Challenges and Application Gems on the Path to Exascale

Alice Koniges, NERSC, Berkeley Lab

ASCR Programming Challenges Workshop
July 2011

Path to Exascale: some examples, their discovered "gems," and implications for programming models

- GTS – 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
- fvCAM Climate Benchmark
  - Hybrid for reducing memory
- S3D – Turbulent Combustion
  - Hybridization for heterogeneous processors

SPECIAL THANKS!

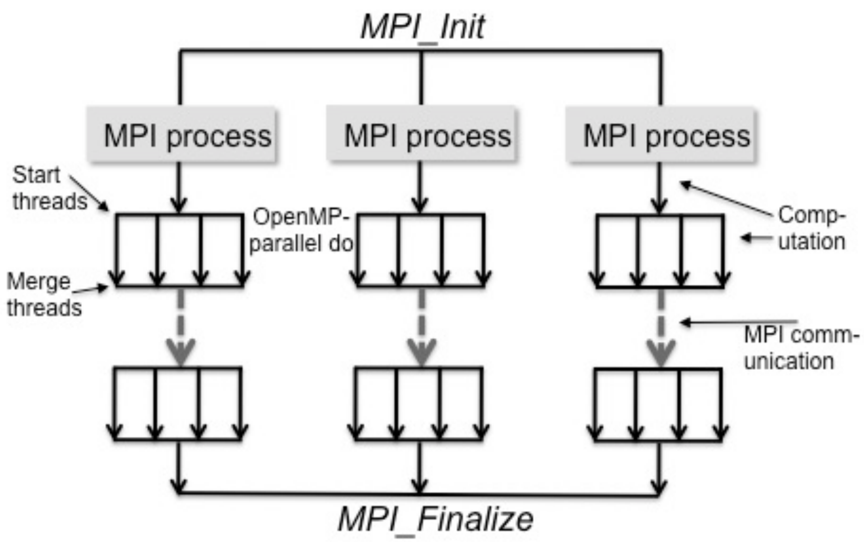
- GTS – Robert Preissl, Petascale Post-doc NERSC
- GPU – Jihan Kim, Petascale Post-doc NERSC
- PIR3D – Gabriele Jost, TACC, Robins, NWRA
- NPB's – Hongzhang Shan, LBNL
- fvCAM– Nick Wright, NERSC
- S3D – John Levesque, CRAY
- General: Kathy Yelick, John Shalf

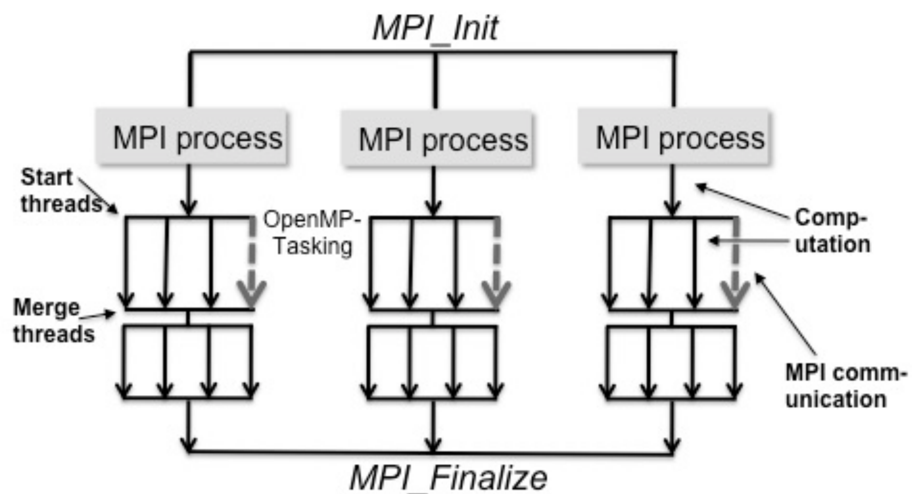
GTS is an optimized PIC magnetic fusion (real) application

- 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 Robert Preissl with advice from NERSC staff and Cray Research, code from PPPL (Ethier, Wang)

Gem: 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 subteams might also be a way to do this.*

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.

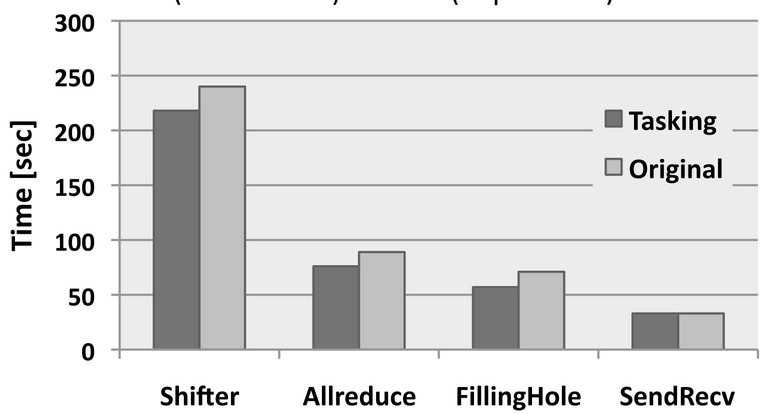
Take-Away: This experience supports the development of advanced capabilities for OpenMP

*Barbara M. Chapman, Lei Huang, Haoqiang Jin, Gabriele Jost, and Bronis R. de Supinski: Toward Enhancing OpenMP's Work-Sharing Directives. In proceedings, W.E. Nagel et al. (Eds.): Euro-Par 2006, LNCS 4128, pp. 645-654, 2006.

