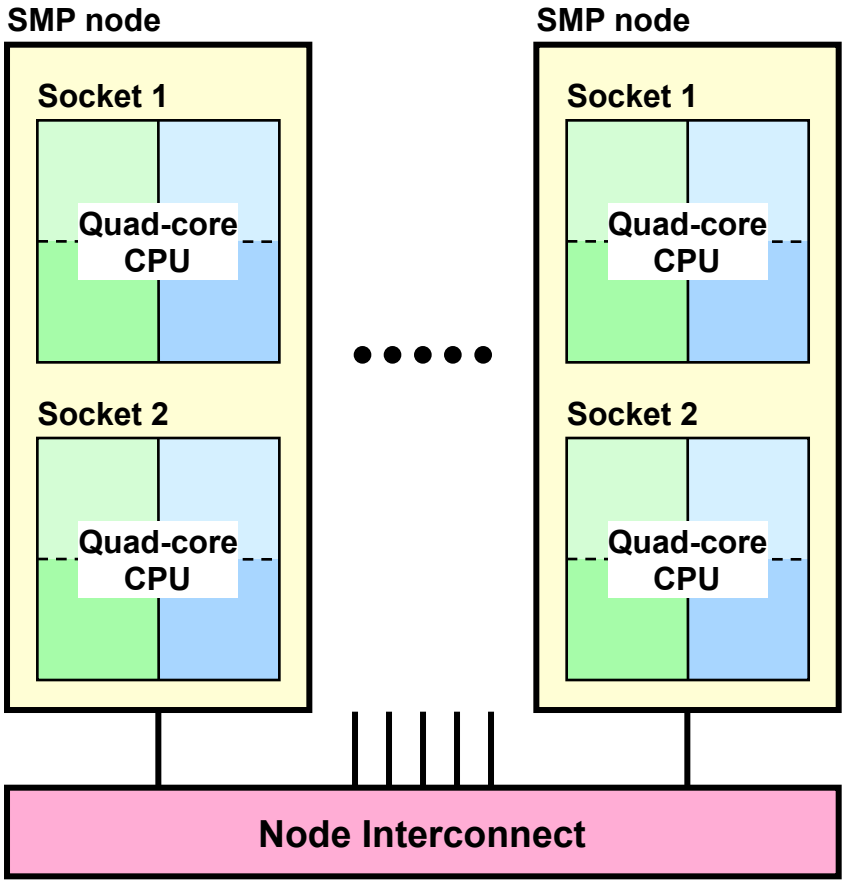
- **Virtually all upcoming systems have various forms of heterogeneous parallelism**
  - **NERSC6 with its multicore design TBA**
  - **Blue Waters with its Power7 hardware threaded design**

  8 cores, 12 execution units/core, 4-way SMT/core
  - **ASC Sequoia (follow-on to BlueGene design) with anticipated support for transactional memory**
- **Experts everywhere are preparing for this architecture revolution with new languages, extensions to old languages, tools (and angst)**
- **Our goal at NERSC is to make this as painless as possible for application scientists**
- **We invite you to comment on our plans**

What's Wrong with MPI Everywhere

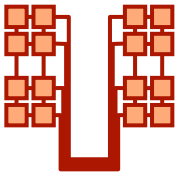
- **We can run 1 MPI process per core (flat model for parallelism)**
  - This works now and will work for a while
  - But this is wasteful of intra-chip latency and bandwidth (100x lower latency and 100x higher bandwidth on chip than off-chip)
  - Model has diverged from reality (the machine is ***NOT*** flat)
- **How long will it continue working?**
  - 4 - 8 cores? Probably. 128 - 1024 cores? Probably not.
  - Depends on performance expectations
- **What is the problem?**
  - Latency: some copying required by semantics
  - Memory utilization: partitioning data for separate address space requires some replication
    - How big is your per core subgrid? At 10x10x10, over 1/2 of the points are surface points, probably replicated
  - Memory bandwidth: extra state means extra bandwidth
  - Weak scaling: success model for the "cluster era;" will not be for the many core era -- not enough memory per core
  - Heterogeneity: MPI per CUDA thread-block?

Within the MPI-OpenMP hybrid model, there are variants depending on system and application

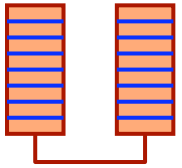


Which programming model is fastest?

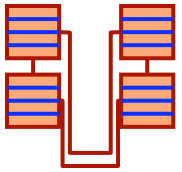
MPI everywhere?



Fully hybrid MPI & OpenMP?

