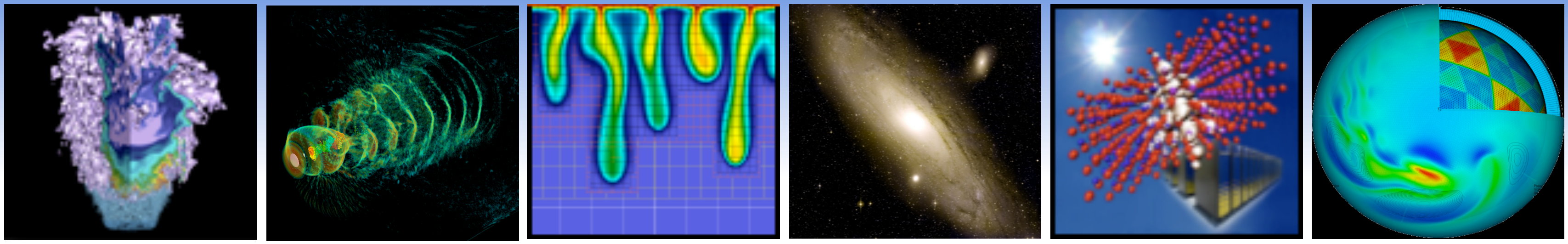


What’s Ahead for Fusion Computing

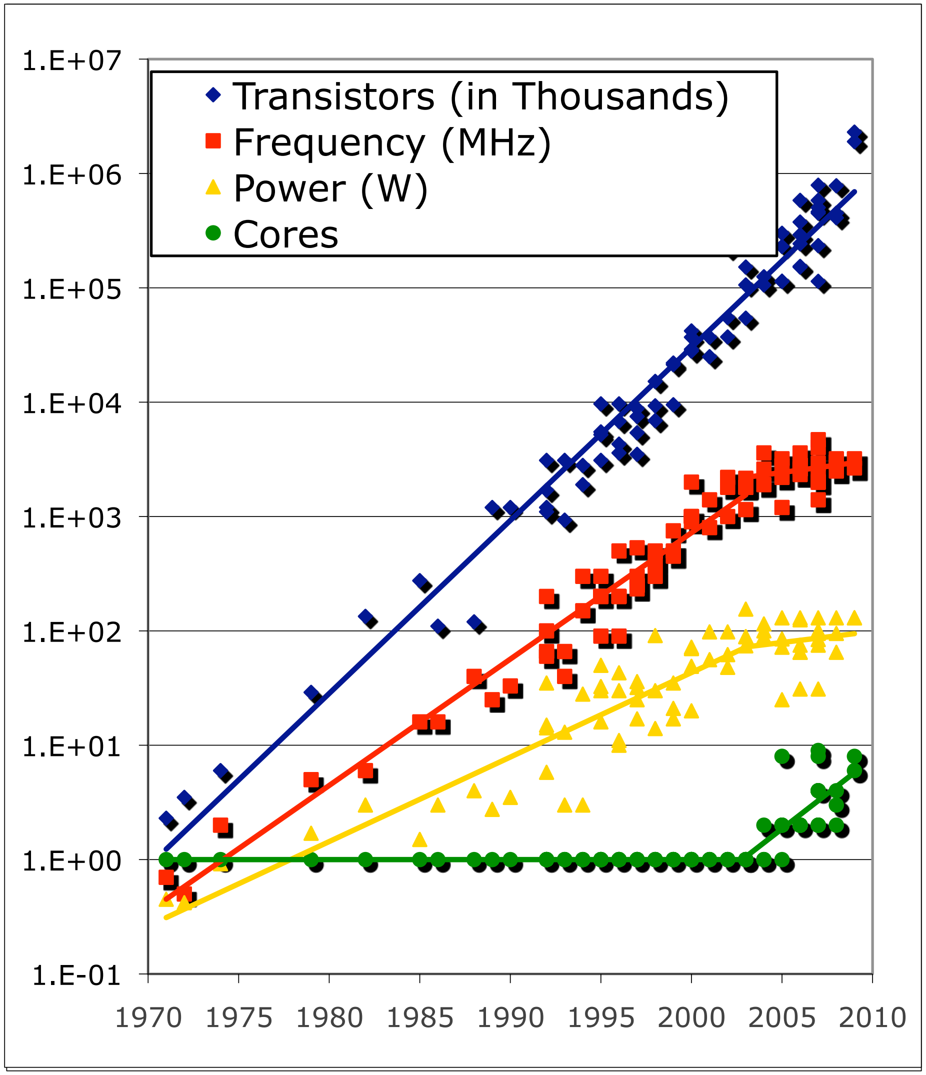
Alice Koniges, NERSC, Berkeley Lab

Robert Preissl, Jihan Kim, John Shalf
Gabriele Jost (TACC), Rolf Rabenseifner (HLRS)
Cray COE at NERSC
Cloud Computing at NERSC

Sherwood Fusion Theory
April 2010

Physics of chip manufacturing has caused a Multicore Revolution

- **Power density limit (yellow) caused processor clock speed to tail off (red)**
  - 15 years of *exponential* clock rate growth has ended
- **Moore's law (doubling of transistors) continues (blue)**
  - How do we use transistors to increase performance
- **Cores per chip is growing (green)**



Sherwood 2010

Computer Centers and Vendors are Responding with New Designs

- **Virtually all upcoming systems have various forms of heterogeneous parallelism**
  - **NERSC6 Hopper with its multicore design**