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

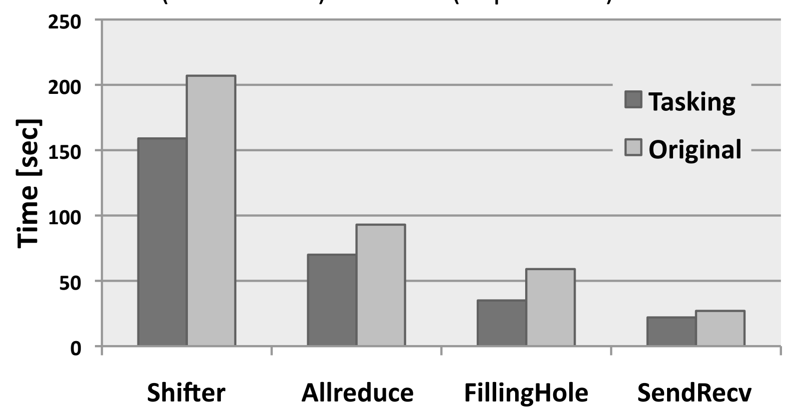
256 size run

(mzetamax=) **128 x 2** (=npartdom)



2048 size run

(mzetamax=) **128 x 16** (=npartdom)

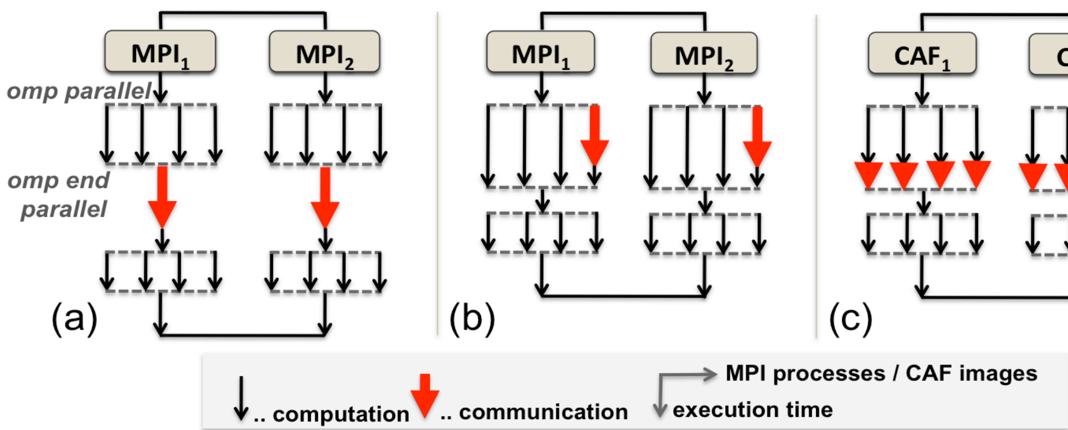


Performance breakdown of GTS shifter routine using 4 OpenMP threads per MPI process with varying domain decomposition and particles per cell on Franklin Cray XT4.

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!

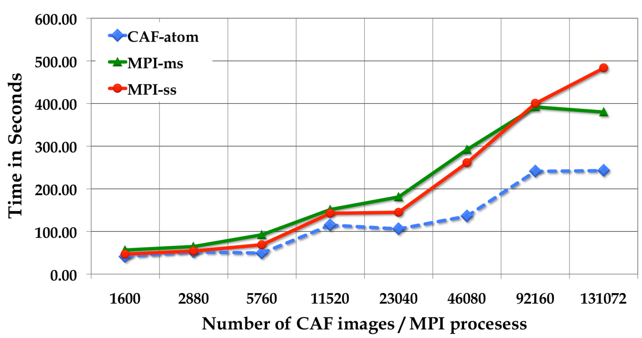
Speed-Up is expected to be higher on larger GTS runs with hundreds of thousands CPUs since MPI communication is more expensive

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

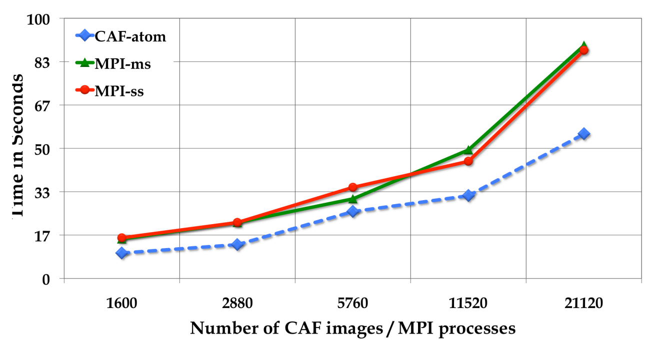


a) Classical hybrid MPI/OpenMP
b) Extension – MPI thread teams for work distribution and collective MPI function calls
c) Hybrid PGAS (CAF) / OpenMP allows ALL OpenMP threads per team to make communication calls to the thread-safe PGAS communication layer

