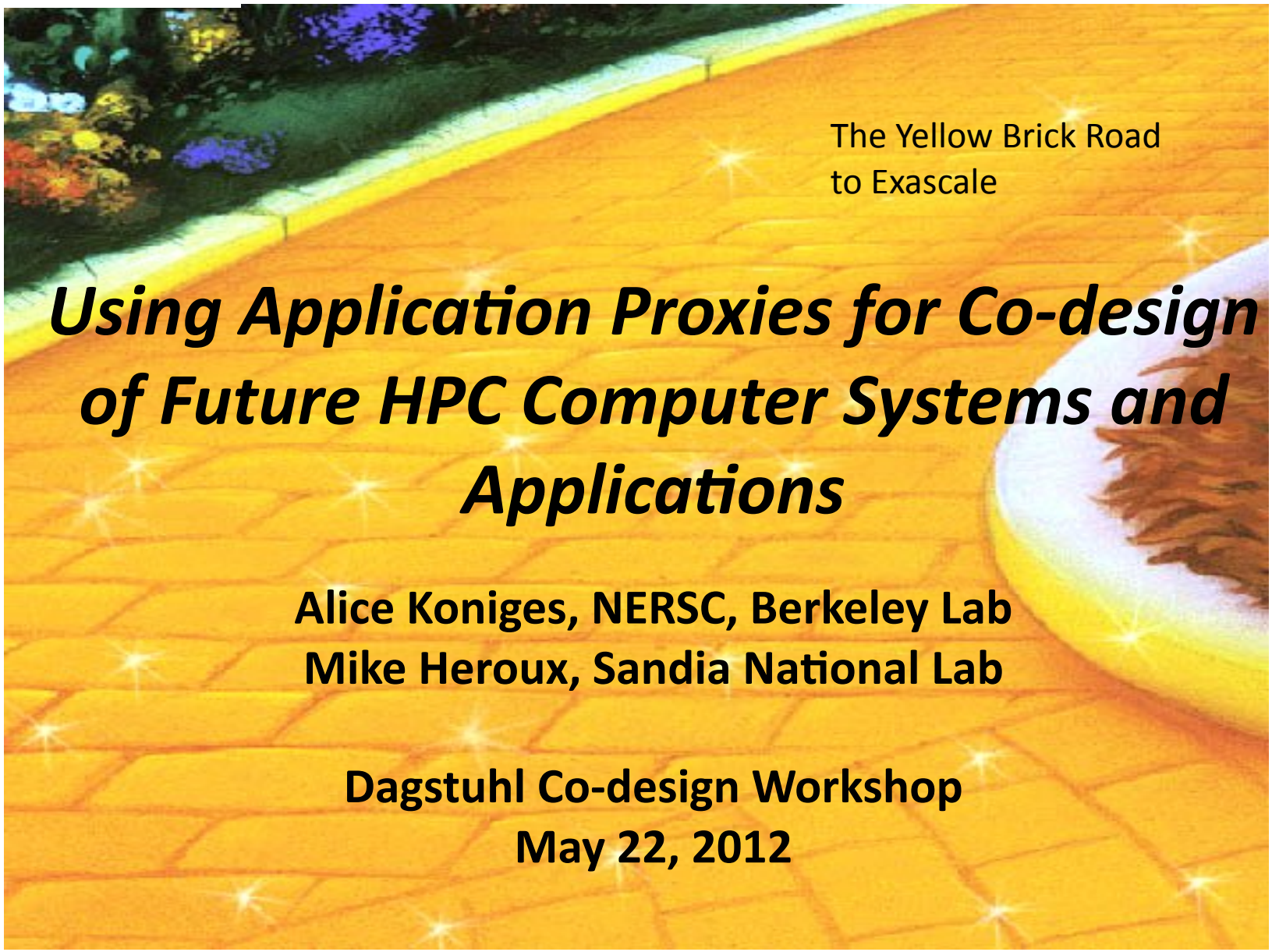


Using Application Proxies for Co-design of Future HPC Computer Systems and Applications

Alice Koniges, NERSC, Berkeley Lab
Mike Heroux, Sandia National Lab

Dagstuhl Co-design Workshop
May 22, 2012

U.S. DEPARTMENT OF ENERGY | NERSC National Energy Research Scientific Computing Center | BERKELEY LAB Lawrence Berkeley National Laboratory | Sandia National Laboratory

We would like to propose a coordinated effort to make compact/mini apps available for co-design

- Co-design based on a single app can lead to the “wrong” machine (e.g., earlier talks)
- Co-design based on compact/mini apps can lead to realistic designs and is more accessible
- These apps are all over, but there is no single repo/testing place to find them (Sandia’s Montevo is a good start)
- Are you interested in an effort (perhaps international) to consolidate these apps?

SPECIAL THANKS

- GTS – Robert Preissl, Petascale Post-doc NERSC
- GPU – Jihan Kim, Petascale Post-doc NERSC
- NPB's – Hongzhang Shan, LBNL, UPC group, others
- PIR3D – Gabriele Jost, TACC, Robins, NWRA
- S3D – John Levesque, CRAY
- Mantevo Project – Heroux, et al., Sandia

We propose to the community that a broad-based international effort of providing and evaluating compact apps and miniapps can give co-design a strong application base. Included in these studies should be linked websites, test cases, and reporting of results on a larger scale.

A Glossary of Application Proxies

- Proxy App: Generic term for all types.
- Skeleton App:
  - Communication accurate, computation fake.
- Compact App:
  - A small version of a real app.
  - Attempting some tie to physics.
- Scalable Synthetic Compact Applications (SSCA):
  - DARPA HPCS.
  - Formal specification.
  - Code and detailed spec to allow re-write.

