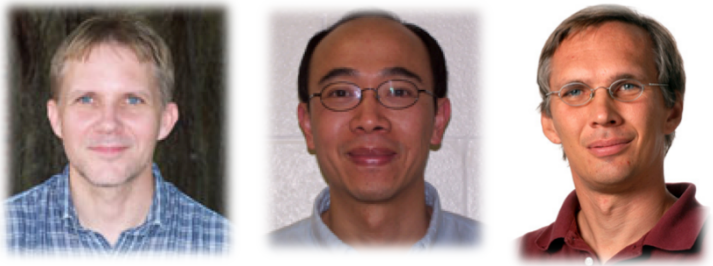
Co-Design of Hardware/Software for Predicting MAV Aerodynamics

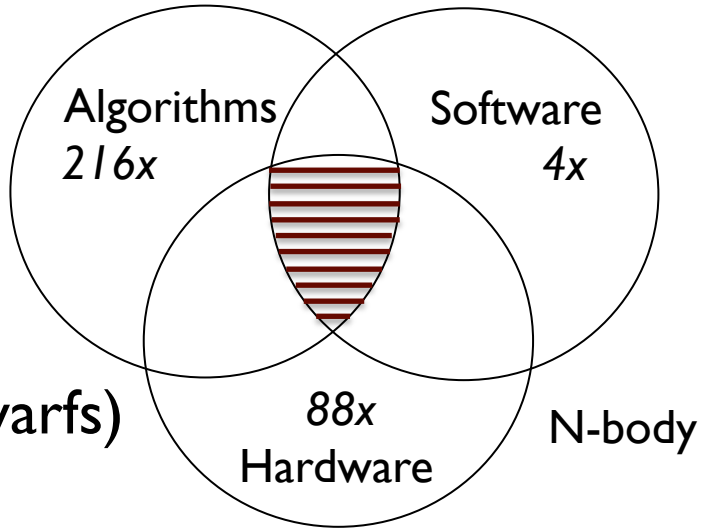
E. de Sturler, **W. Feng**, C. Roy, A. Sandu, D. Tafti



J. Edwards, H. Luo, F. Mueller

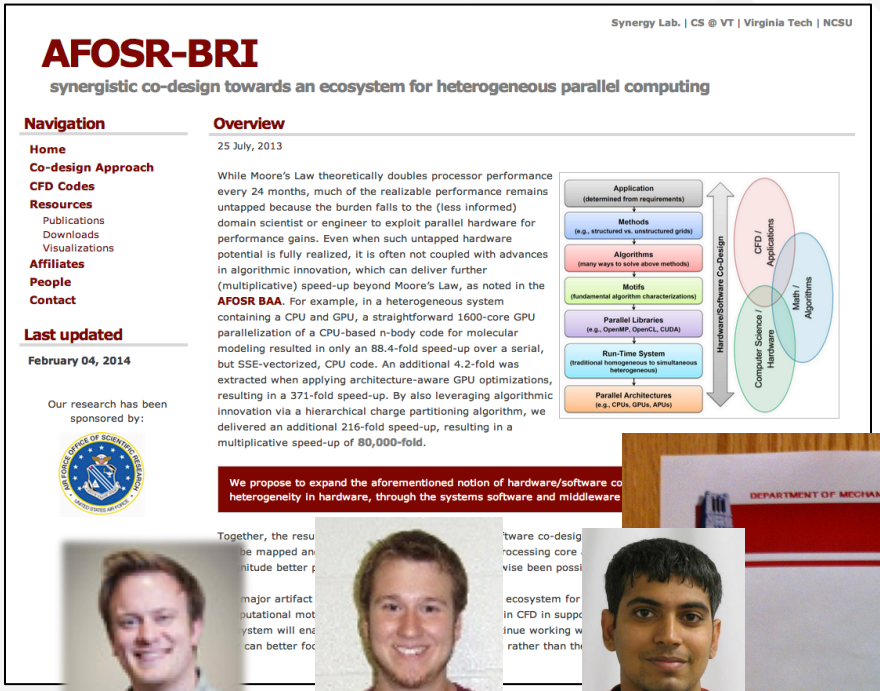
Fariba Fahroo, Computational Mathematics

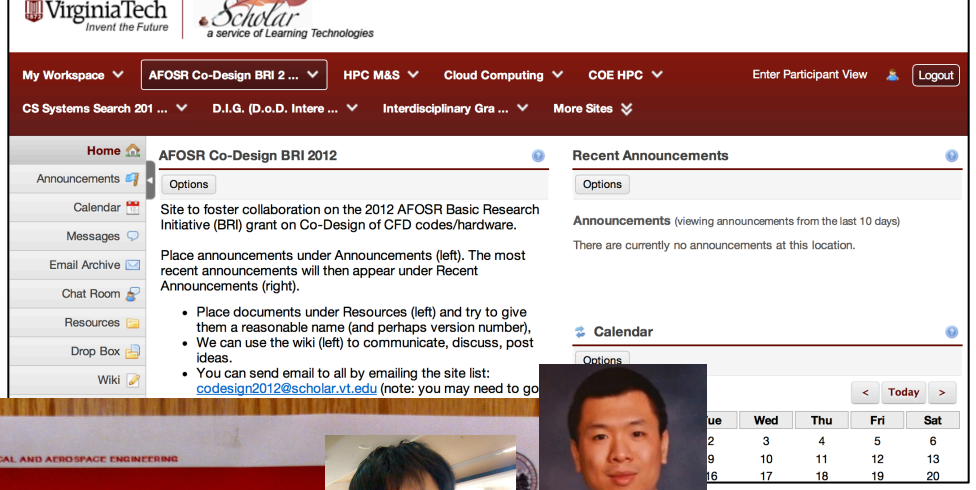
Vision



- Synergistic co-design process for the *structured/unstructured grid motifs* (or dwarfs) in computational fluid dynamics (CFD) to support aerodynamic predictions for micro-air vehicles (MAVs).
  - Malleable **algorithms**
    … that can be mapped and optimized in **software**
    … onto the right type of processing core in **hardware**
    … at the right time
  - Co-design feedback to vendors to assist in guiding future hardware design