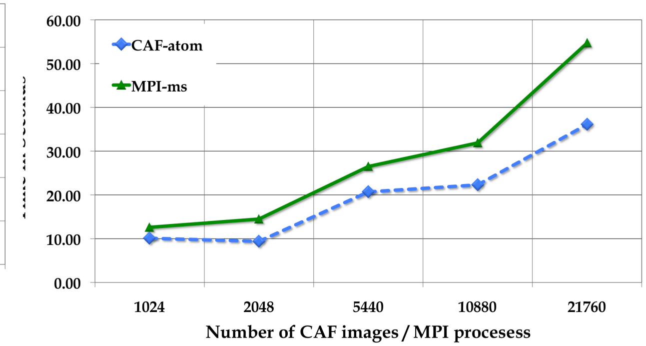
Robert Preissl, Whichman, Long, Shalf, Ethier, Koniges, SC11 Best Paper Nominee



Single-Threaded (Benchmark Suite)



Multi-Threaded (Benchmark Suite)

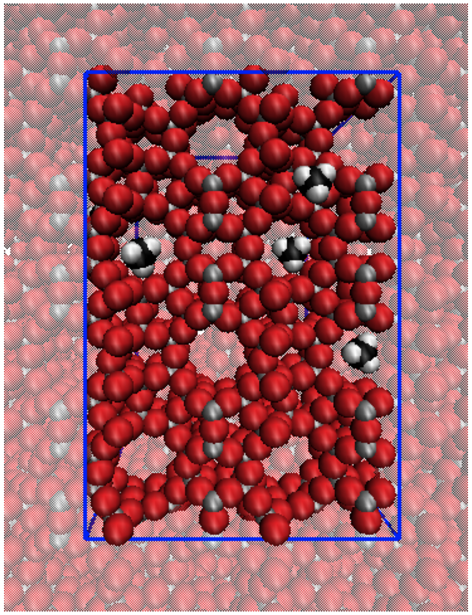


Multi-Threaded (GTS)

Take-Away: Requires interoperability of MPI, OpenMP, and PGAS

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

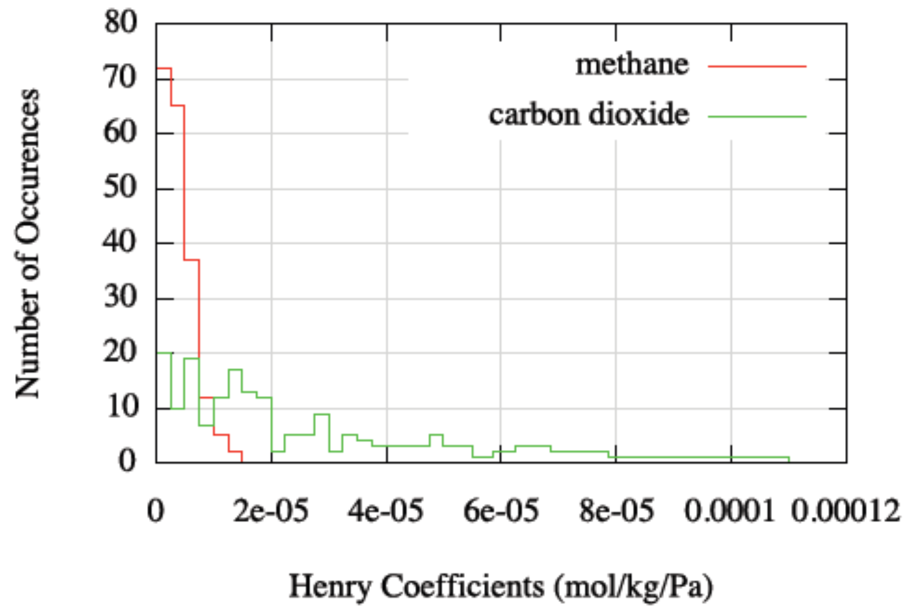
GEM: Screening of over 5 million materials would have taken many years of CPU-time. GPU code developed has reduced this to a few weeks of CPU-time.