  Two 12 core on a node

  - **Blue Waters with its Power7 hardware threaded design**

  8 cores, 12 execution units/core, 4-way SMT/core

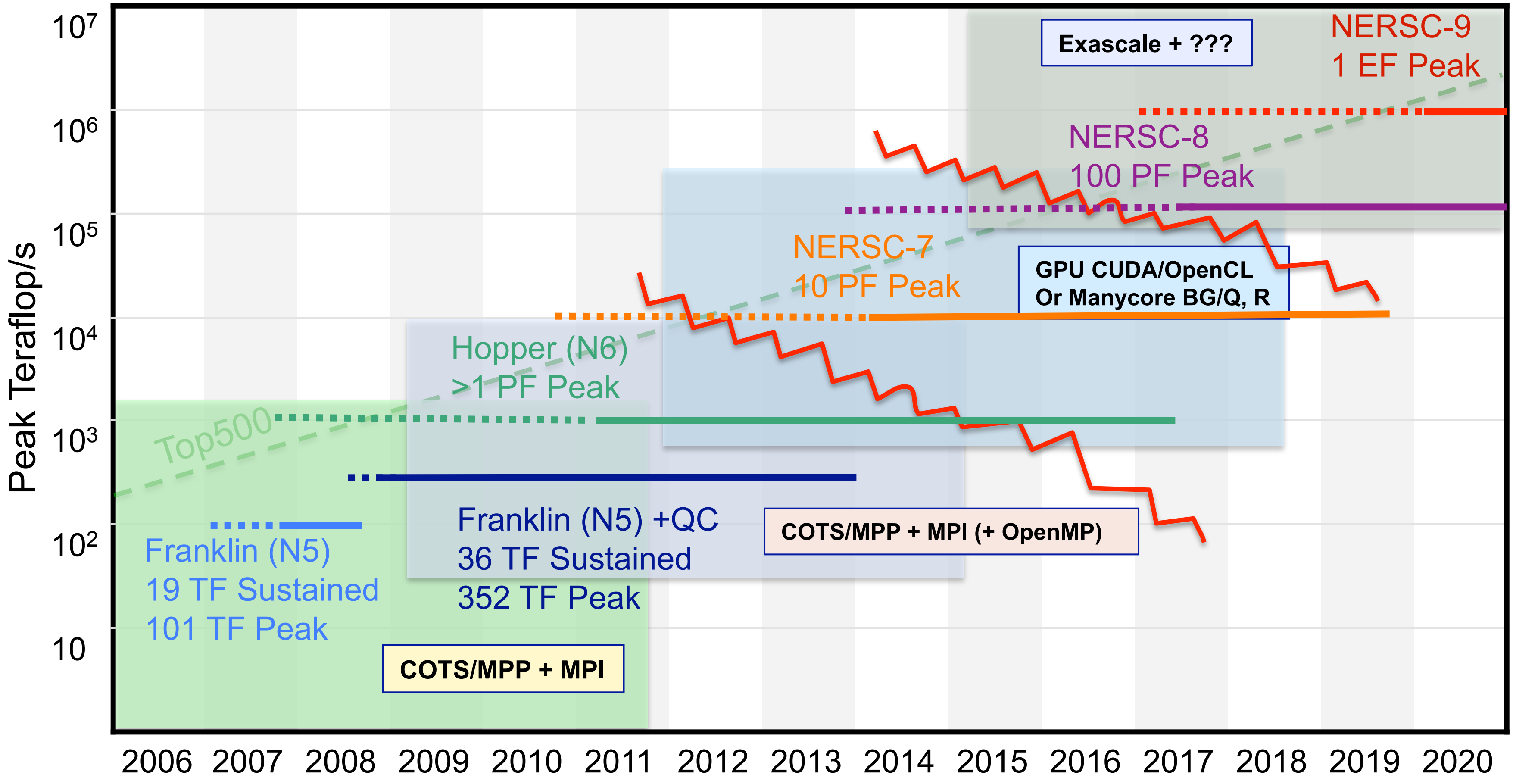
  - **ASC Sequoia (follow-on to BlueGene design) with anticipated support for transactional memory**
- **Experts everywhere are preparing for this architecture revolution with new languages, extensions to old languages, tools (and angst)**
- **What does this mean for fusion applications?**

The next advances in computing will be highly disruptive (again)

- Gigaflops (10^9) to Teraflops (10^12) was highly disruptive
  - Moved from vector machines to MPPs with message passing
  - Required new algorithms and software
- Teraflops to Petaflops (10^15) was *not* very disruptive
  - Continued with MPI+Fortran/C/C++ with incremental advances
- Petaflops to Exaflops (10^18) will be highly disruptive
  - No clock increases → hundreds of simple "cores" per chip
  - Less memory and bandwidth → cores are not MPI engines
  - x86 too energy intensive → more technology diversity (GPUs/accel.)
  - Programmer controlled memory hierarchies likely
- <u>Computing at every scale will be *transformed (not just exascale)*</u>

Following the paths to exascale can transform Fusion computing

- Multicore: replicated complex cores(X86 and Power7)
- Manycore/Embedded:Simpler, Low power (BlueGene)
- GPU/Accelerator: specialized processors from gaming space (Nvidia Fermi, Cell)



NERSC-6 System “Hopper” is coming in 2 phases

- Cray “Baker” Nodes with Gemini Interconnect in Phase 2

2003	2005	2007	2009	2009	2010
Opteron	Opteron	Barcelona	Shanghai	Istanbul	Magny-Cours
Single Core	Dual Core	Quad Core	Quad Core	6 - Core	12 - Core
90 nm	90nm	65 nm	45 nm	45 nm	45 nm

Phase 1: Cray XT5

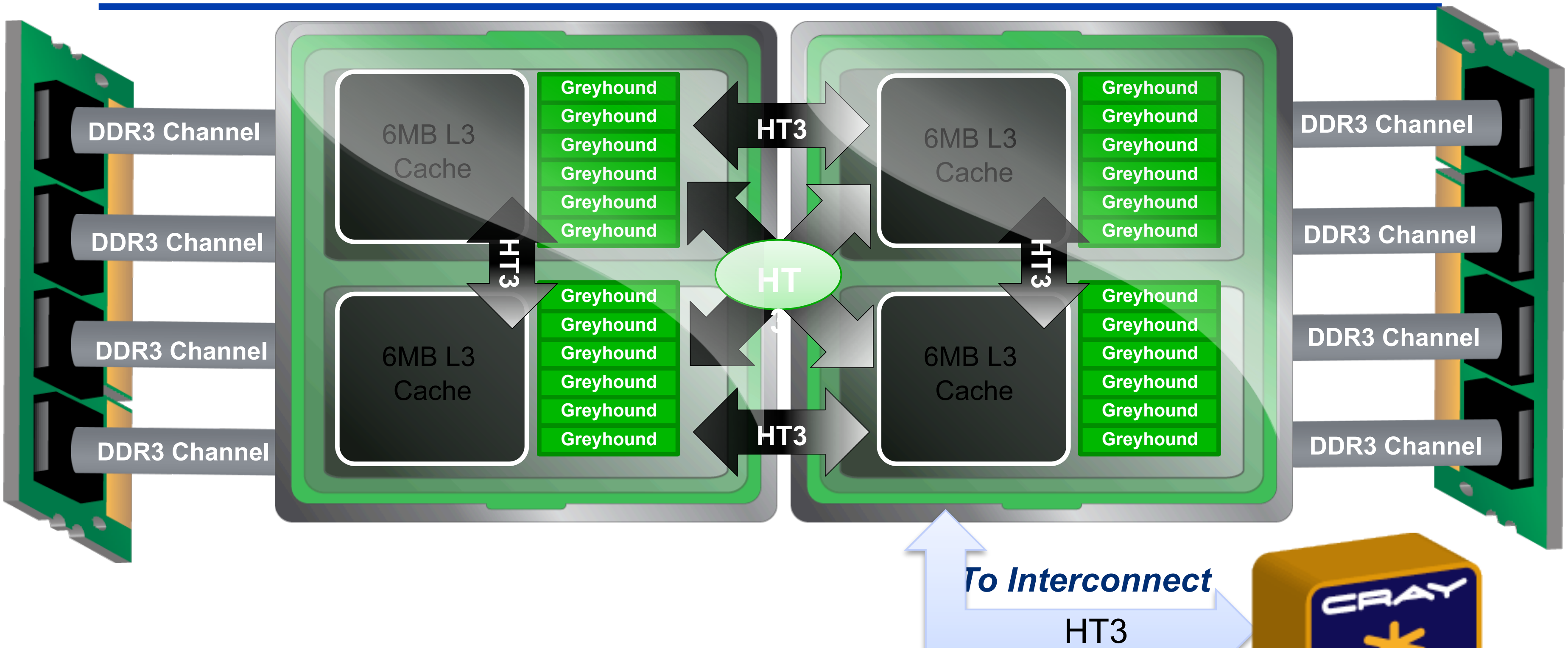
- 668 nodes, 5,344 cores
- 2.4 GHz AMD Shanghai
- 2 PB disk, 25 GB/s
- Air cooled