Synergistic Co-Design: Enabling and Empowering





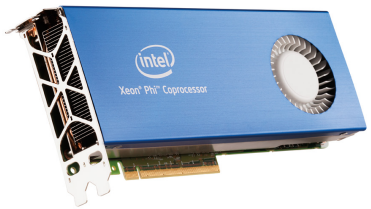
Team @ VT & NCSU

- Virginia Tech (13)
  - Eric de Sturler — Numerical Methods (solvers & preconditioners)
  - Wu Feng — Parallel Computing (performance, power, portability)
  - Chris Roy — CFD (structured grid and ALE mesh movement)
  - Adrian Sandu — Numerical Methods (time stepping & discretization)
  - Danesh Tafti — CFD (pressure-based multiblock structured)
  - Research Scientist (1), Postdocs (2), and Graduate Students (5)
- North Carolina State University (7)
  - Jack Edwards — CFD (multiblock structured w/ implicit solvers)
  - Hong Luo — CFD (unstructured grid / compressible)
  - Frank Mueller — Parallel Computing (languages, compilers, scalability)
  - Postdocs (2), Graduate Students (2)

Why Synergistic Co-Design? Why Now?

- Increasing heterogeneity in computing resources

Altera FPGA



Intel MIC

NVIDIA GPU

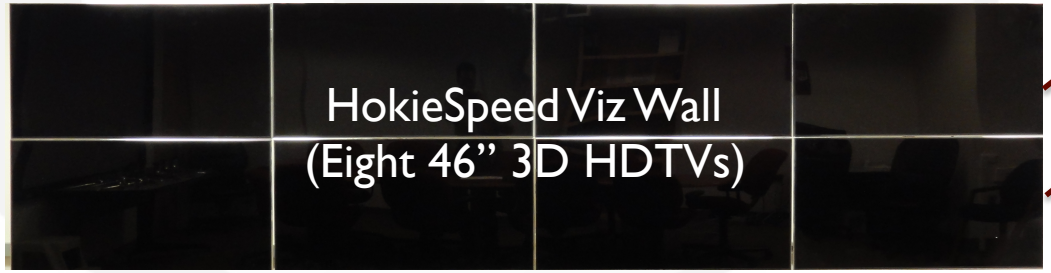
TI DSP

… across a wide variety of environments

Heterogeneous Systems in HPC

- Statistics
  - *Four* out of top 10 systems
  - Performance share in Top500 systems
    5% (2009) → 35% (2013)
- HokieSpeed
  - CPU+GPU heterogeneous supercomputer with large-scale visualization wall

Debuted as *GREENEST* commodity supercomputer in the U.S. in Nov. 2011 on THE GREEN 500

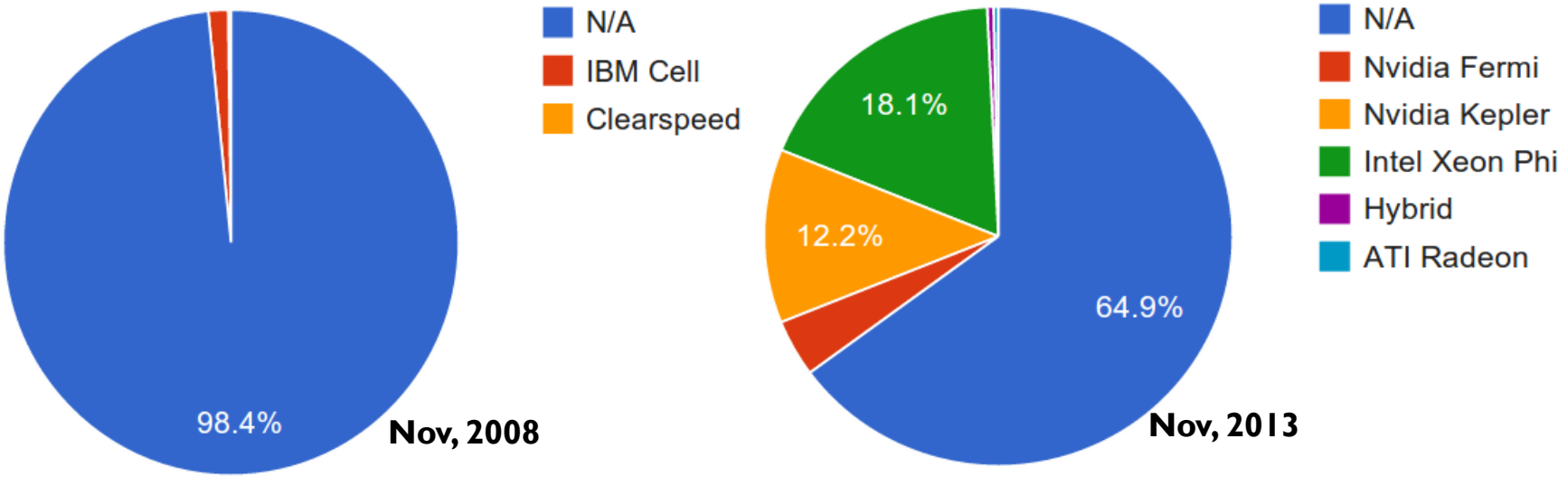


Tianhe-2

Top 10 Source: top500.org

1. **Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 ...ss-2, Intel Xeon Phi 31S1P
   NUDT
2. **Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x
   Cray Inc.
3. **Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom
   IBM
4. **K computer**, SPARC64 VIIIfx 2.0GHz, Tofu interconnect
   Fujitsu
5. **Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom
   IBM
6. **Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x
   Cray Inc.
7. **Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P
   Dell
8. **JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect
   IBM
9. **Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect
   IBM
10. **SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR

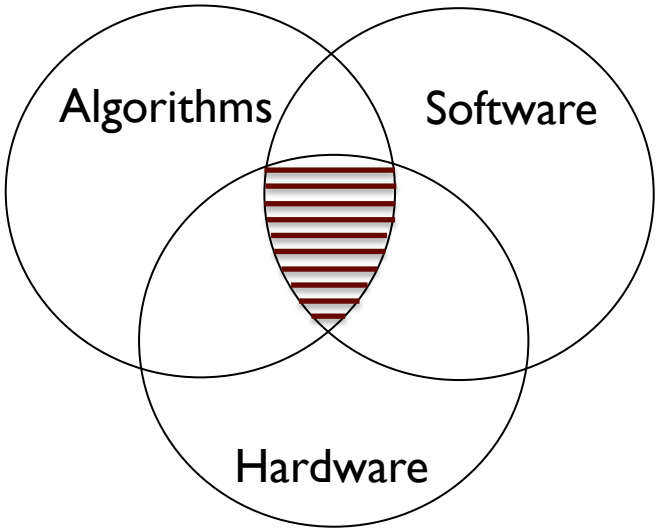
Diversity of Heterogeneous Systems



Performance Share of Accelerators in Top500 Systems

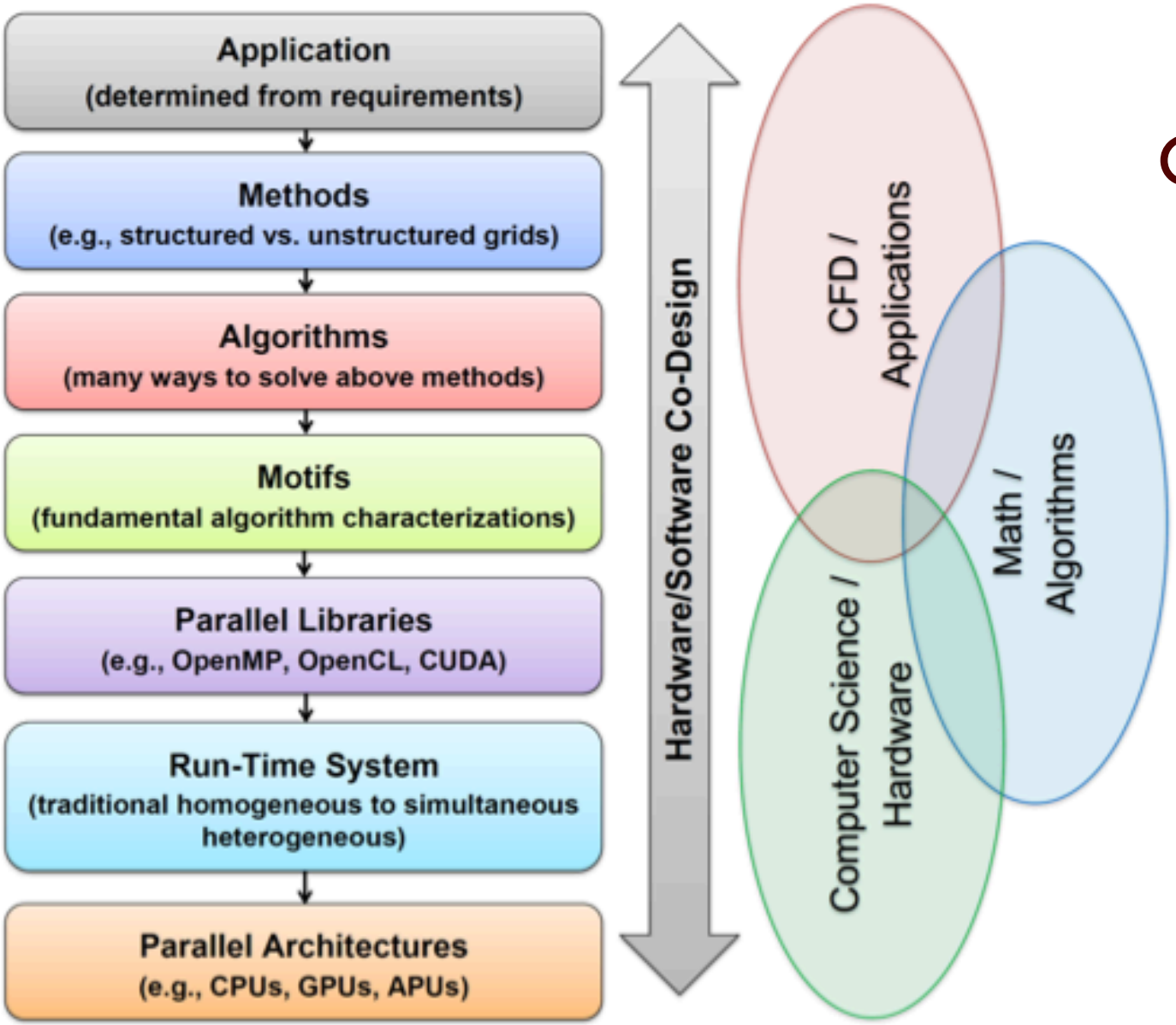
Roadmap

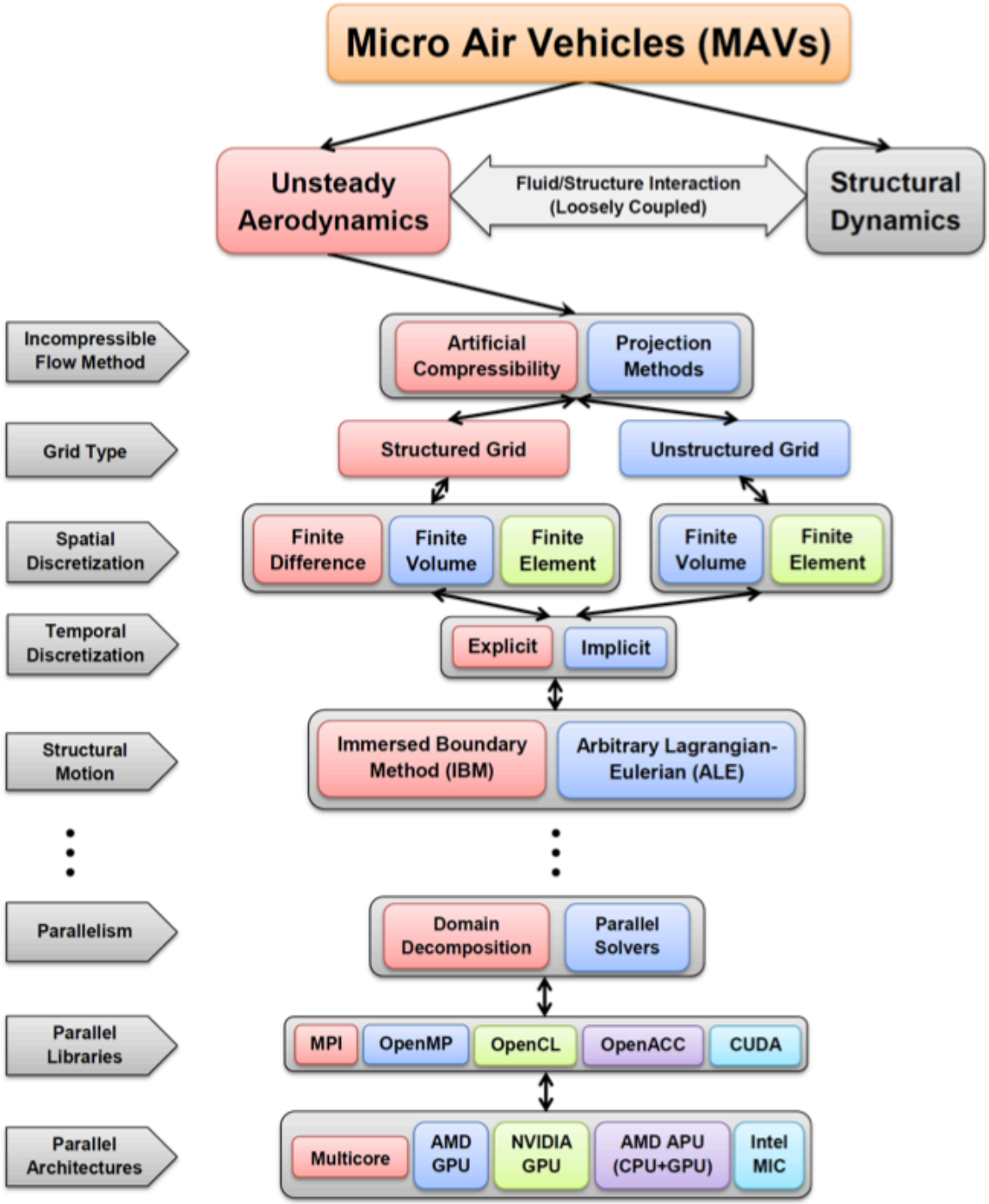
- Vision
- Team
- Approach
- Infrastructure
- Co-Design Research
  - Computer Science
  - CFD Codes (4)
  - Mathematics
- Achievements & Publications
- Next Steps



Concept presented at a White House BIGDATA Event in May 2013

Layered Co-Design





Co-Design for Micro Air Vehicles (MAVs)