Glossary (cont).

- HPC Challenge Benchmarks.
- NAS Parallel Benchmarks.
- All the benchmarks used for procurements
- Small versions-
  - Livermore Loops
  - GUPS, and Bandwidth Benchmarks

… And There are More: A crowded space

- UHPC Challenge Problems:
  - Formal specification.
  - Math, kernel extraction.
  - Intended to be open source?
- Motifs, aka dwarves.
  - Really are patterns, not actionable.
  - There is a downloadable Microsoft version

Question: How can we make this more accessible and, more importantly, usable?

How will these apps prepare us for co-design? Are there changes which must be made? What is the best way to disseminate? Are there groups of apps that span international borders, yet are not too loosely connected to design a functional exascale machine?

Miniapps: Specs

- Size: O(1K) lines—is this appropriate?
- Focus: Proxy for key app performance issue.
- Availability: Open Source.
- Scope of allowed change: Any and all.
- Intent: Co-design: From HW registers to app itself.
- Reference version developer & owner: *Application team*.
- Lifespan: *Until it's no longer useful.*

Sandia Mantevo* Project

* Greek: augur, guess, predict, presage

- Multi-faceted application performance project.
- Started 5 years ago.
- Two types of packages:
  - Miniapps: Small, self-contained programs.
    - MiniFE/HPCCG: unstructured implicit FEM/FVM.
    - phdMesh: explicit FEM, contact detection.
    - MiniMD: MD Force computations.
    - MiniXyce: Circuit RC ladder.
    - MiniGhost: Data exchange pattern of CTH.
    - And many more coming online…
  - Minidrivers: Wrappers around Trilinos packages.
    - Beam: Intrepid+FEI+Trilinos solvers.
    - Epetra Benchmark Tests: Core Epetra kernels.
    - Dana Knoll working on new one.
- Open Source (LGPL, moving to New BSD?)
- Staffing: Application & Library developers.

A few examples of using proxy apps to test programming models

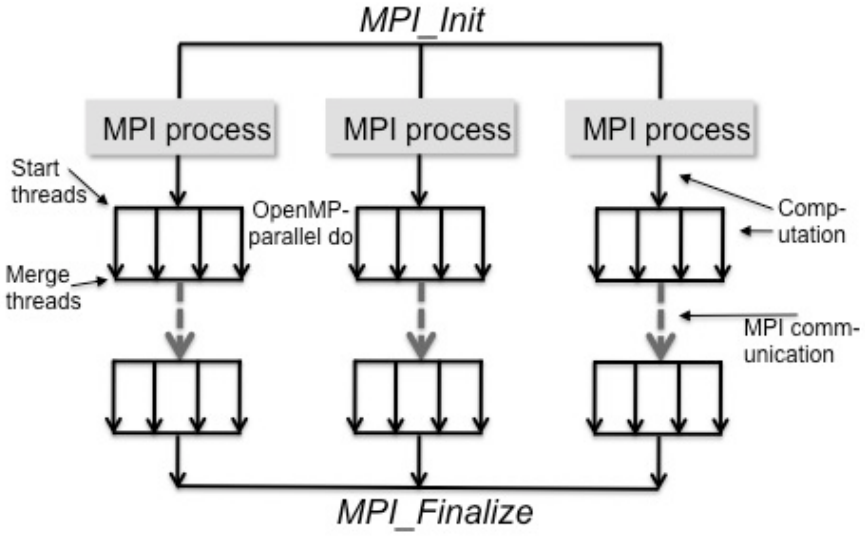
- GTC/S – magnetic fusion particle-in-cell (PIC) code
  - Already optimized and hybrid (MPI + OpenMP)
  - Consider advanced hybrid techniques and PGAS
- GPU Screening of Carbon Capture Materials
  - Optimization for GPU
- PIR3D Three-dimensional Flow Solver
  - Hybridization via expert + application scientist
- NBP's NAS Parallel Benchmarks in various languages
  - Comparison of MPI, Hybrid, and UPC
- S3D – Turbulent Combustion
  - Hybridization for heterogeneous processors

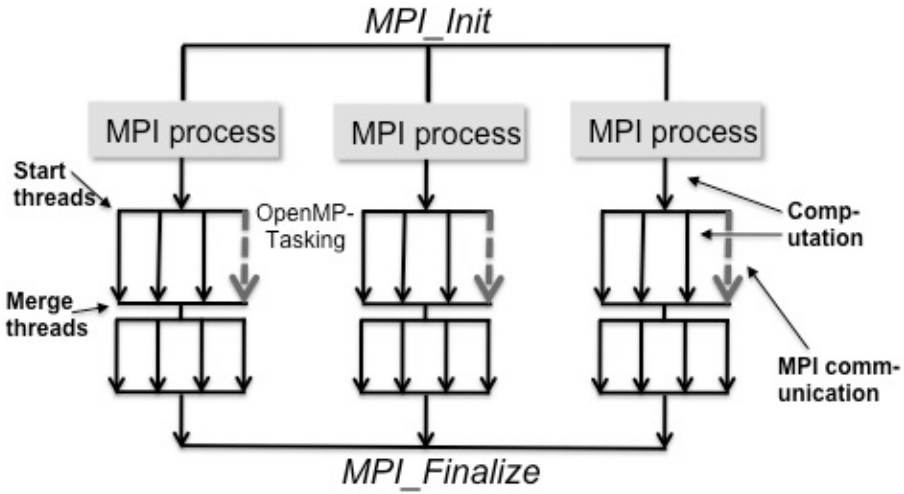
GTS is an optimized PIC magnetic fusion (real) application