Take-Away: Can you beat this? Certain applications are "rocking" with GPUs

(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

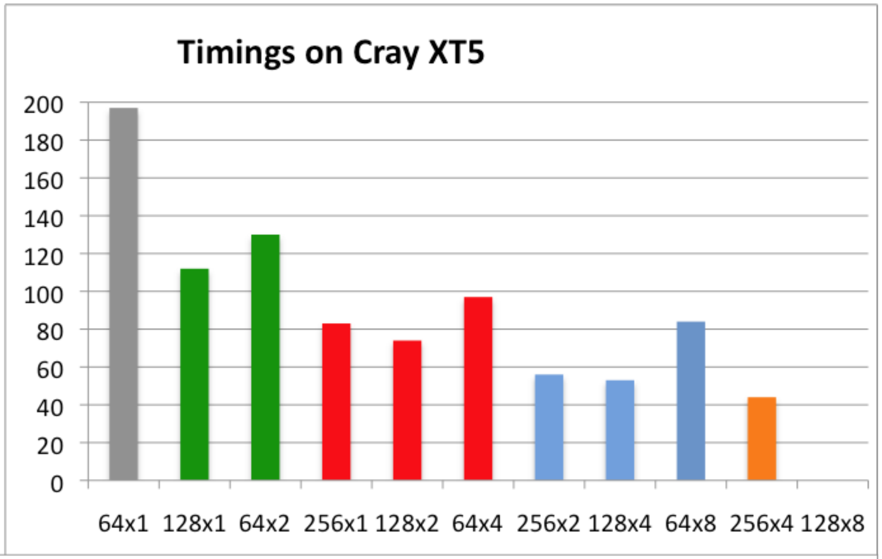
OpenMP Hybridization of PIR3D

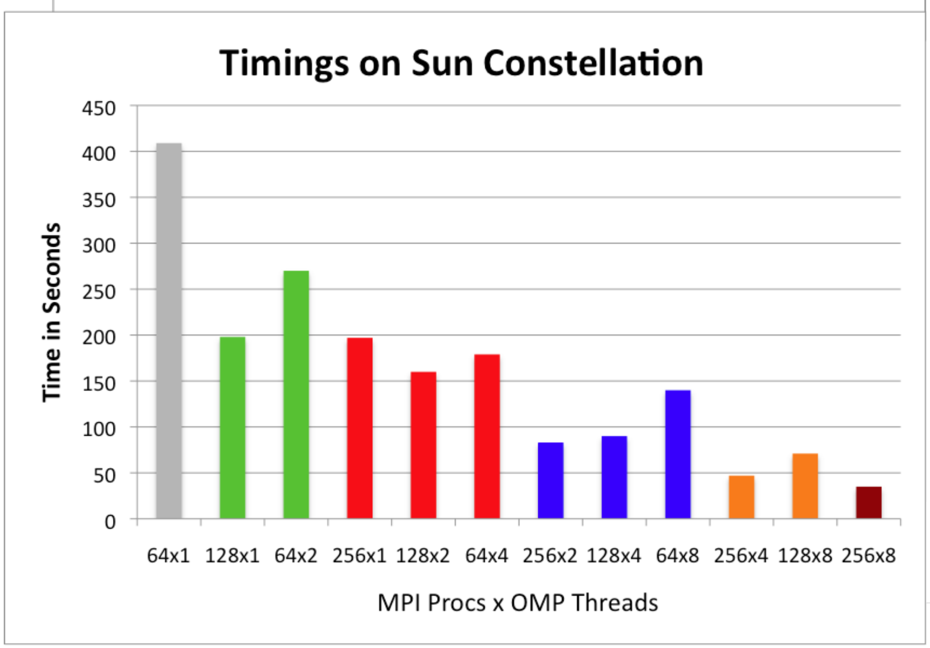
- Motivation:
    - Increase performance by taking advantage of idle cores within one shared memory node
- OpenMP Parallelization strategy:
    - Identify most time consuming routines
    - Place OpenMP directives on the time consuming loops
    - Only place directives on loops across undistributed dimension
    - 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
```

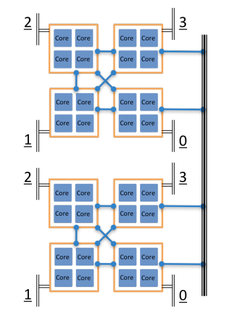
From Parallelization of a 3-D Flow Solver for Multi-Core Node Clusters: Experiences Using Hybrid MPI/OpenMP In the Real World, Gabriele Jost, University of Texas at Austin; Bob Robins, NorthWest Research Associates