Insert titles here
For co-design

2014 AFOSR BRI Workshop
W. Feng, wfeng@vt.edu, 540.315.1545

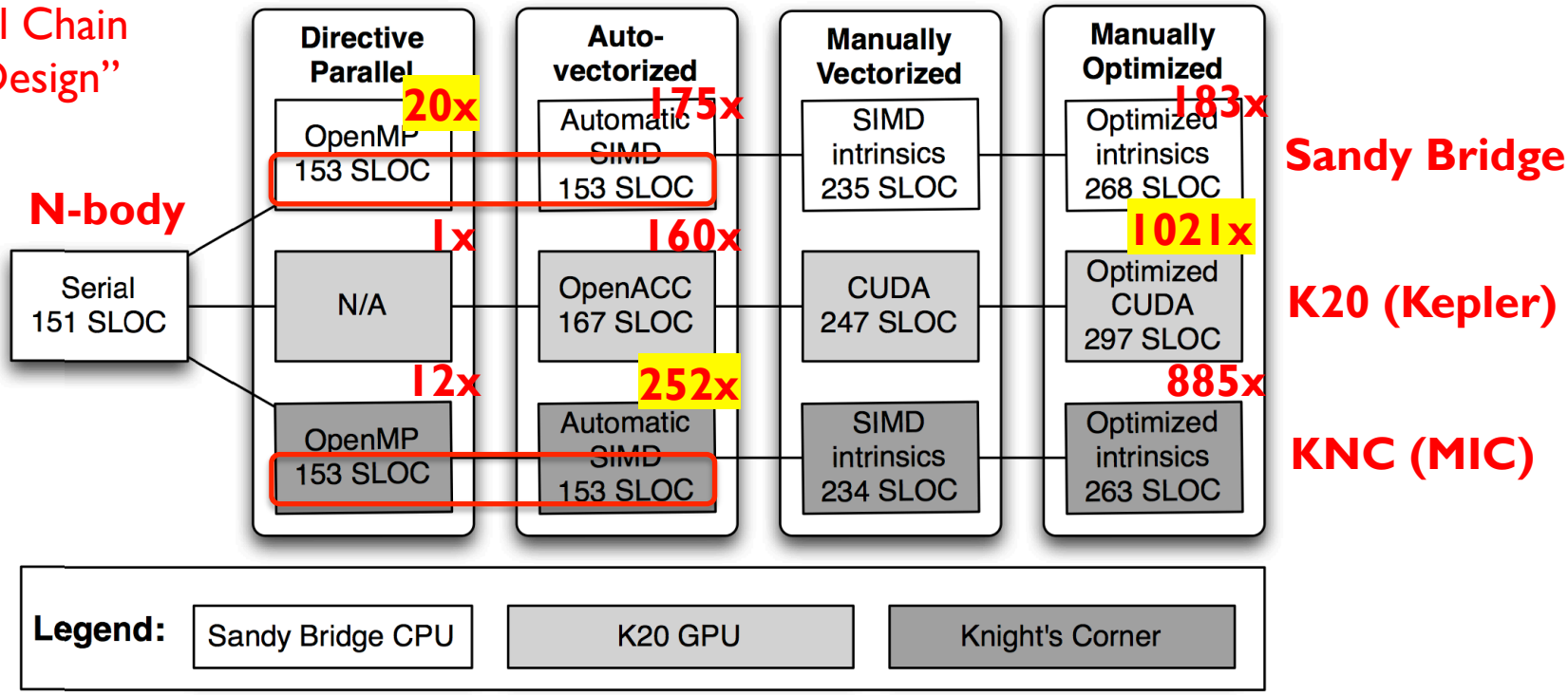
Co-Design Around Three P's:
Performance, Programmability, Portability

"Productivity = Performance + Programmability + Portability"

- Multi-dimensional optimization across two or more P's

… first *manual co-design* … then *automated co-design*

See "Tool Chain for Co-Design"