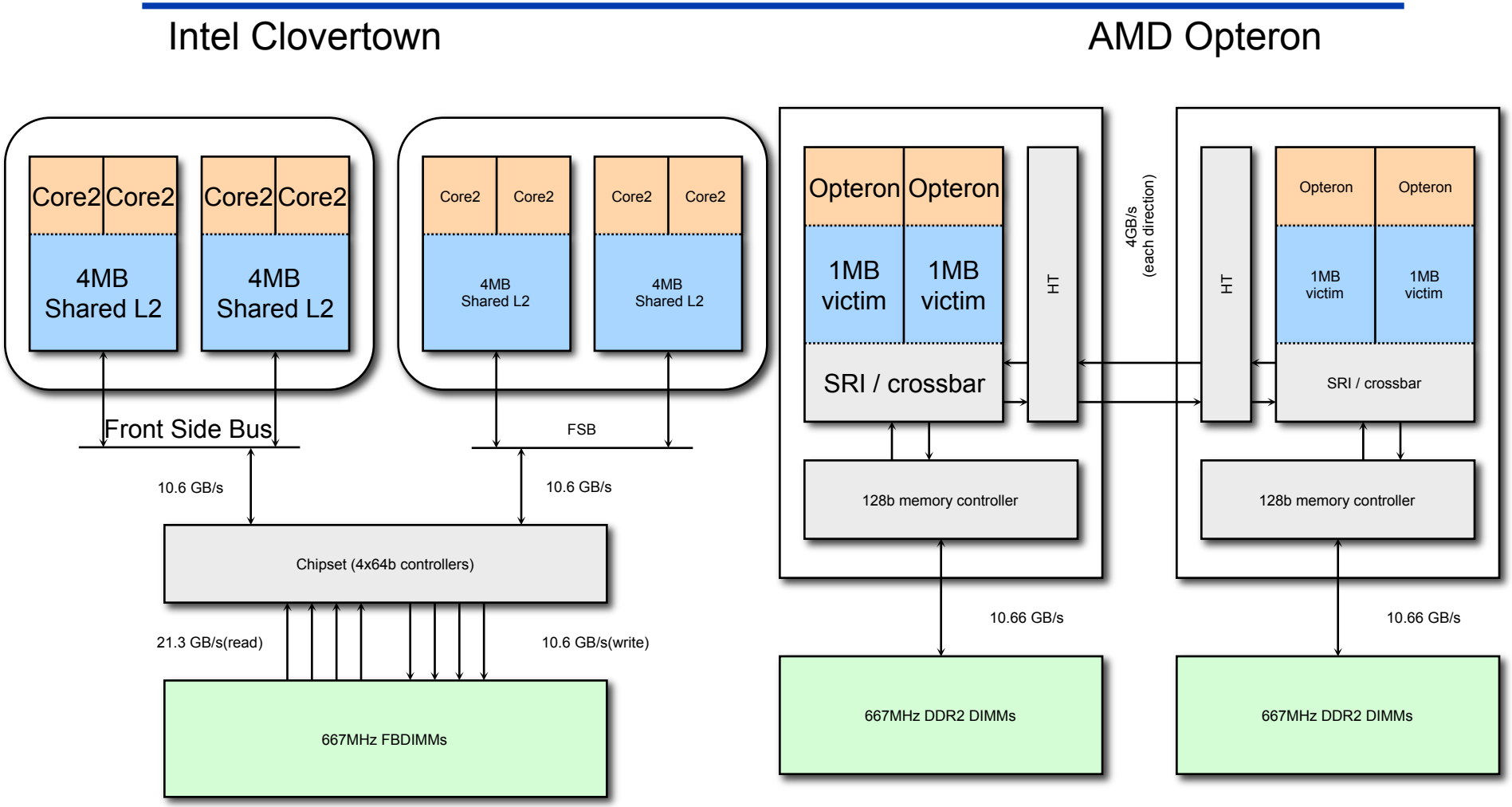


In - between? (Mixed model)



Historically hybrid programming can be **slower** than pure MPI ?

Current Multicore SMP Systems can have different memory access and cache use patterns



Thus a flat memory model like standard OpenMP may not be sufficient for the core programming model

Programming Models are Changing to Accommodate the Multicore Revolution

- **A *programming model* is an abstraction that we program by writing instructions**
- **Multiple classes of models differ in how we think about communication and synchronization among processes**
  - Shared memory
  - Distributed memory
  - Some of each
- **Shared Memory (really globally addressable)**
  - Processes (or threads) communicate through memory addresses accessible to each
- **Distributed memory**
  - Processes move data from one address space to another via sending and receiving messages
- **Multiple cores per node make the shared-memory model efficient and inexpensive**

Back to basics: writing parallel programs can be expressed in different ways

- **Parallel programming models are expressed:**
  - In libraries callable from conventional languages
  - In languages compiled by their own special compilers
  - In structured comments that modify the behavior of a conventional compiler
  - New ideas or “natural ways” to parallel program