- Gyrokinetic Tokamak Simulation (GTS) code
- Global 3D Particle-In-Cell (PIC) code to study microturbulence & transport in magnetically confined fusion plasmas of tokamaks
- Microturbulence: complex, nonlinear phenomenon; key to modeling loss of confinement in tokamaks
- GTS: Highly-optimized Fortran90 (+C) code
- Massively parallel **hybrid** (MPI+OpenMP) parallelization : tested and optimized on multiple platforms

GTS Results from NERSC Petascale Post-doc RobertPreissl with advice from NERSC staff and Cray Research, code from PPPL (Ethier, Wang)

Two different hybrid models in GTS: Using traditional OpenMP worksharing constructs and OpenMP tasks





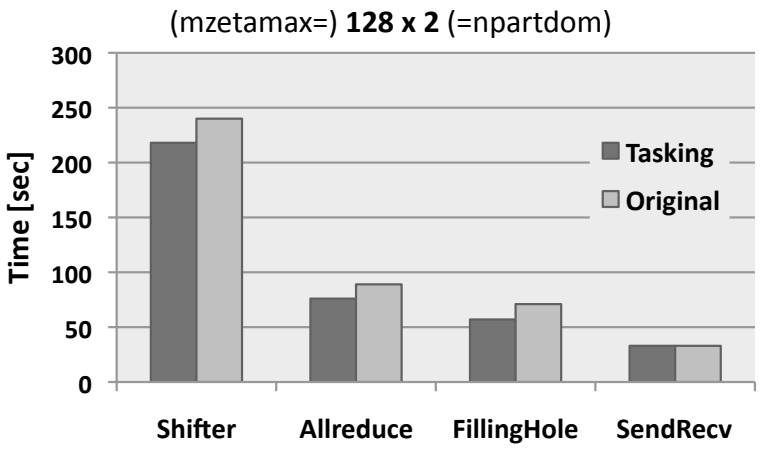
NEW OpenMP Tasking Model gives a new way to achieve more parallelism from hybrid computation.

OpenMP tasks enables us to overlap MPI communication with independent computation and therefore the overall runtime can be reduced by the costs of MPI communication.

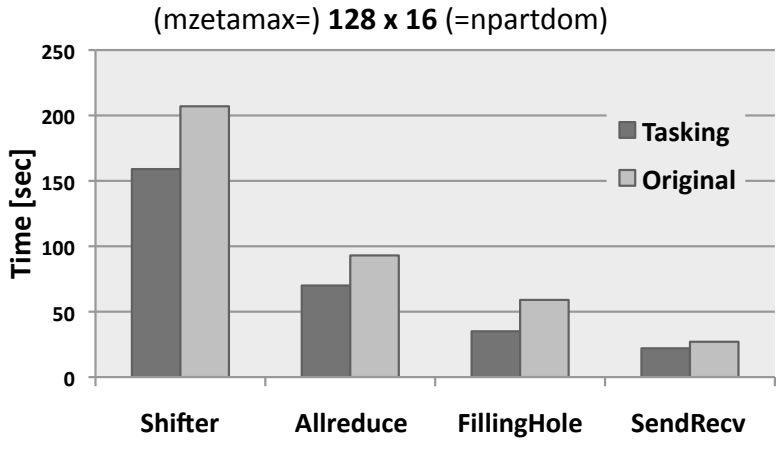
Overlapping Communication with Computation using OpenMP tasks on the GTS magnetic fusion code, R. Preissl, A. Koniges, S. Ethier, W. Wang, N. Wichmann, Journal of Scientific Programming, 2011

OpenMP tasking version outperforms original shifter, especially in larger poloidal domains

256 size run



2048 size run

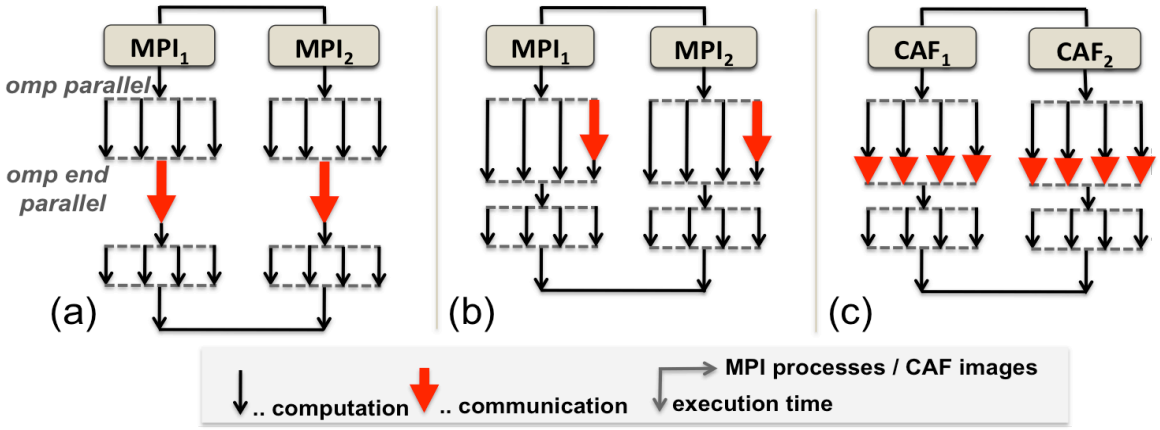


Performance breakdown of GTS shifter routine using 4 OpenMP threads per MPI process with varying domain decomposition and particles per cell on Franklin Cray XT4.