Phase 2: Cray system

- > 1 Pflop/s peak, ~150K cores
- Two 12-core MCMs per node
- AMD Magny-Cours
- 2 PB disk, 80 GB/s
- Liquid cooled

3Q09	4Q09	1Q10	2Q10	3Q10	4Q10

Heterogenous memory access within node complicates programming model choices



- **Baker Node Details--24-core Magny Cours**
  - 2 Multi-Chip Modules, 4 Opteron Dies
  - 8 Channels of DDR3 Bandwidth to 8 DIMMs
  - 24 (or 16) Computational Cores, 24 MB of L3 cache
  - Non-uniform memory access within the node

Standard model of “MPI everywhere” will likely not last forever

- **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?

Programming Models are Changing to Accommodate the Multicore Revolution

- **Programming 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**

New Models MPI + x or ? are in the future

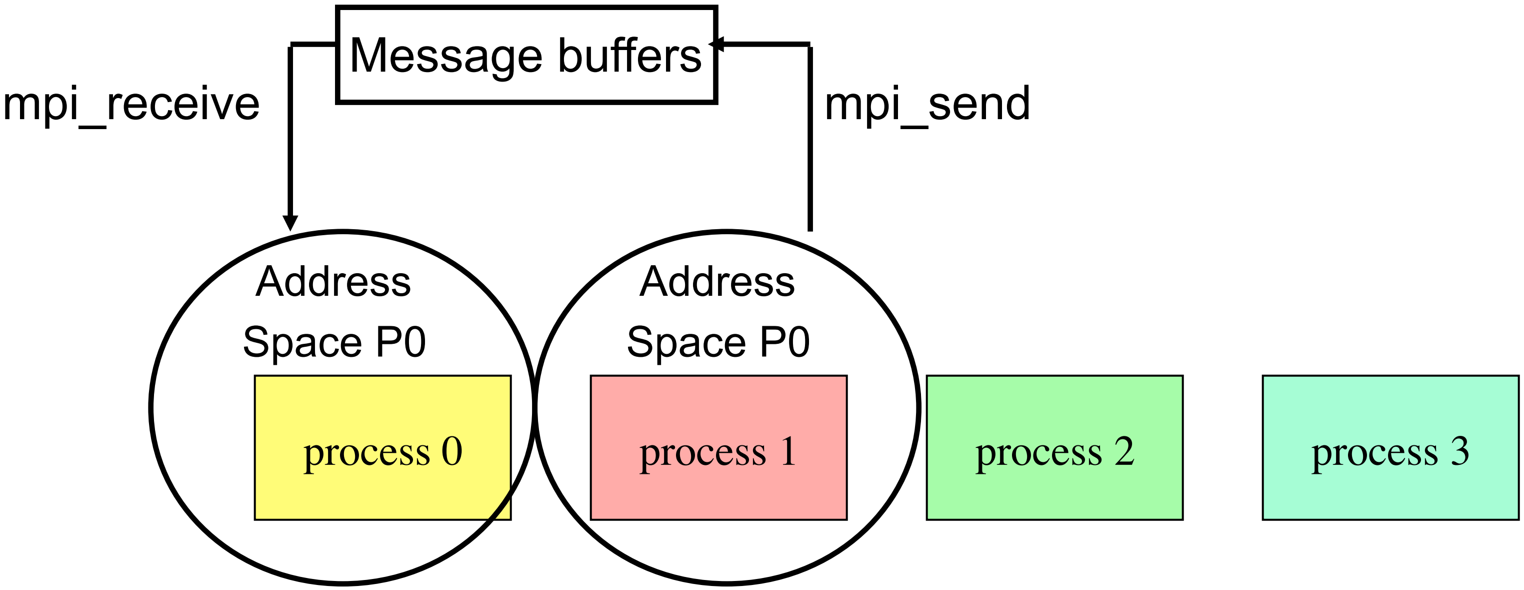
- MPI is likely to stay – for a while at least
- However, simpler cores and limited memory are likely to make the MPI-everywhere model obsolete.
- We are considering new programming models that combine MPI with another language such as UPC or CAF in addition to the standard hybrid method of MPI+OpenMP

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

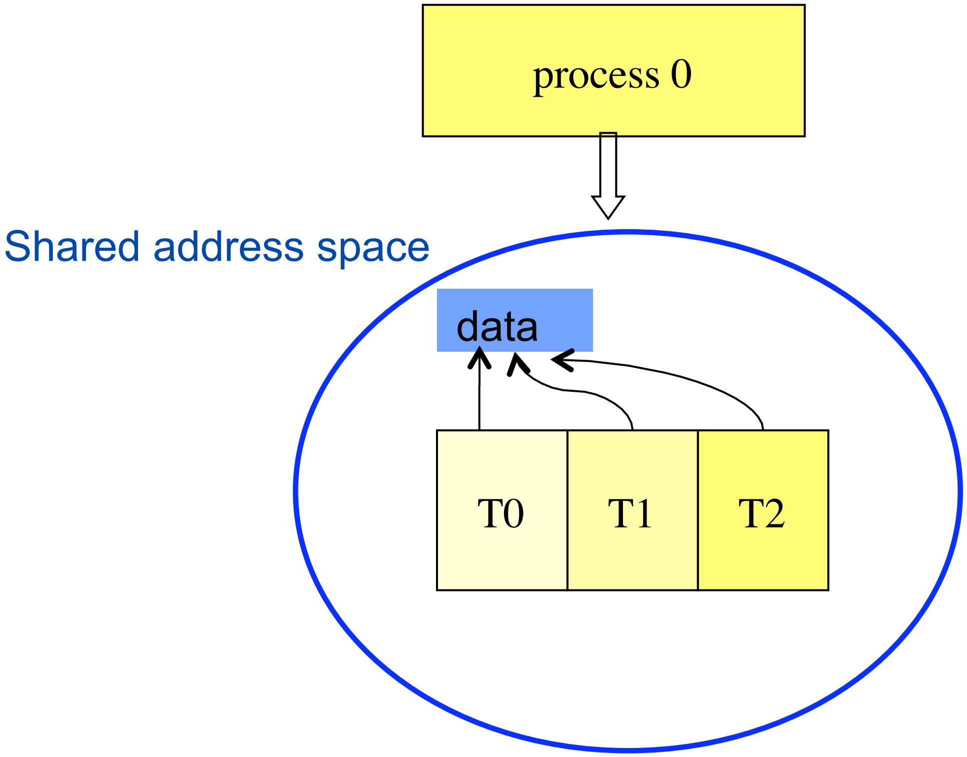
MPI Memory Model

- **Message Passing Interface**
- **Memory Model:**
  - MPI assumes a private address space
  - Private address space for each MPI Process
  - Data needs to be explicitly communicated
- **Applies to distributed and shared memory computer architectures**