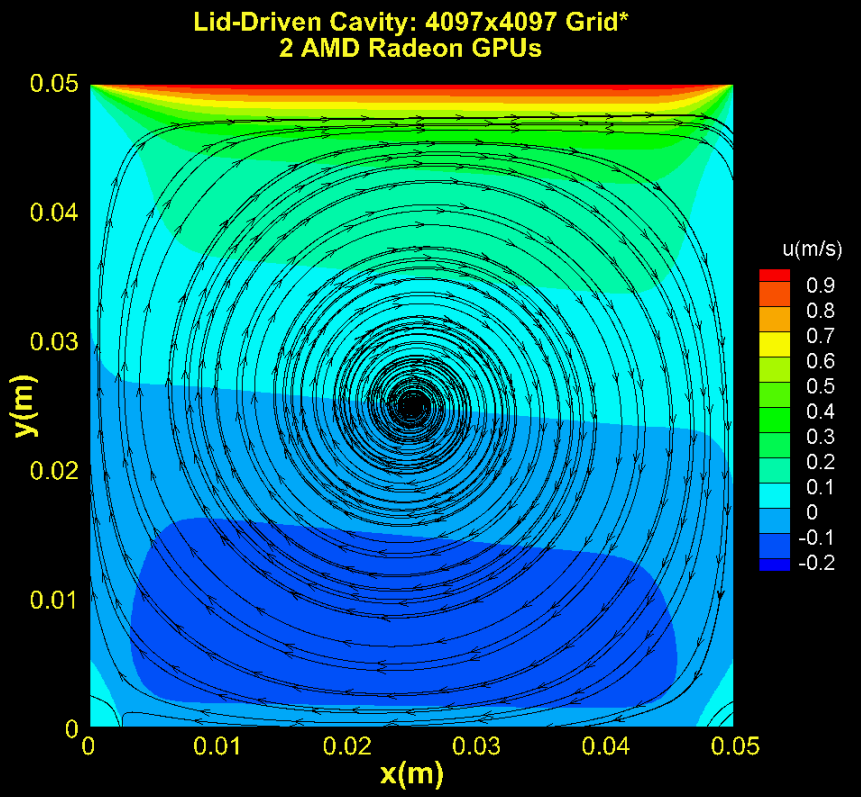


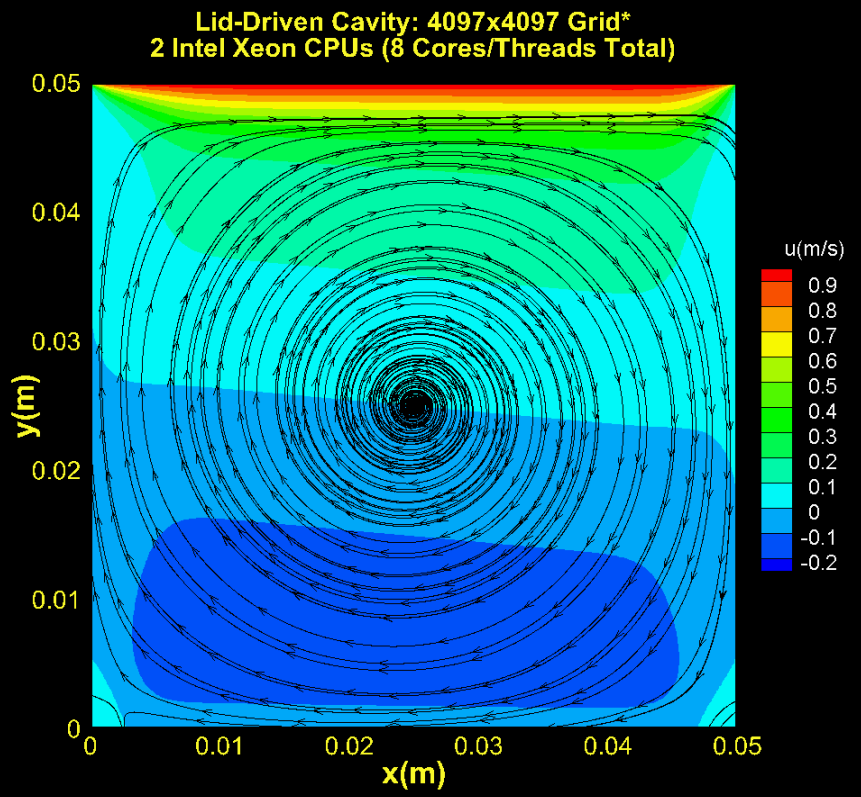
On the Programmability & Performance of Heterogeneous Platforms, *19th IEEE Int'l Conf. on Parallel and Distributed Systems*, Dec. 2013.