MPI communication in the shift phase uses a **toroidal MPI communicator** (constantly 128)
Large performance differences in the 256 MPI run compared to 2048 MPI run!

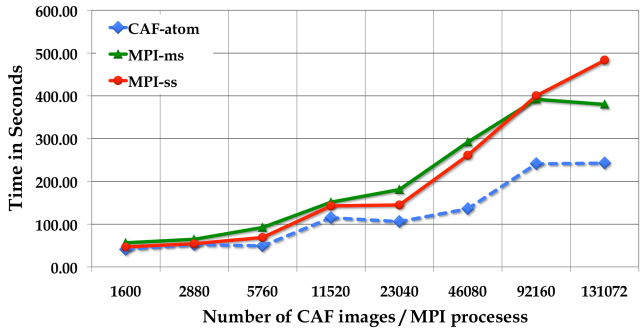
Speed-Up is expected to be higher on larger GTS runs with hundreds of thousands CPUs since MPI communication is more expensive

A new "shifter algorithm" using a combination of MPI, OpenMP, and CAF gives significant performance improvement on 130K cores

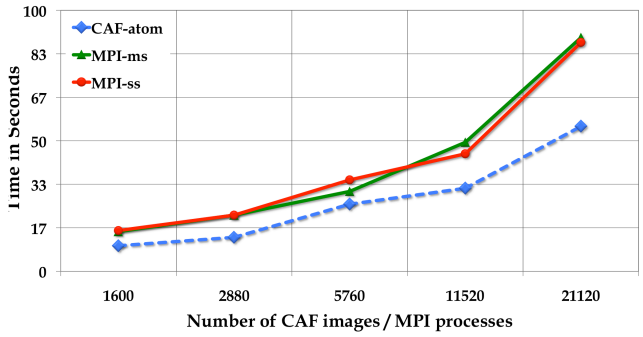


a) Classical hybrid MPI/OpenMP
b) Extension – MPI thread teams for work distribution and collective MPI function calls
c) Hybrid PGAS (CAF) / OpenMP allows ALL OpenMP threads per team to make communication calls to the thread-safe PGAS communication layer

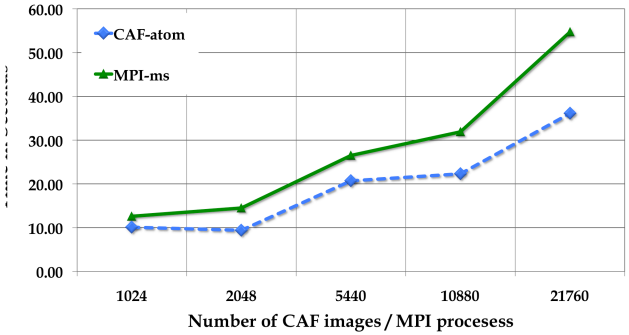
Robert Preissl, Whichman, Long, Shalf, Ethier, Koniges, SC11 Paper



Single-Threaded (Benchmark Suite)



Multi-Threaded (Benchmark Suite)

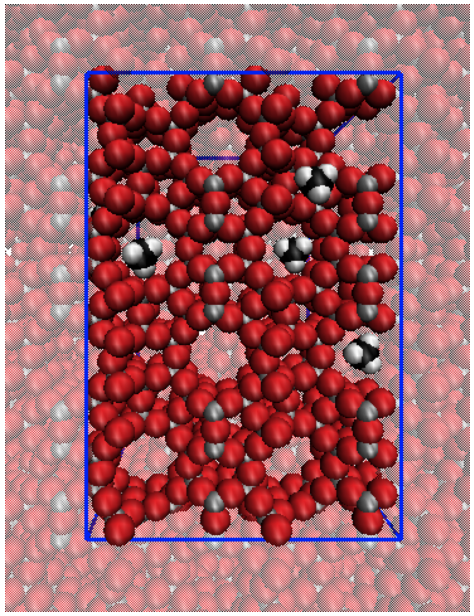


Multi-Threaded (GTS)

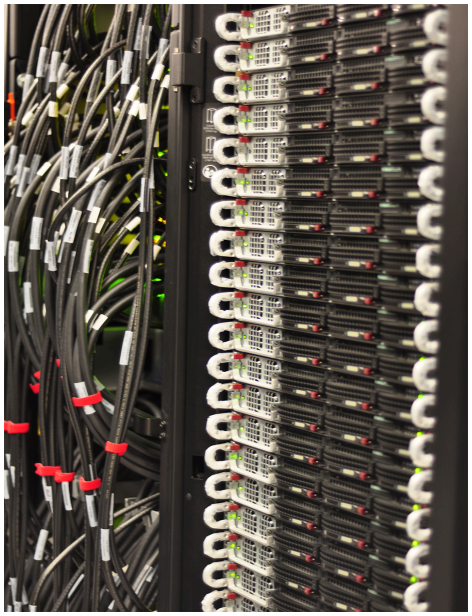
Requires interoperability of MPI, OpenMP, and PGAS; takes advantage of new network design

GPU Monte Carlo Algorithms for Molecules within a Microporous Framework

- Post-doctoral Researcher: Jihan Kim, PIs: Berend Smit, Martin Head-Gordon, LBNL
- The goal of this work is to develop GPU Monte Carlo algorithms to model mobile gases within framework molecules for capture of CO_2.



Graphical illustration of methane molecules (grey-black) inside of a zeolite MFI framework (red). The void spaces within the framework are represented by light circles.