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

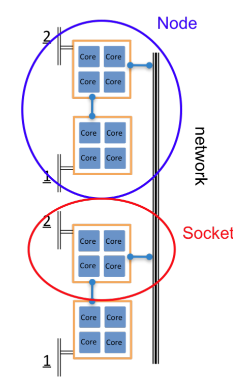




- Use all 4 cores/per socket
- Benefits of OpenMP:
- Increase the number of usable cores
- 128x2 outperforms 256x1 on 256 cores, 128x4 better than 256x2 on 512 cores

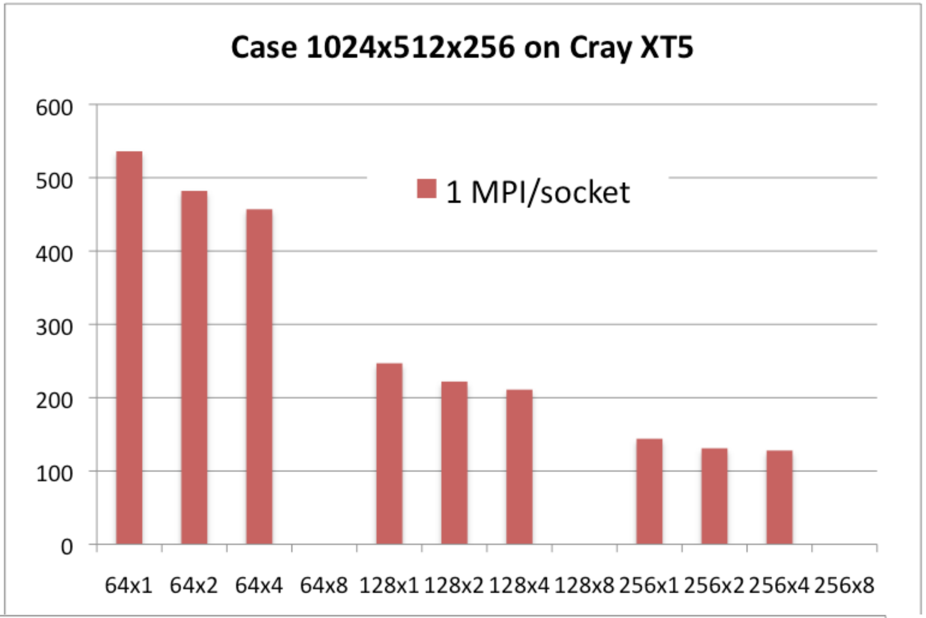


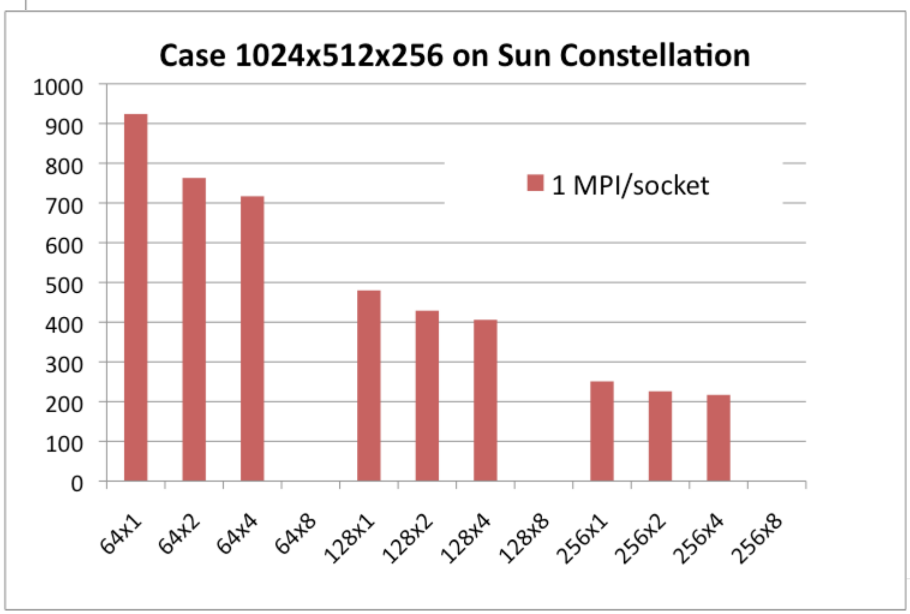
But: Most of the performance due to "spacing" of MPI. About 12% improvement due to OpenMP



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

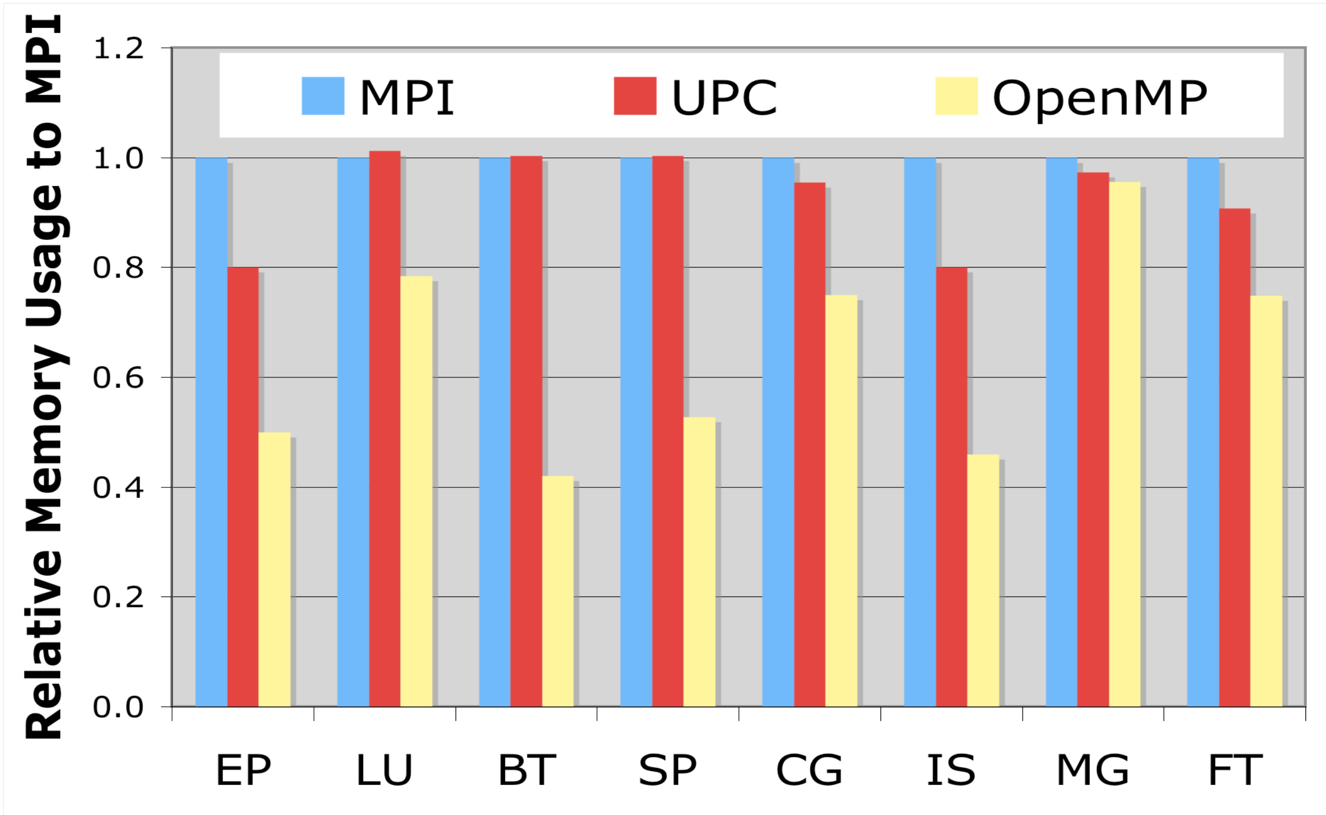
Take-Away: Expert Hybrid Programming done with extra care gives performance improvements

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

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

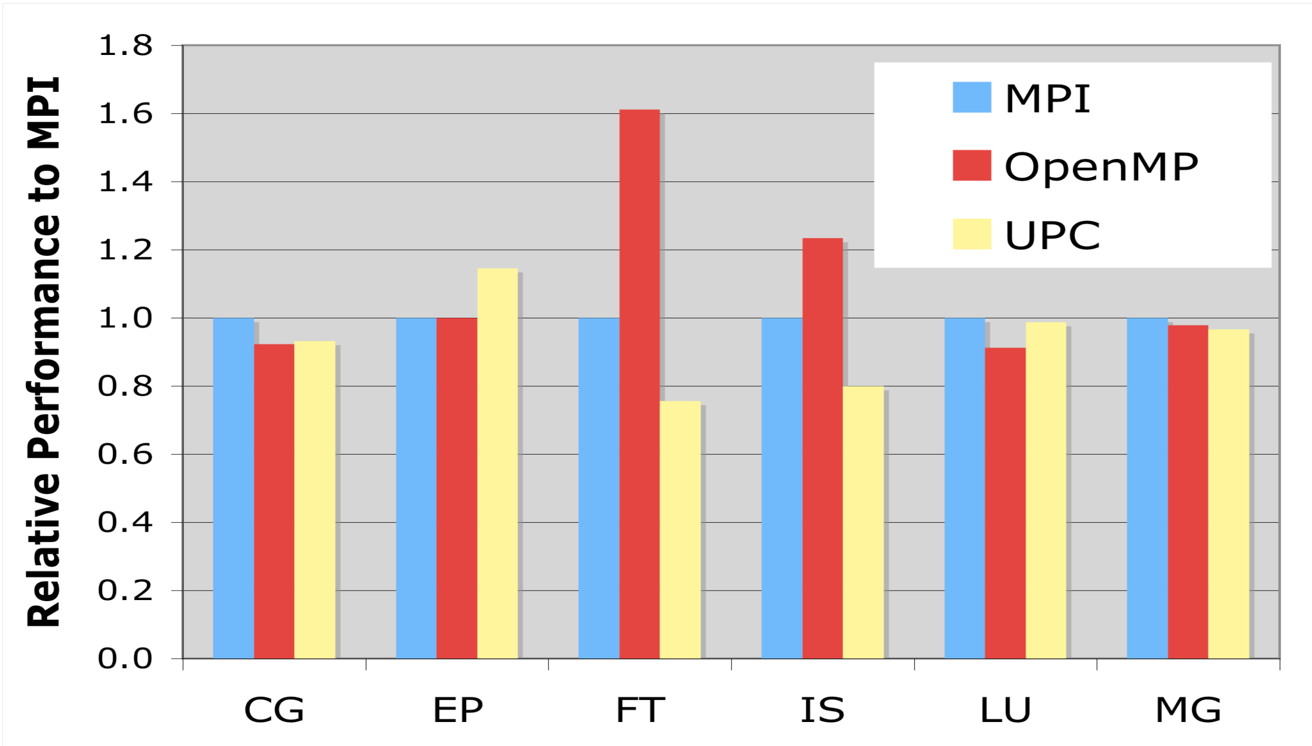