Lid-Driven Cavity: 2 GPUs vs. 2 CPUs

Brent Pickering and Christopher Roy, Virginia Tech

8X Speed-Up



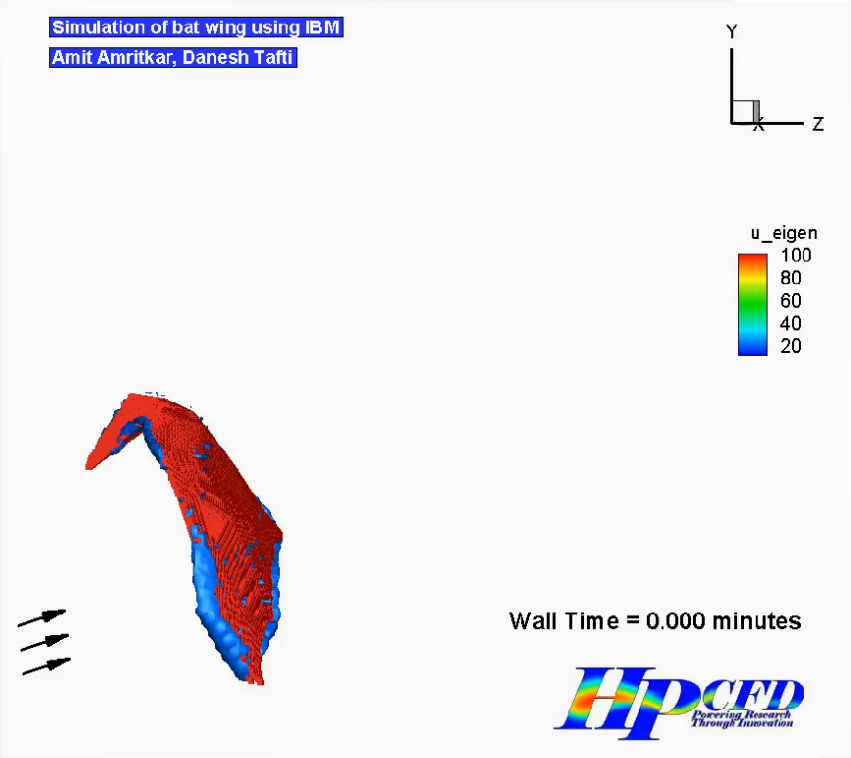


*Results shown on a 257x257 mesh to reduce file size

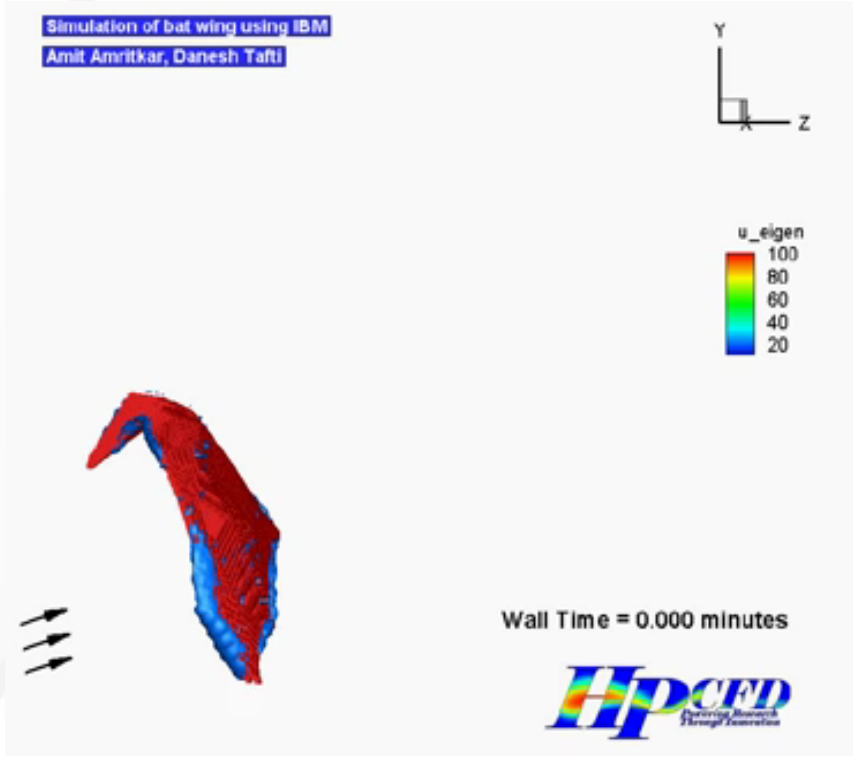
Bat Wing Simulation on GPU and CPU

Amit Amritkar and Danesh Tafti, Virginia Tech

GPU

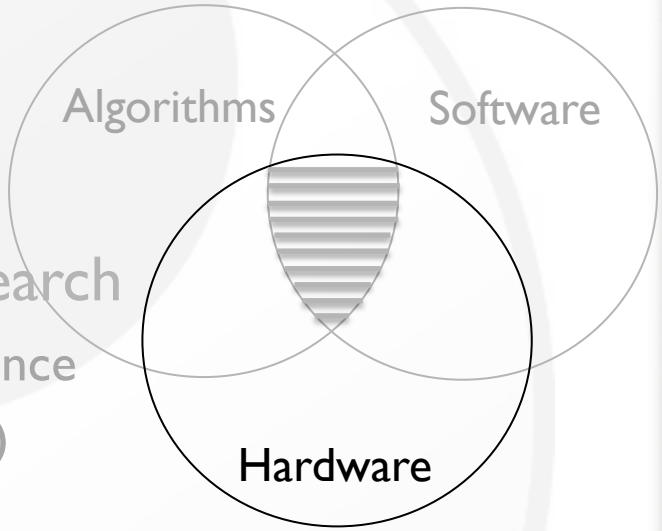


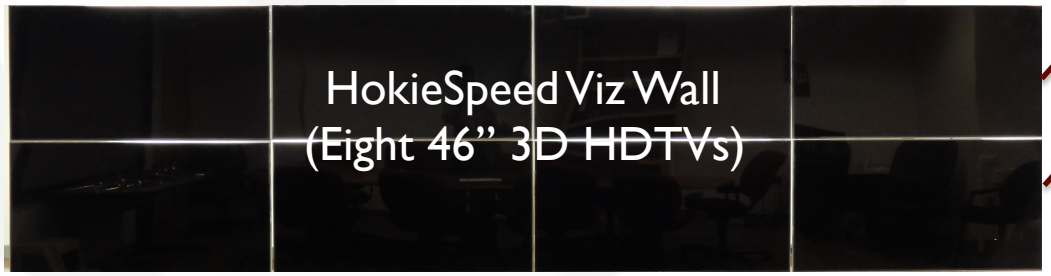
CPU



Roadmap

- Vision
- Team
- Approach
- Infrastructure
- Co-Design Research
  - Computer Science
  - CFD Codes (4)
  - Mathematics
- Achievements & Publications
- Next Steps

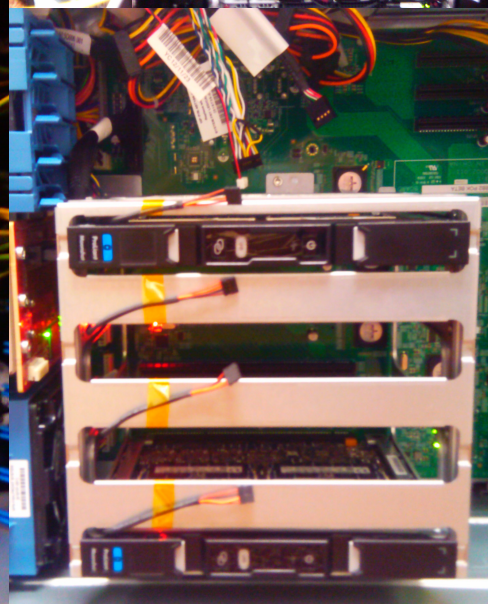




Experimental Systems





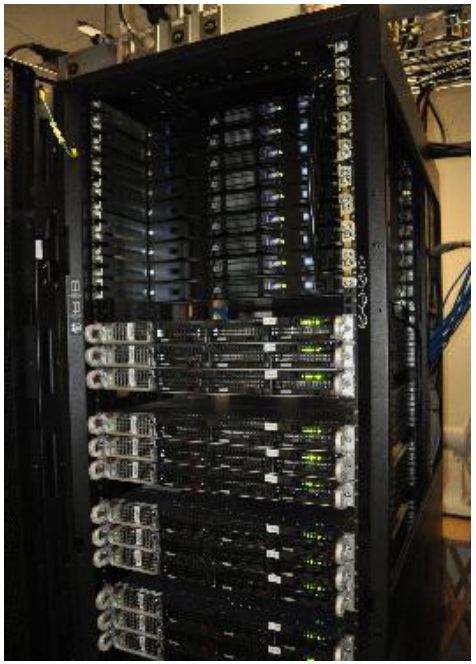
Add Intel MIC!