Dirac GPU cluster (rack) at NERSC (44 Tesla C2050 Fermi cards)

GPU Computational Screening of Carbon Capture Materials, J. Kim, A. Koniges, R. Martin, M. Haranczyk, J. Swisher, and B. Smit | Source: SciDAC 2011

GPU Computational Screening of Carbon Capture Materials

Screening of over 5 million materials would have taken many years of CPU-time. GPU code developed has reduced this to a few weeks of CPU-time.

(1) GPU screening code

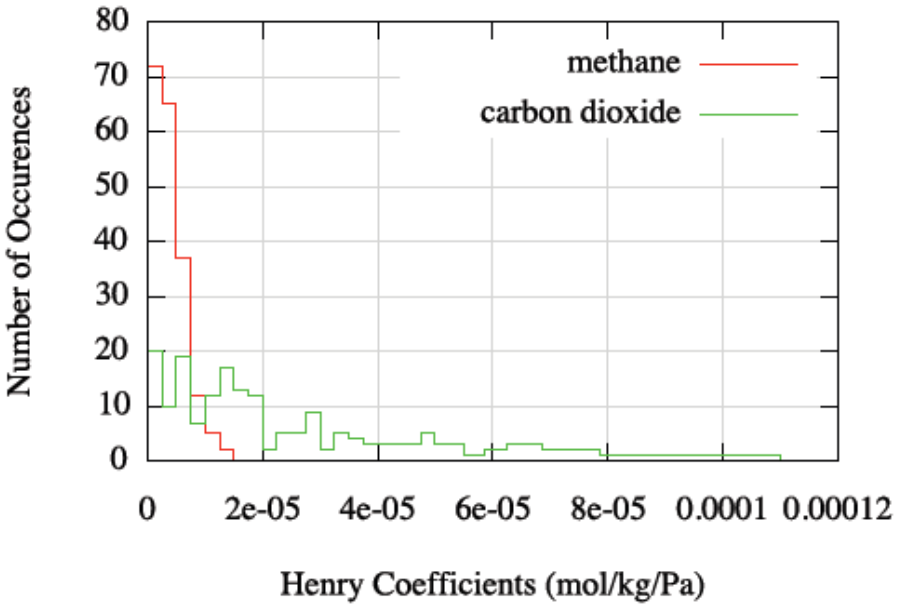
- Characterize and screen a large database of zeolite structures to determine the optimal frameworks for carbon capture
- Computes the Henry coefficient, which characterize selectivity at low pressure

(2) GPU code structure

- Energy grid construction (GPU) bottleneck
- Pore Blocking (CPU)
- Widom Insertion Monte Carlo cycles (GPU)

(3) Performance

- Compute Bound
- Over 50x speedup compared to single CPU core



Histogram of the Henry coefficient distribution for carbon dioxide and methane gases inside 193 IZA zeolite structures