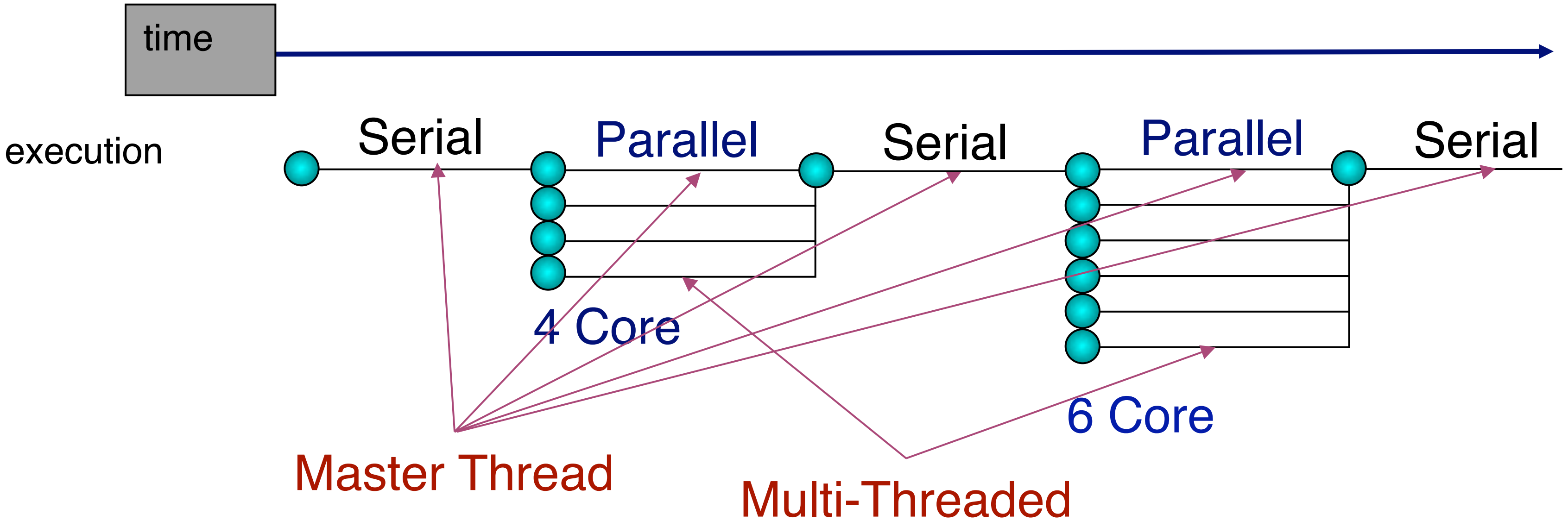
OpenMP Memory Model

- OpenMP assumes a shared address space
- No communication is required between threads
- Thread Synchronization is required when accessing shared data



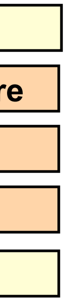
OpenMP Code General Structure

- Fork-Join Model:
- Execution begins with a single “Master Thread”
- A team of threads is created at each parallel region
- Threads are joined at the end of parallel regions
- Execution is continued after parallel region by the Master Thread until the beginning of the next parallel region

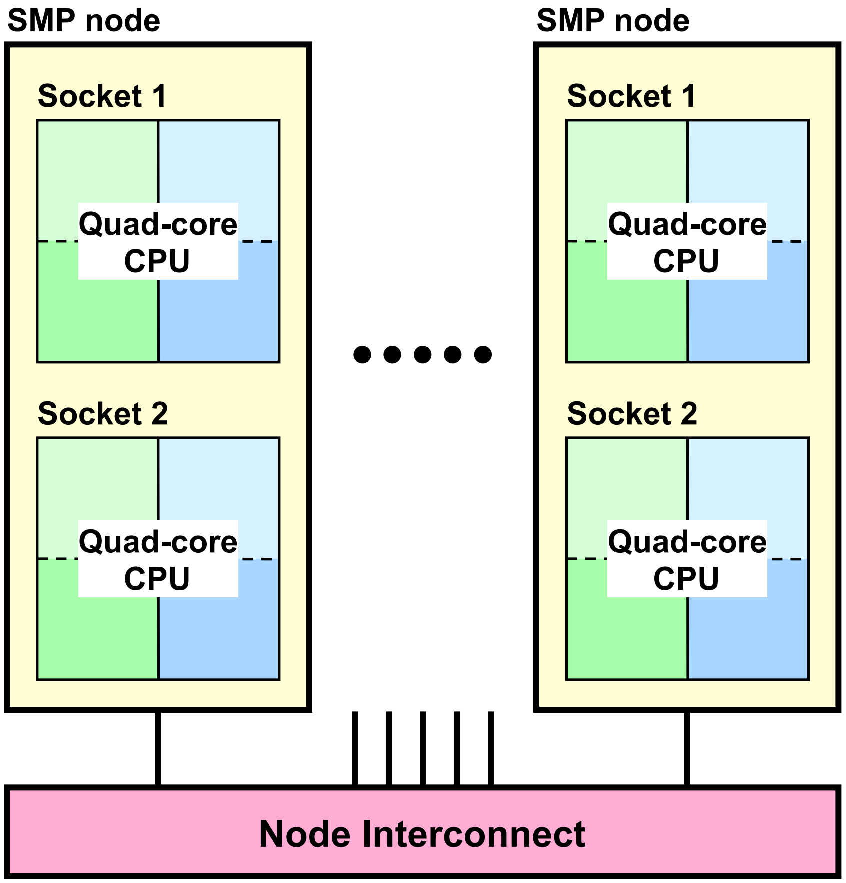


The PGAS Languages

- **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

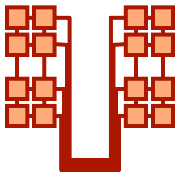


Even the MPI-OpenMP hybrid model, is complicated on multicore

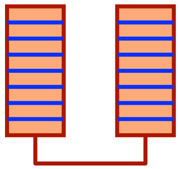


Which programming model is fastest?

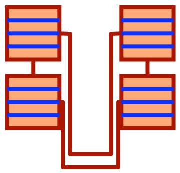
MPI everywhere?



Fully hybrid MPI & OpenMP?