- MPI uses most memory, UPC uses slightly less
- OpenMP savings due to direct data access

Take-Away: We need more examples such as these
In different languages to spur development

Experiments by H. Shan, LBNL

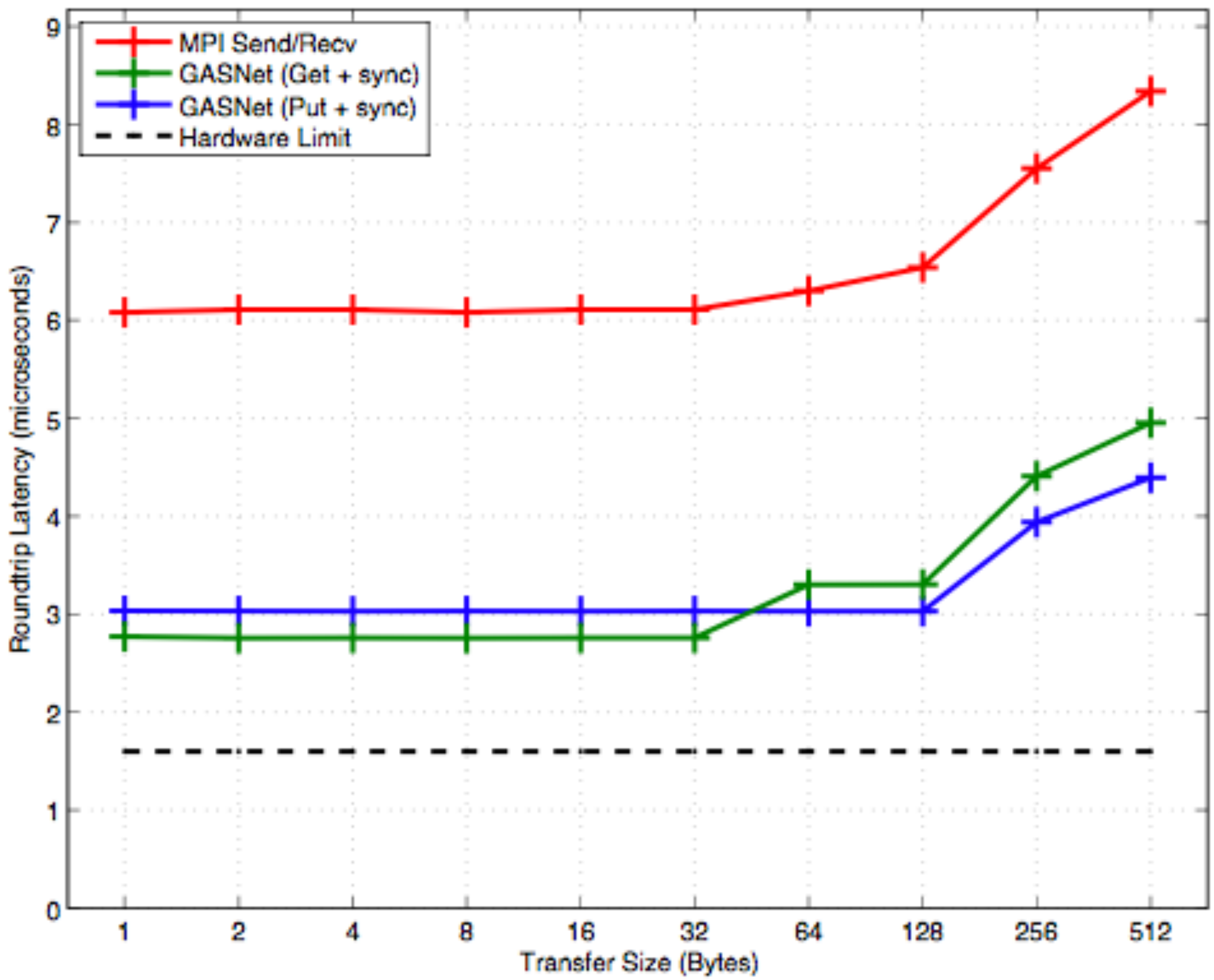
Performance properties of languages is an evolving issue as compilers and techniques improve



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

Correct implementation is required for performance

GASNet vs MPI Latency on BG/P



GASNet support for XE6 makes a critical difference

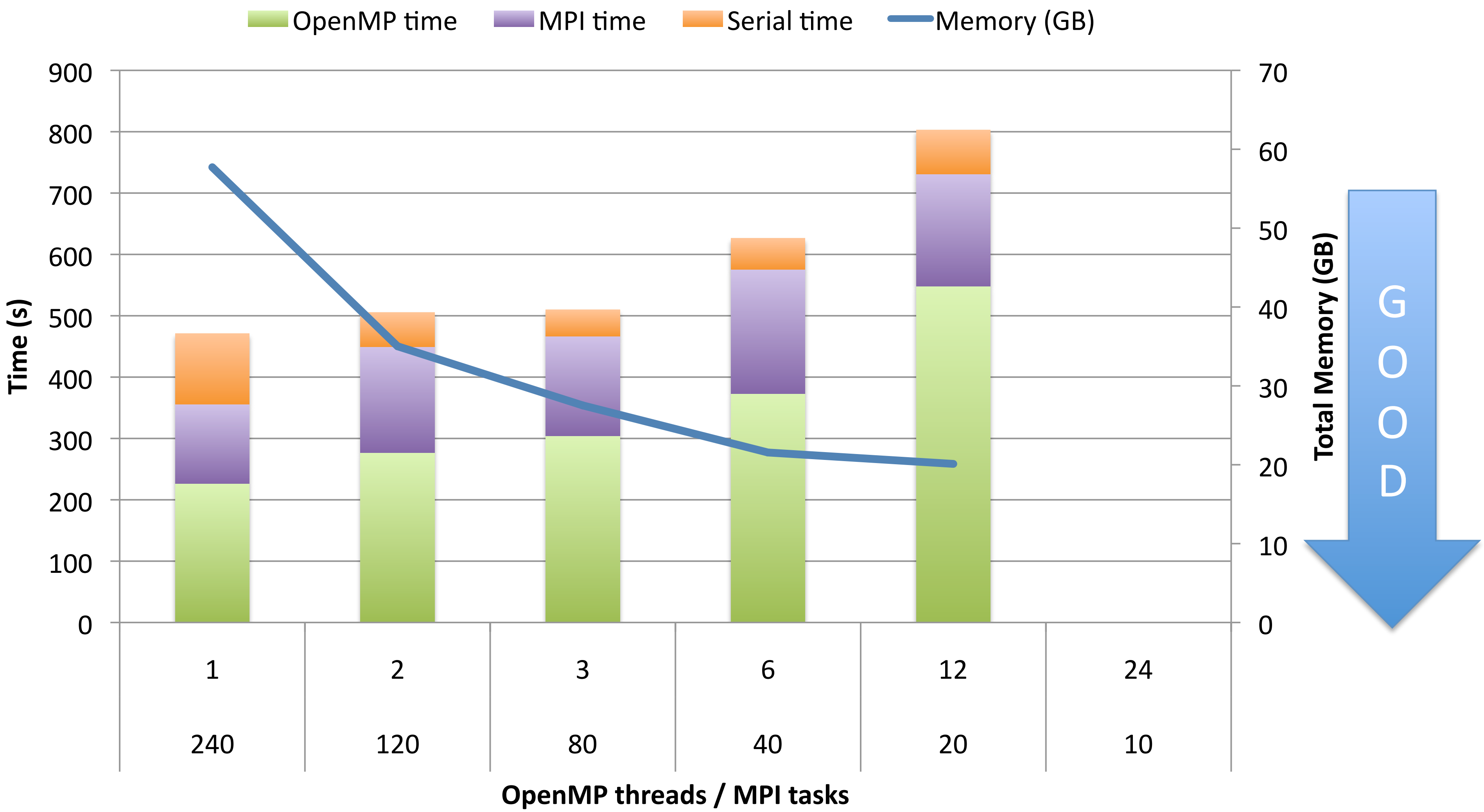
GEM: Significant improvement of GASNet over Gemini to GASNet over MPI conduit on Cray XE6

- AMMedium latency
  - Gemini-conduit - 1.25 uSec
  - MPI-conduit - 1.9 uSec
- AMLong throughput
  - Gemini-conduit - ~5 GB/sec at 64KB message size