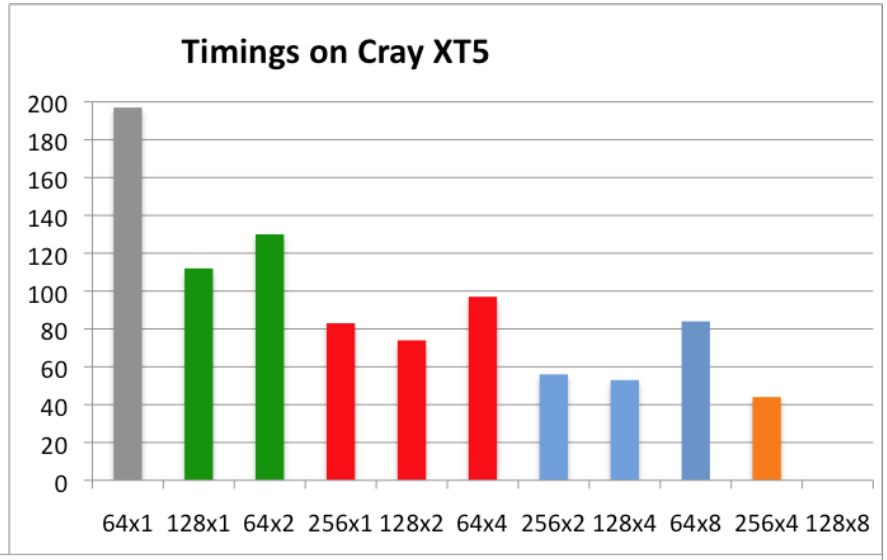
OpenMP Hybridization of PIR3D

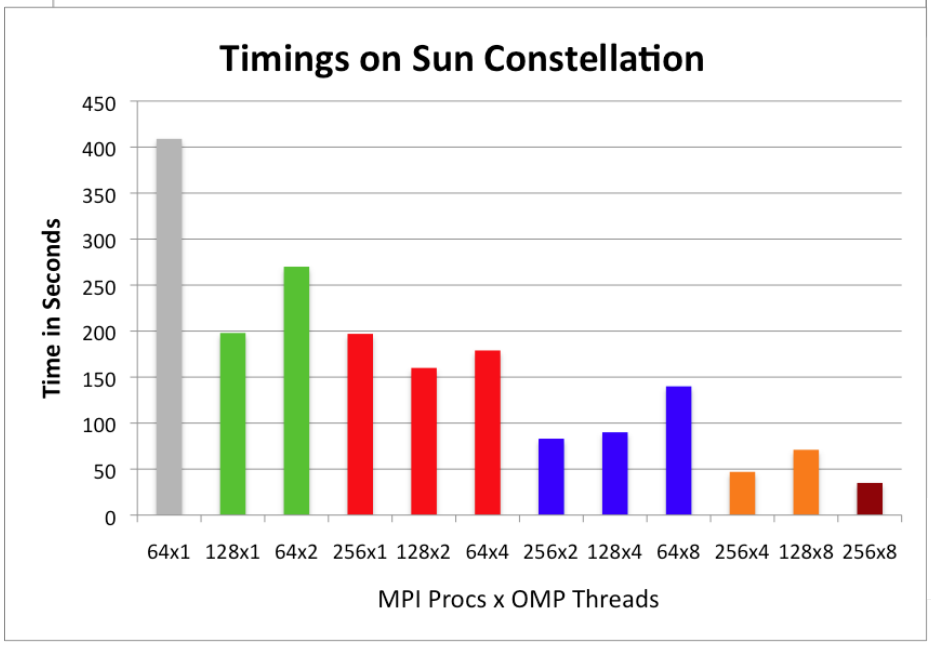
- Motivation:
  - Increase performance by taking advantage of idle cores within one shared memory node
- OpenMP Parallelization strategy:
  - Identify most time consuming routines
  - Place OpenMP directives on the time consuming loops
  - Only place directives on loops across undistributed dimension
  - MPI calls only occur outside of parallel regions: No thread safety is required for MPI library
- Requirements:
  - Thread safe LAPACK and FFTW Routines
  - Note FFTW initialization routine not thread safe: Execute outside

```
DO 2500 IX=1,LOCNX
….
!$omp parallel do private(iy,rvsc)
 DO 2220 IZ=1,NZ
      DO 2220 IY=1,NY
           VYIX(IY,IZ) = YF(IY,IZ)
           VY_X(IZ,IY,IX) = YF(IY,IZ)
           RVSC = RVISC_X(IZ,IY,IX)
           DVY2_X(IZ,IY,IX) =
        DVY2_X(IZ,IY,IX) - (VYIX
(IY,IZ)+VBG(IZ)) * YDF(IY,IZ)
+RVSC*YDDF(IY,IZ)
2220 CONTINUE
!$omp end parallel do
….
2500 CONTINUE
```

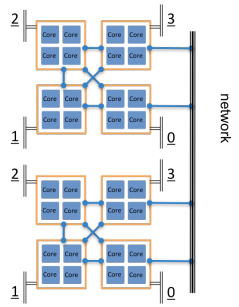
From Parallelization of a 3-D Flow Solver for Multi-Core Node Clusters: Experiences Using Hybrid MPI/OpenMP In the Real World, Gabriele Jost, University of Texas at Austin; Bob Robins, NorthWest Research Associates

All configurations benefit from hybridization, but some more than others: Hybrid Timings for Case 512x256x256

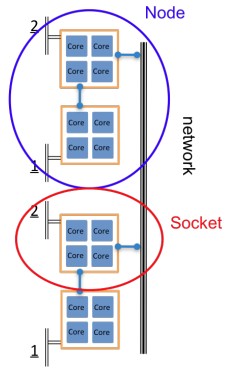




- Use all 4 cores/per socket
- Benefits of OpenMP:
- Increase the number of usable cores
- 128x2 outperforms 256x1 on 256 cores, 128x4 better than 256x2 on 512 cores