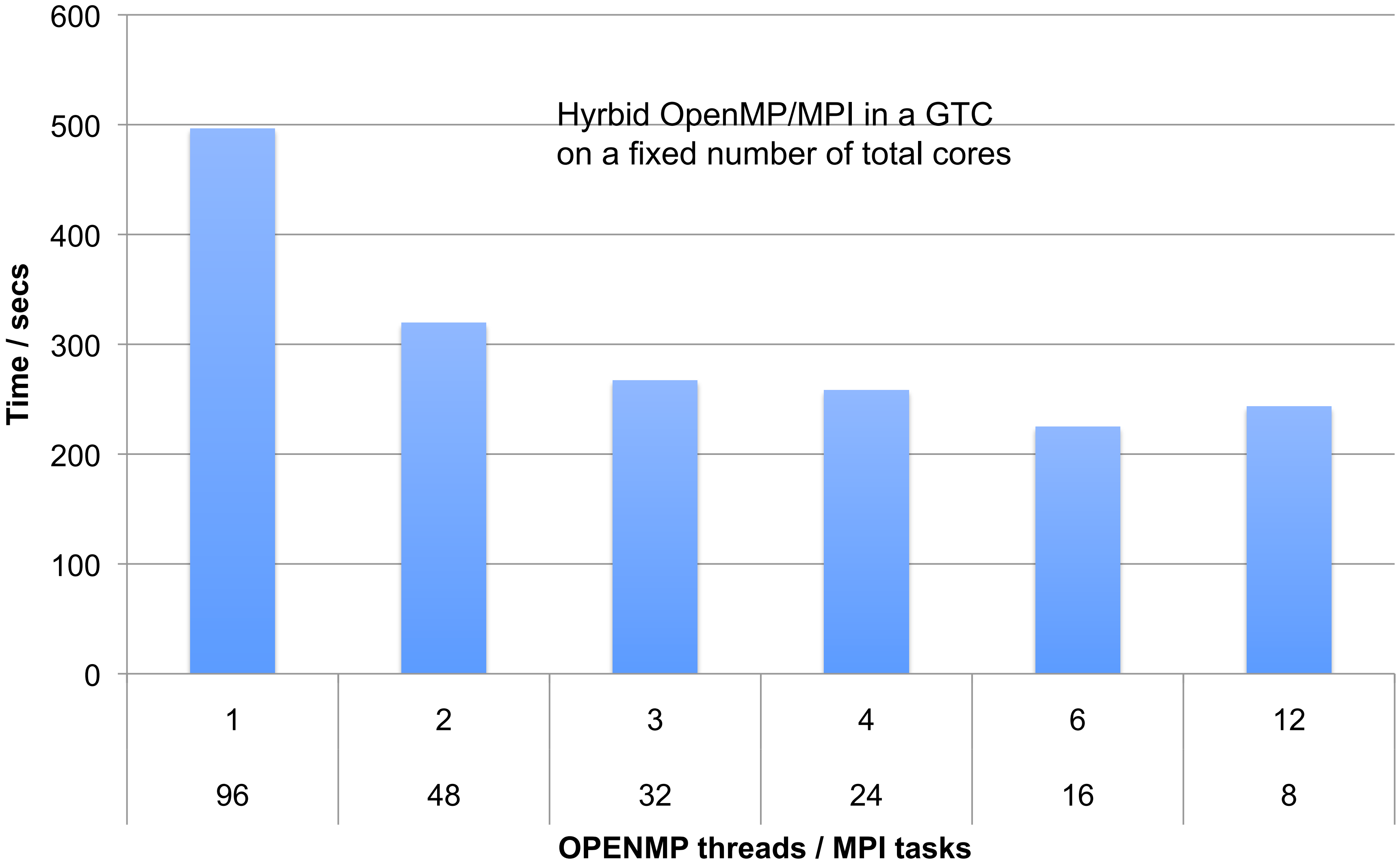
In - between? (Mixed model)



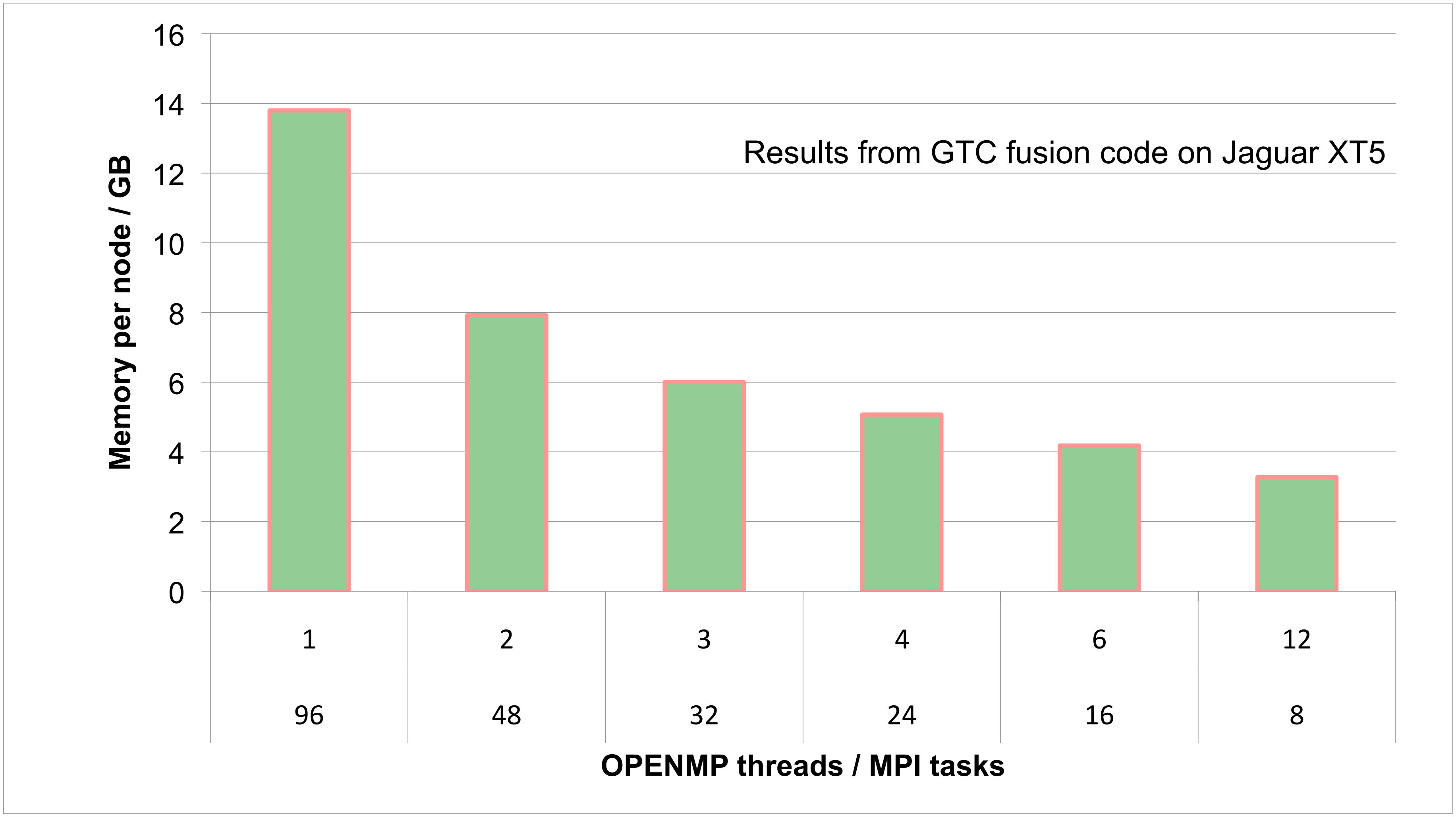
Historically hybrid programming can be **slower** than pure MPI ?