  - MPI-conduit - ~ 2 GB/sec at 512 KB message size
- Put
  - Gemini-conduit - 1.25 uSec
  - MPI-conduit - 4 uSec
- Get
  - Gemini-conduit - 2.2 uSec
  - MPI-conduit - 4 uSec

Take-Away: To get good performance from a language, we must have the appropriate infrastructure/implementations

From: The Berkeley GASNet and UPC Group, Lawrence Stewart Serissa Research

Memory (only) Gem: Hybrid Performance of Climate Code



NERSC Hopper from Cray Center of Excellence
Nick Wright, Marcus Wagner, Tony Drummond, John Shalf

Take away: Sometimes performance is better, sometimes not. Almost always memory reduction. Hybrid performance not automatic – requires human.

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

GEM: 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

Current Application Path-Hybrid MPI/OpenMP Codes are becoming increasingly prevalent; is this enough?

- OpenMP/MPI Hybrid Codes provide these opportunities:
  - Lower communication overhead
  - Lower memory requirements
  - Provide for flexible load-balancing on coarse and fine grain
  - Increase parallelism
- PGAS
  - Can use customized communication to avoid hot-spots
    - But UPC Collectives do not support some communication patterns
  - Benefits of message aggregation depends on the arch./interconnect
  - UPC – Shared Memory Programming
    - Less communication with reduced memory utilization
  - Mapping BUPC language-level threads to Pthreads and/or Processes
    - Mix of processes and pthreads often gives the best performance
- Other Models must answer – Can You Beat This?
  - Concurrent Collections? Automatic tools for CUDA? New parallelism models and languages?