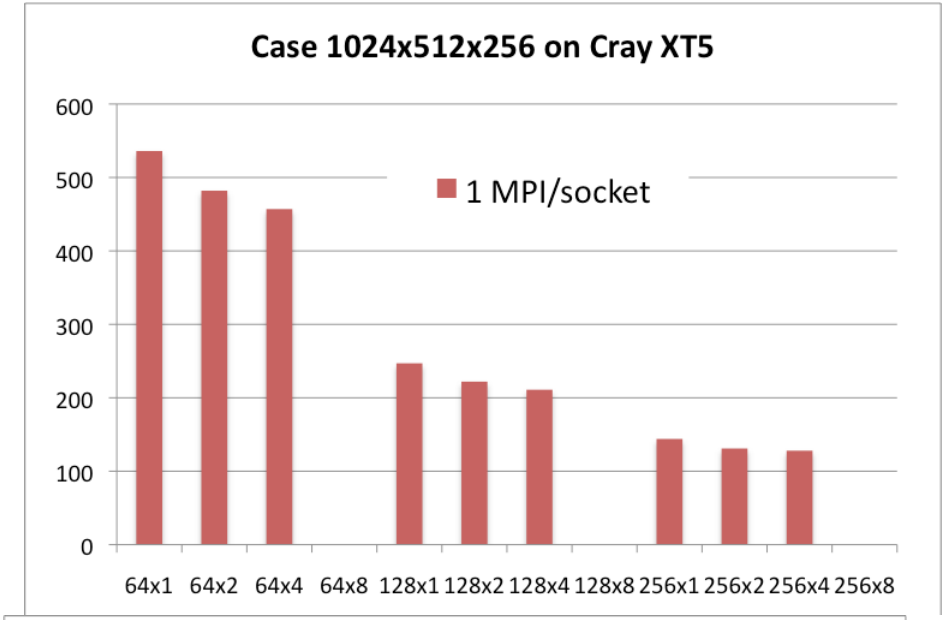


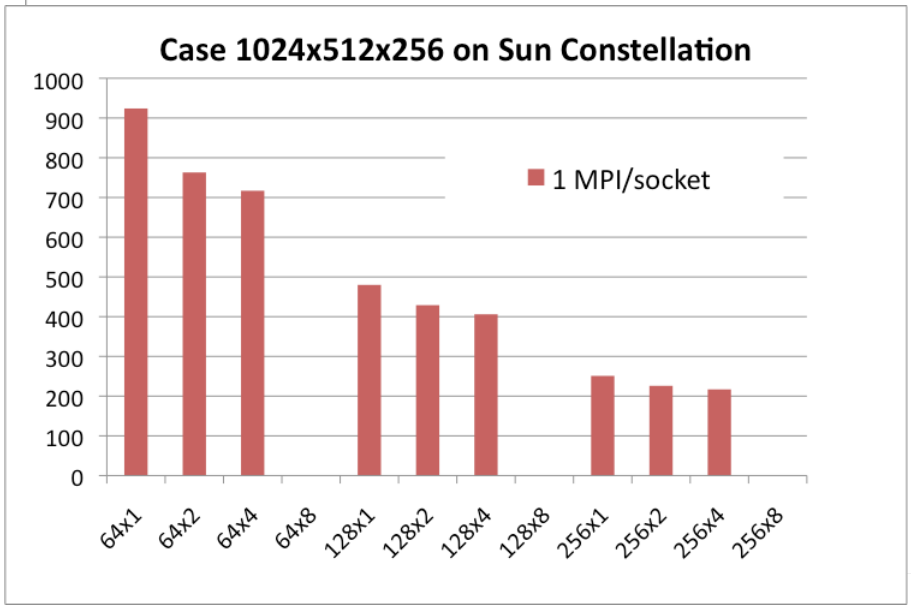
But: Most of the performance due to "spacing" of MPI. About 12% improvement due to OpenMP



TACC NWRA

Hybrid Timings for Case 1024x512x256





- Only 1 MPI Process per socket due to memory consumption
- 14%-10% performance increase on Cray XT5
- 13% to 22% performance increase on Sun Constellation

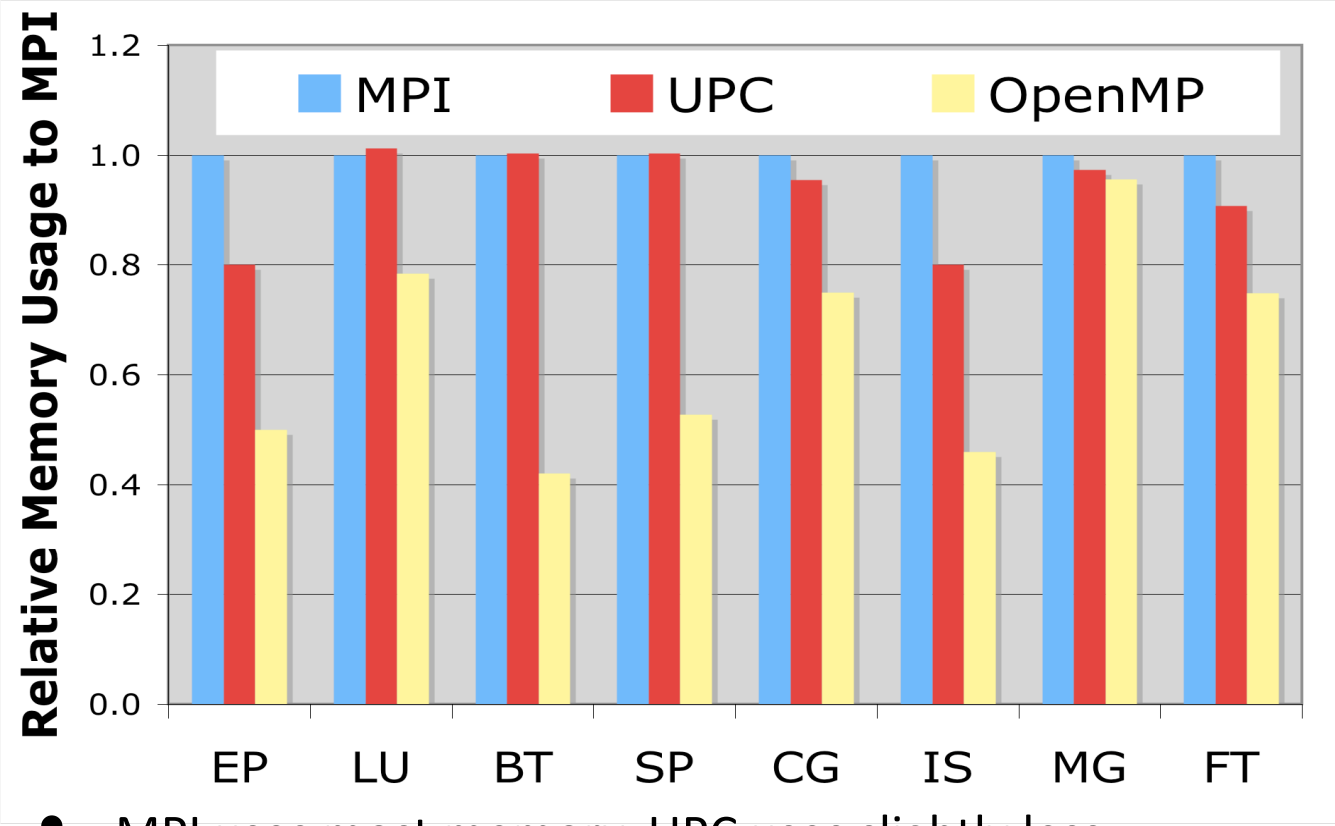
Take-Away: Expert Hybrid Programming done with extra care gives performance improvements

TACC NWRA

The eight NAS parallel benchmarks (NPBs) have been written in various languages including hybrid for three

MG	Multigrid	Approximate the solution to a three-dimensional discrete Poisson equation using the V-cycle multigrid method	
CG	Conjugate Gradient	Estimate smallest eigenvalue of sparse SPD matrix using the inverse iteration with the conjugate gradient method	
FT	Fast Fourier Transform	Solve a three-dimensional PDE using the fast Fourier transform (FFT)	
IS	Integer Sort	Sort small integers using the bucket sort algorithm	
EP	Embarrassingly Parallel	Generate independent Gaussian random variates using the Marsaglia polar method	
BT SP LU	Block Tridiagonal Scalar Pentadiag Lower/Upper	Solve a system of PDEs using 3 different algorithms	MZ