Some examples of hybrid (MPI+OpenMP)

In this code, gain from hybrid tails off after 6 cores due to NUMA effects

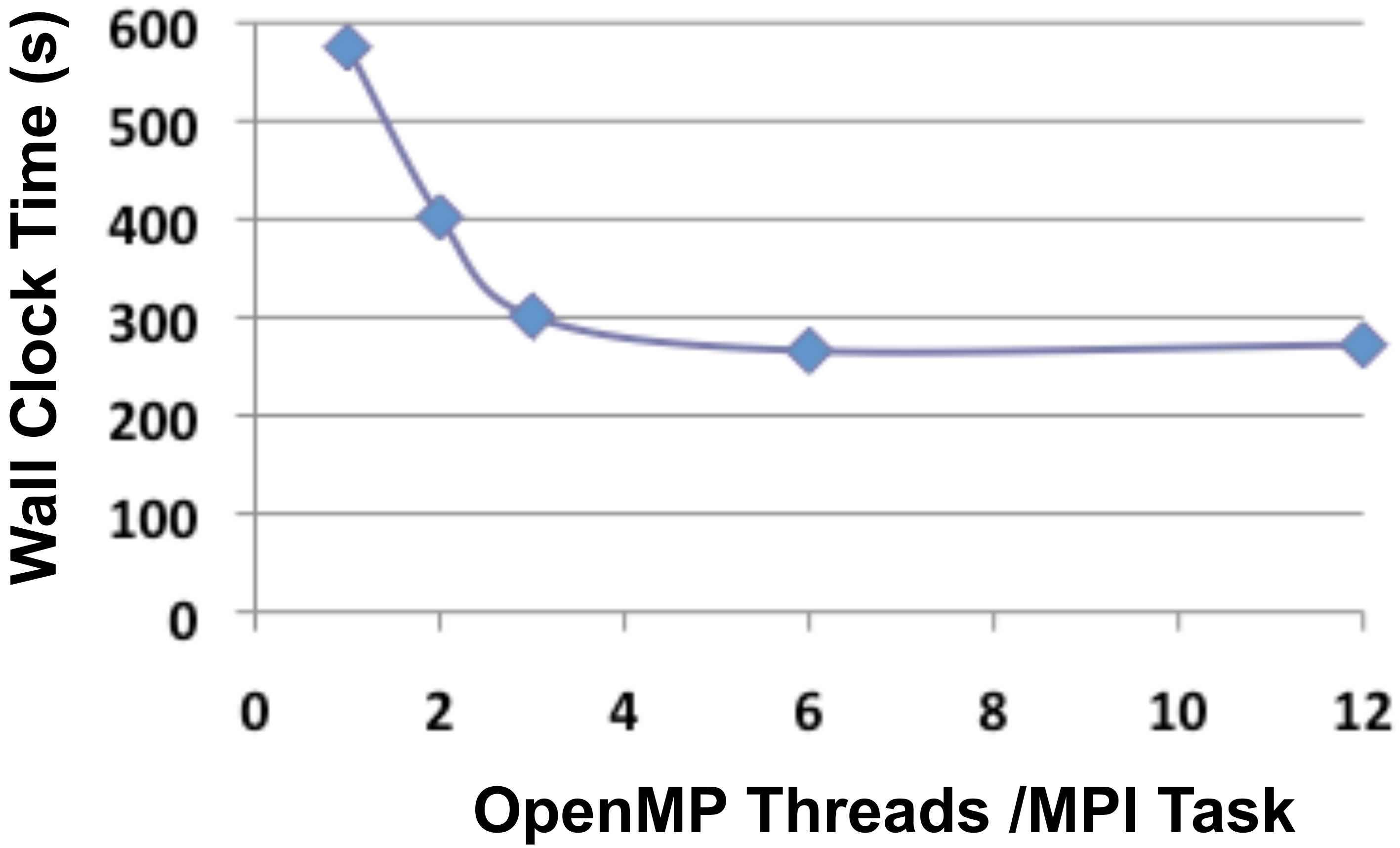


Memory Usage is a major gain in using Hybrid on a fixed number of cores



Results from GTC fusion code on Jaguar XT5

Hybrid OpenMP in a Climate Code
Fixed number of 240 MPI Tasks



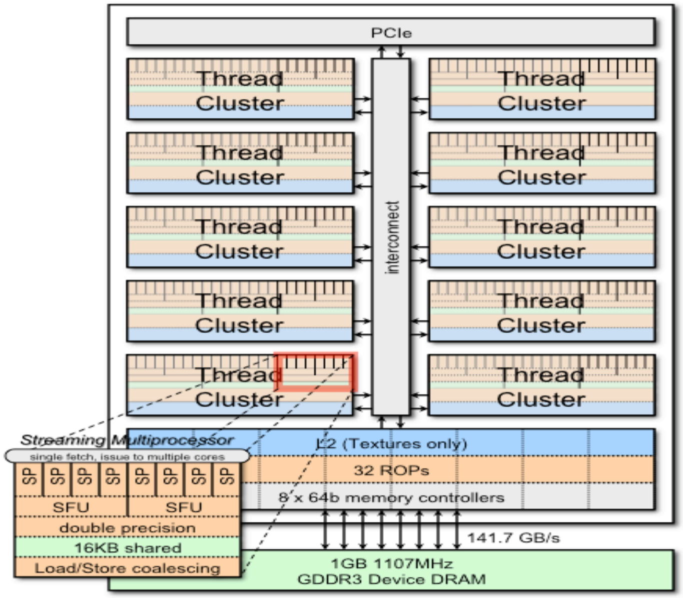
Other Current Developments

GPUs

Clouds

Accelerator/GPU Solution to Power Density Crises

- GPUs are early adopter of simpler cores
  - Many simple functional units, specialized for graphics computations
  - Dramatic shift in programming model
  - CUDA is huge advance over GPGPUs, but programmability is still a concern
- GPU's are also used as accelerators (e.g., Roadrunner)
  - Disjoint memory space difficult to manage
  - PCIe bottleneck can compromise delivered performance
  - Similar to using vector co-processors on the CM5, must move into local memory



NERSC Dirac Cluster is part of a research program

- 48 nodes
- 2 quad core Nehalems 2.4GHz, 24 GB per node memory
- currently has Tesla accelerators in it, but will be upgraded with Fermi when they come out
- The fermi (E9) boards will have ECC and 3 GB of memory per board. (16 lane PCIe to each board)
- One Fermi board per host node.
- OS same as Carver
- The nodes will be connected to the carver/magellan switch infrastructure with QDR IB 4x links (same as carver nodes).
- Each Fermi card is 1Tflop/s so Dirac's theoretical peak performance will be 48 Tflops