Need to think beyond the MPI – everywhere model using a callable library

MPI and Threads

- **MPI describes parallelism between *processes* (with separate address spaces)**
- ***Thread* parallelism provides a shared-memory model within a process**
- **OpenMP and Pthreads are common but different models**
  - OpenMP provides convenient features for loop-level parallelism
  - Pthreads provide more complex and dynamic approaches
  - OpenMP 3.0 (which adds task parallelism) adds some of these capabilities to OpenMP
- **MPI combined with OpenMP is the most common current means of adapting for heterogenous architecures**
  - Doesn't always work
  - Is not able to deal with NUMA on the nodes

The PGAS Languages

- **PGAS (Partitioned Global Address Space) languages attempt to combine the convenience of the global view of data with awareness of data locality, for performance**
  - Co-Array Fortran, an extension to Fortran-90)
    - SPMD – Single program, multiple data
    - Replicated to a number of images
    - Variables declared as co-arrays are accessible by another image through a set of array subscripts, delimited by [ ] and mapped to image indices by the usual rule
  - UPC (Unified Parallel C), an extension to C
    - UPC is an extension of C (not C++) with shared and local addresses
    - *Shared* keyword in type declarations
    - What we have been calling processes are called *threads* in UPC
      - and may be implemented as OS threads
  - Titanium, a parallel version of Java
    - Titanium is a PGAS language based on Java
    - The langauge is compiled, not interpreted
      - Implementations do not use the JVM

New Models MPI + x or ?

- **We are considering new programming models that combine MPI with another language such as UPC or CAF in addition to the standard hybrid method of MPI+OpenMP**
- **There are also a large number of new languages to consider:**
  - Intels's CnC or Concurrent Collections
    - Invites users to rethink their problem into 2 pieces:
      - Data dependence and control dependence
  - Microsoft's parallel language suites including:
    - Axum
    - Parallel Patterns Library
  - OpenCL
    - A framework for writing parallel programs that execute heterogeneous platforms
- **Also, most current languages (OpenMP, MPI, etc) are looking at what changes should be made for architecture evolution**

Path Forward

NERSC is a new Cray Center of Excellence

- Joint with Cray we each have dedicated 2FTE's over the next two years to examine programming models and prepare training materials
- Plan is to start with the NERSC benchmark series, particularly those which are already hybrid, and characterize their performance and effectiveness
- Then we will move on to other benchmarks and consider how to add other models, the MPI + x model
- Along the way, we will develop and improve tools for hybrid analysis
- By the end of 2010, we will have course material prepared based on these experiences

The Parallel Motifs Program

- NERSC is teaming with members of the Berkeley Lab, Berkeley Campus, Microsoft, and Intel to quantify smaller units of code that represent the majority of scientific computations
- We are writing these codes in different languages for people do download and examine
- We are starting a website parallelmotifs.org, which will eventually house the motif codes for analysis and experimentation

The Computational Science and Engineering Petascale Initiative

- NERSC received 3.125M$ in stimulus money for this initiative
- This money is being used to fund 8 post-docs who will work closely with application codes to enhance performance and consider new models
- Although the money is tied to helping certain project areas, the benefit to general NERSC will be evident through knowledge gained and teaching materials that result from this research
- The post-docs will spend half of their time at the NERSC facility, to directly interact with NERSC staff to ensure there is communication of new ideas and “what works”

Courses and Presentations

- **NERSC is giving presentations to help users at major conferences, for example:**
  - "Application Supercomputing and the Many-Core Paradigm Shift," full day tutorial at SC09 organized by NERSC, yet includes expert speakers from other laboratories
- **We have organized dedicated sessions at the SIAM Conference on Scientific Computing, Seattle, Feb. 10 – 14 2010**
  - These invited talks will feature keynote speakers on the PGAS languages, hybrid programming, Microsoft parallel languages, Intel parallel languages, and the parallel motifs project
- **Summer Tutorials at ParCFD, SciDAC**
- **Participated in ParLab Boot Camp with all lectures now online**
- **We invite you to suggest other forums and venues for such presentations**

Collaboration with other sites for tools, languages, and courses

- **Some of our current collaborations include:**
  - Lawrence Livermore National Laboratory
    - compiler tools that allow for optimization (ROSE) and tools for analysis
    - Specific OpenMP analyzers and other correctness tools
  - High Performance Computing Center Stuttgart
    - collaboration on teaching materials and sample codes for hybrid programming, UPC, CAF and other topics
  - Texas Advanced Computer Center
    - collaboration on designing and testing OpenMP and hybrid codes and models
  - Argonne National Laboratory
    - collaboration on teaching materials and hybrid analysis tools
  - Microsoft and Intel, on languages, tools, and motifs

Talking Points and Questions

- The multi-core revolution – what is your opinion?
- How interested are application developers in changing their codes to get performance?
- What other plans should we be making?
- INPUT and QUESTIONS ??