Significant improvement in Memory Usage is possible for new hardware—Comparison of OpenMP, UPC, MPI using NPB's

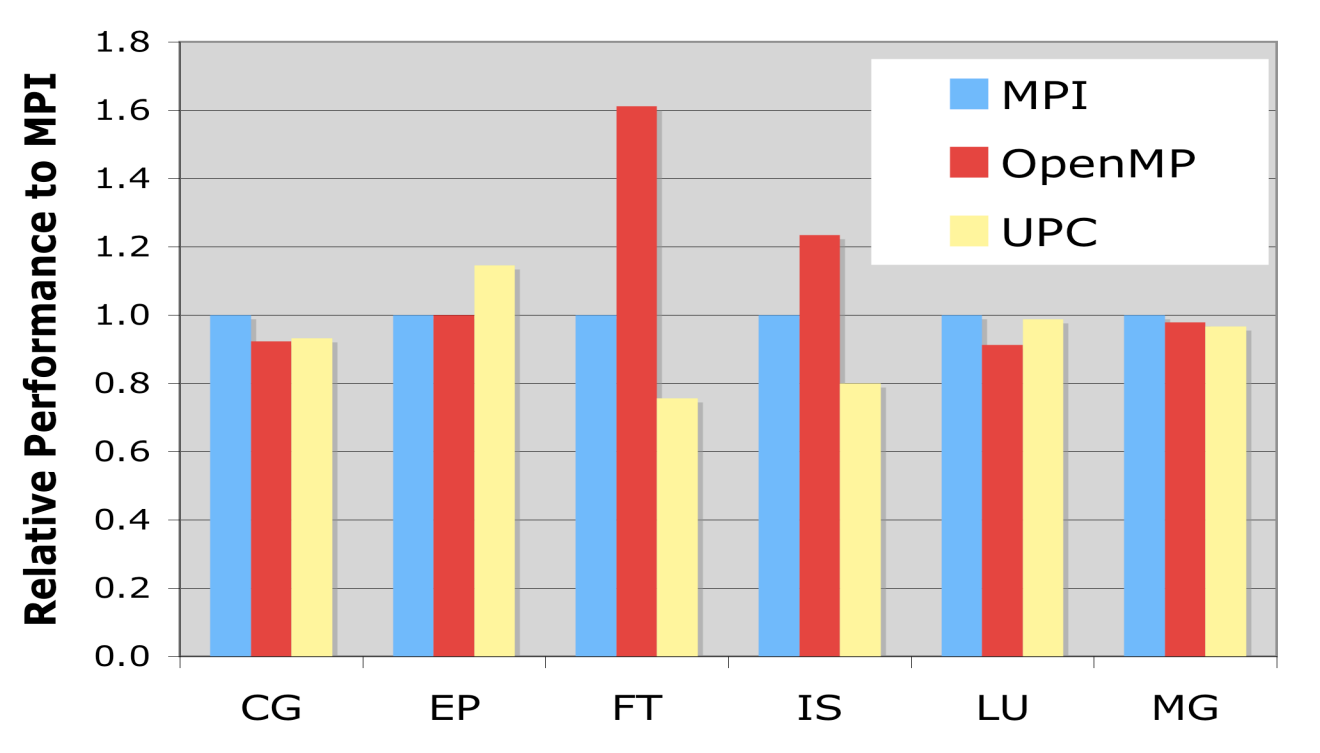


- MPI uses most memory, UPC uses slightly less
- OpenMP savings due to direct data access

"A programming model performance study using the NAS parallel benchmarks," Hongzhang Shan, Filip Blagojevic, Seung-Jai Min, Paul Hargrove, Haoqiang Jin, Karl Fürlinger, Alice Koniges, Nicholas J. Wright **Scientific Programming 18, 153-167 (2010)**

Experiments by H. Shan, LBNL; UPC codes from Berkeley UPC group and NAS

Performance properties of languages is an evolving issue as compilers and techniques improve



- Similar performance for CG, EP, LU, MG
- For FT, IS, OpenMP delivers significantly better performance due to efficient programming

Cray Study: OpenMP accelerator extensions yield promise for ease of transition to heterogeneous architectures

- OpenMP accelerator extensions
  - Are the directives powerful enough to allow the developer to pass information on to the compiler
  - Can the compiler generate code that get performance close to Cuda
- Application of this method to S3D combustion code
  - Part of Cray Center of Excellence, John Levesque
  - Requires a combination of programmer intervention for code re-arrangement and automatic

S3D yields performance and portability

	Original OpenMP 12 cores best out of 24	Cuda Fortran	Directive Approach	Directive Approach Restructured
Kernel Only	.0417 Seconds	.0061 Seconds	.0113 Seconds	.0067 Seconds

•In S3D all of the arrays used in this computation will reside on the accelerator prior to the invocation of the kernel.

Take away: This appeals to application programmers – one version of code for multiple machines. Again, human intervention is required.

Courtesy John Leveque, Cray Center of Excellence

Questions for Discussion

- Can we have many of these apps in one place
- Can we use a system, e.g. NERSC science gateways that will allow people to seamlessly run the apps in small versions on parallel machines?
- Will this help to unify performance comparisons, provide vendors, and the others with a sandbox
- How big/small should these apps be?
- Should we include a system to run them?
- Do you want to join us?