Fusion Codes with GPU's

- **Tech-X Work:**
- **Finite-difference time domain computations needed in edge for**
  - Accurate representation of antennas
  - Analysis of nonlinear effects and sheaths
- **Prototyping (exporting VORPAL geometry) using GPUlib (next slide):**
  - 40x speedup GPU/CPU
  - Bandwidth limited

- **NERSC GTC port to GPUs**

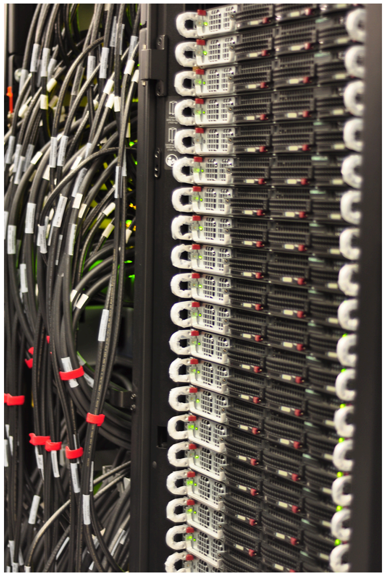
We are experimenting with GPUs for a variety of codes

- **Q-Chem used to model Carbon capture, i.e., reactivity of CO_2 with other materials (input from quantum calculations)**
  - **Molecular equilibrium structures:**
  - **2nd order Moller-Plesset perturbation theory (MP2)**

$$E_0^{(2)} = \frac{1}{4} \sum_{a,b,r,s} \frac{\left| \langle ab \| rs \rangle \right|^2}{\varepsilon_a + \varepsilon_b - \varepsilon_r - \varepsilon_s}$$

$$\langle ij \| kl \rangle = \langle ij | kl \rangle - \langle ij | lk \rangle = \int dx_1 dx_2 \chi_i^*(x_1) \chi_j^*(x_2) r_{12}^{-1} (1 - P_{12}) \chi_k(x_1) \chi_l(x_2)$$

  - **Expensive fifth-order computational dependence on system size**
  - **Focus is to speed up the MP2 portion of the Q-Chem code**



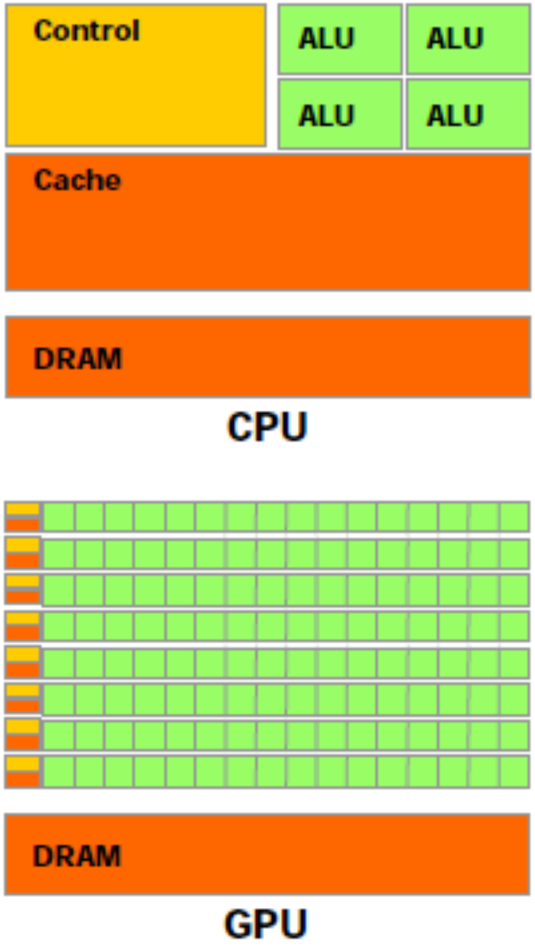
Fermi GPU Racks - NERSC

Jihan Kim[1], Alice Koniges[1], Berend Smit[2], Martin Head-Gordon[2]

[1]NERSC/LBNL and [2]UC Berkeley

The CPU and the GPU are significantly different

- High-performance computing with GPUs
  - Tesla/Turing NERSC cluster (4 Nvidia FX-5800 Quadroplex, 4GB Memory)
  - New GPU cluster at NERSC (48 Nvidia Fermi cards)
- Parallelizing codes (C++/Fortran)
  - CUDA (compute unified device architecture): parallel architecture developed by Nvidia
  - Some can be done with libraries: replace blas routines with cublas (50-75GFLOP/sec)
  - Asynchronous operation with GPU/CPU (overlap I/O operations with blas3 matrix-matrix multiplications)
  - About 8-10 times speedup



Code Example

Regular MP2

```
QAllocDouble(C, …);
QAllocDouble(iaP, …);
QAllocDouble(iajb, …);

for (i = 0; i < num1; i++)
{
    LongFileMan(FM_Read, iaP);
    for (j = 0; j < num2; j++)
    {   …
        for (k = 0; k < num3; k++)
        {
            LongFileMan(FM_Read(k), C);

            AtimsB(iajb, iaP, C, …);
            …
        }
    }
}
```

MP2 with CUDA

```
#include “cublas.h”
#include “cuda_runtime_api.h”
cublasStatus = status;
Status = cublasInit();

QAllocDouble(C, …);
QAllocDouble(iaP, …);
QAllocDouble(iajb, …);
cudaMalloc((void**)d_C, sizeof(C));
cudaMalloc((void**)d_iaP, sizeof(iaP));
cudaMalloc((void**)d_iajb, sizeof(iajb));

for (i = 0; i < num1; i++)
{   LongFileMan(FM_Read, iaP);
    cudaMemcpy(d_iaP, iaP, sizeof(iaP), cudaMemcpyHostToDevice);
    for (j = 0; j < num2; j++)
    {   LongFileMan(FM_Read(0), C);
        cudaMemcpy(d_C, C, sizeof(C), cudaMemcpyHostToDevice);
        for (k = 1; k < num3; k++)
        {   cublasDgemm(‘n’, ‘n’, m, n, k, 1.0, d_iaP, lda, d_C, ldb, 0, d_iajb, ldc)
            LongFileMan(FM_Read(k), C);
            cudaMemcpy(d_C, C, sizeof(C), cudaMemcpyHostToDevice);
            …
        }
    }
}
```

Magellan – Exploring Cloud Computing

A Test bed to explore Cloud Computing for Science

- National Energy Research Scientific Computing Center (NERSC)
- Argonne Leadership Computing Facility (ALCF)
- $32M total funding, equally divided between the two facilities
- Funded by DOE under the American Recovery and Reinvestment Act (ARRA)

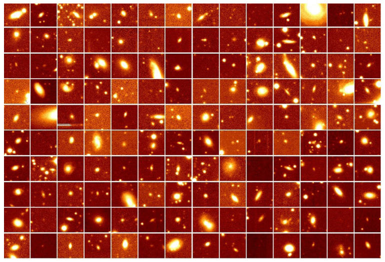
FERDINAN· MAGELLANVS· SVPERATIS
ANTARCTICI· FRETI· ANGVSTIIS· CLARISS

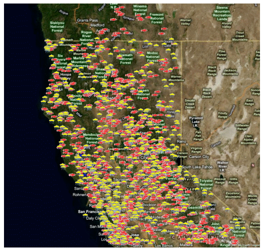
Magellan Research Agenda

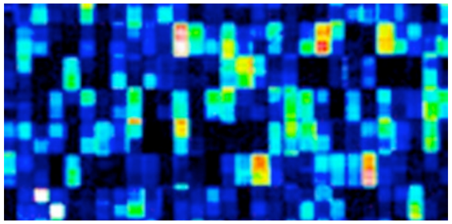
- What are the unique needs and features of a science cloud?
- What applications can efficiently run on a cloud?
- Are cloud computing Programming Models such as Hadoop effective for scientific applications?
- Can scientific applications use a data-as-a-service or software-as-a-service model?
- Is it practical to deploy a single logical cloud across multiple DOE sites?
- What are the security implications of user-controlled cloud images?
- What is the cost and energy efficiency of clouds?

Why Clouds for Science?

- **On-demand access to compute resources**
  - e.g. Cycles from a credit card. Avoid batch wait times.
- **Overflow capacity to supplement existing systems**
  - e.g., Berkeley Water Center has analysis that far exceeds the capacity of desktops
- **Customized and controlled environments**
  - e.g. Supernova Factory codes have sensitivity to OS/compiler version
  - CernVM provides a fully integrated environment for LHC analysis
- **Parallel programming models for data intensive science**
  - e.g., BLAST with Hadoop



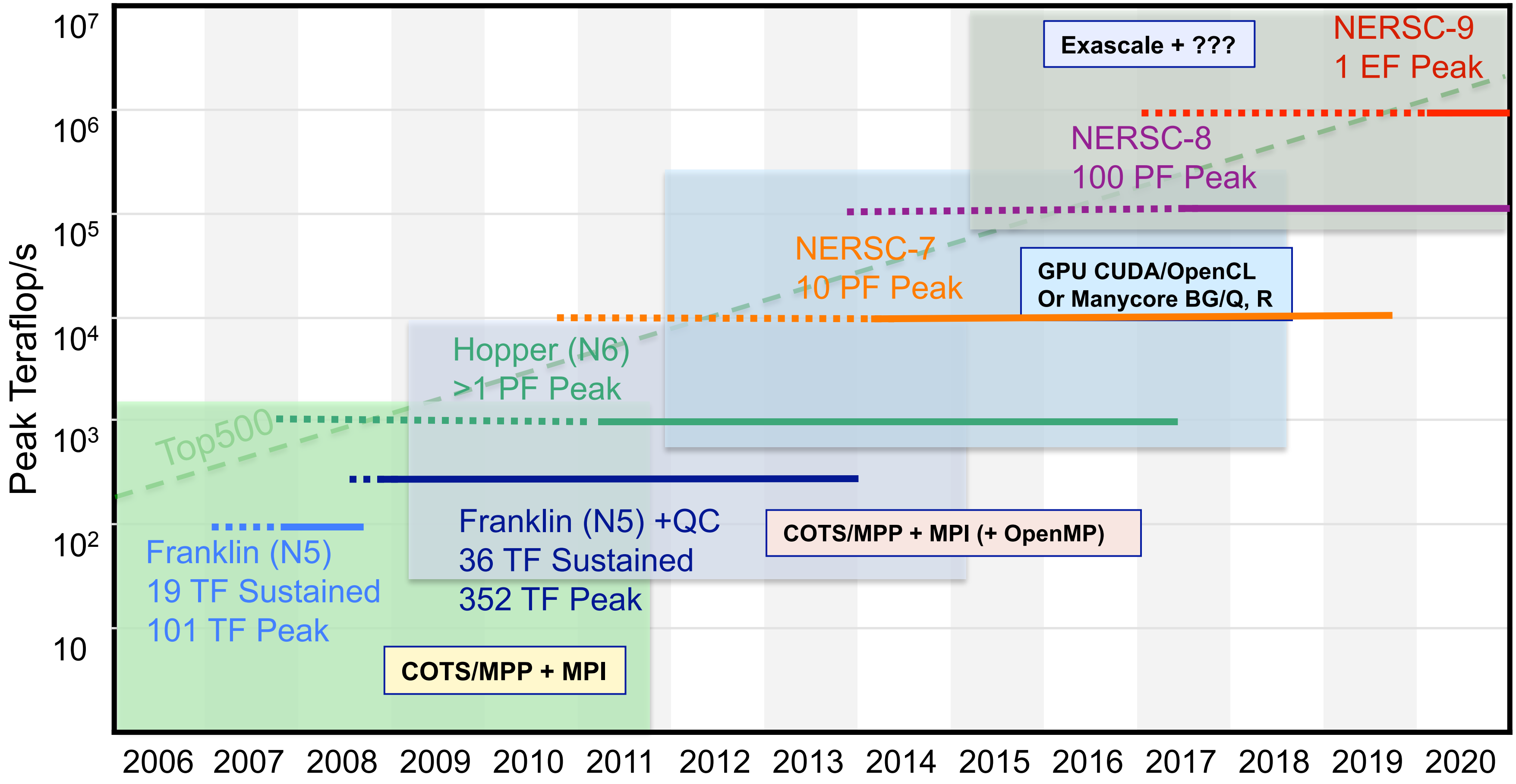




NERSC Benchmarks on Commercial Cloud

Codes	Science Area	Algorithm Space	Configuration	Slow-down	Reduction factor (SSP)	Comments
				Relative to Franklin		
CAM	Climate (BER)	Navier Stokes CFD	200 processors Standard IPCC5 D-Mesh resolution	3.05	0.33	Could not complete 240 proc run due to transient node failures. Some I/O and small messages
MILC	Lattice Gauge Physics (NP)	Conjugate gradient, sparse matrix; FFT	Weak scaled: 14^4 lattice on 8, 32, 64, 128, and 256processors	2.83	0.35	Erratic execution time
IMPACT-T	Accelerator Physics (HEP)	PIC, FFT component	64 processors, 64x128x128 grid and 4M particles	4.55	0.22	PIC portion performs well, but 3D FFT poor due to small message size
MAESTRO	Astrophysics (HEP)	Low Mach Hydro; block structured-grid multiphysics	128 processors for 128^3 computational mesh	5.75	0.17	Small messages and all-reduce for implicit solve.

Increasing computational power should not be ignored—a continued path to exascale exists



Sherwood 2010

Conclusions

- Multicore revolution is changing computing
  - Hybrid?, New Languages?
- New opportunities for fusion
  - Path to exascale is coming
  - Advances in self-consistent simulations requires tackling the next generation hardware
- Effective use of Hopper requires new programming techniques
  - Even simple MPI+OpenMP is complicated
- other technologies for computing are upcoming
  - GPUs
  - Cloulds