Add descriptive captions

W. Feng, wfeng@vt.edu, 540.315.1545

ARC Cluster

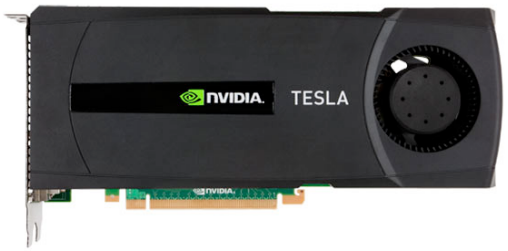
- Hardware
  - 2x AMD Opteron 6128 (8 cores each)
    - 108 nodes = 1728 CPU cores
  - NVIDIA GTX480, GTX680 , C2050, K20c: 108 GPUs
  - Mellanox QDR InfiniBand: 40Gbit/s
- Software
  - CUDA 5.0
  - PGI Compilers V13.4 w/ CUDA Fortran & OpenACC support
  - OpenMPI & MVAPICH2-1.9 w/ GPUDirect V2 capability
  - Torque/Maui job management system



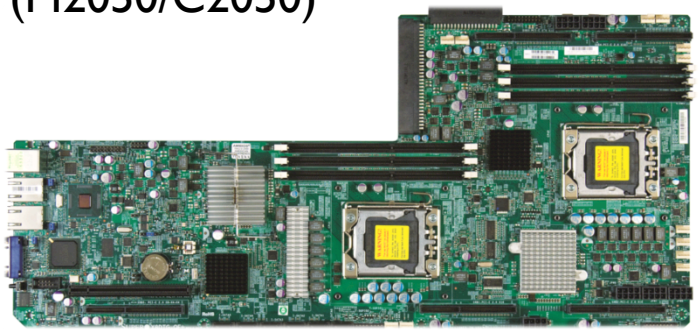
HOKIESPEED

A GPU-Accelerated Supercomputer for the Masses

- Purpose
  - To catalyze new approaches for conducting research via the synergistic amalgamation of heterogeneous CPU-GPU hardware and software
- Profile
  - Total Nodes: 209, where each compute node consists of
    - Motherboard: Supermicro 2026GT-TRF Dual Intel Xeon
    - CPUs: Two 2.4-GHz Intel Xeon E5645 6-core (12 CPU cores per node)
    - GPUs: Two NVIDIA Tesla Fermi GPUs (M2050/C2050)

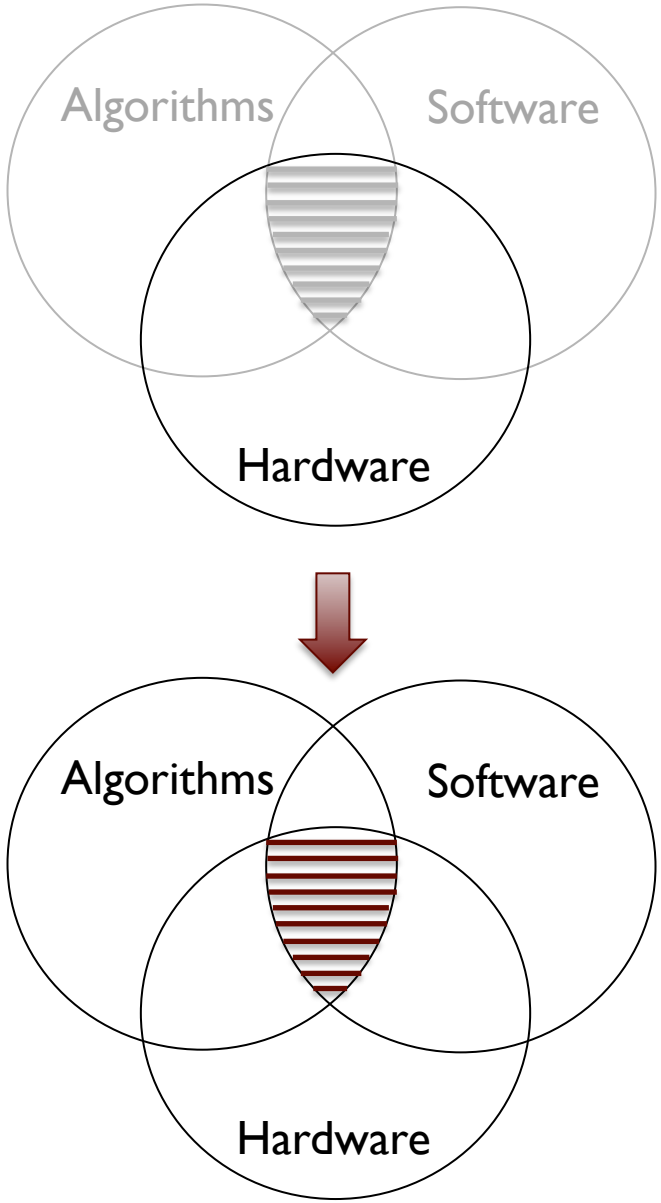


Add
"Software Environment"



Roadmap

- Vision
- Team
- Approach
- Infrastructure
- Co-Design Research
  - Computer Science (Feng, Mueller)
  - CFD Codes (Edwards, Luo, Roy, Tafti)
  - Mathematics (de Sturler, Sandu)
- Achievements & Publications
- Next Steps

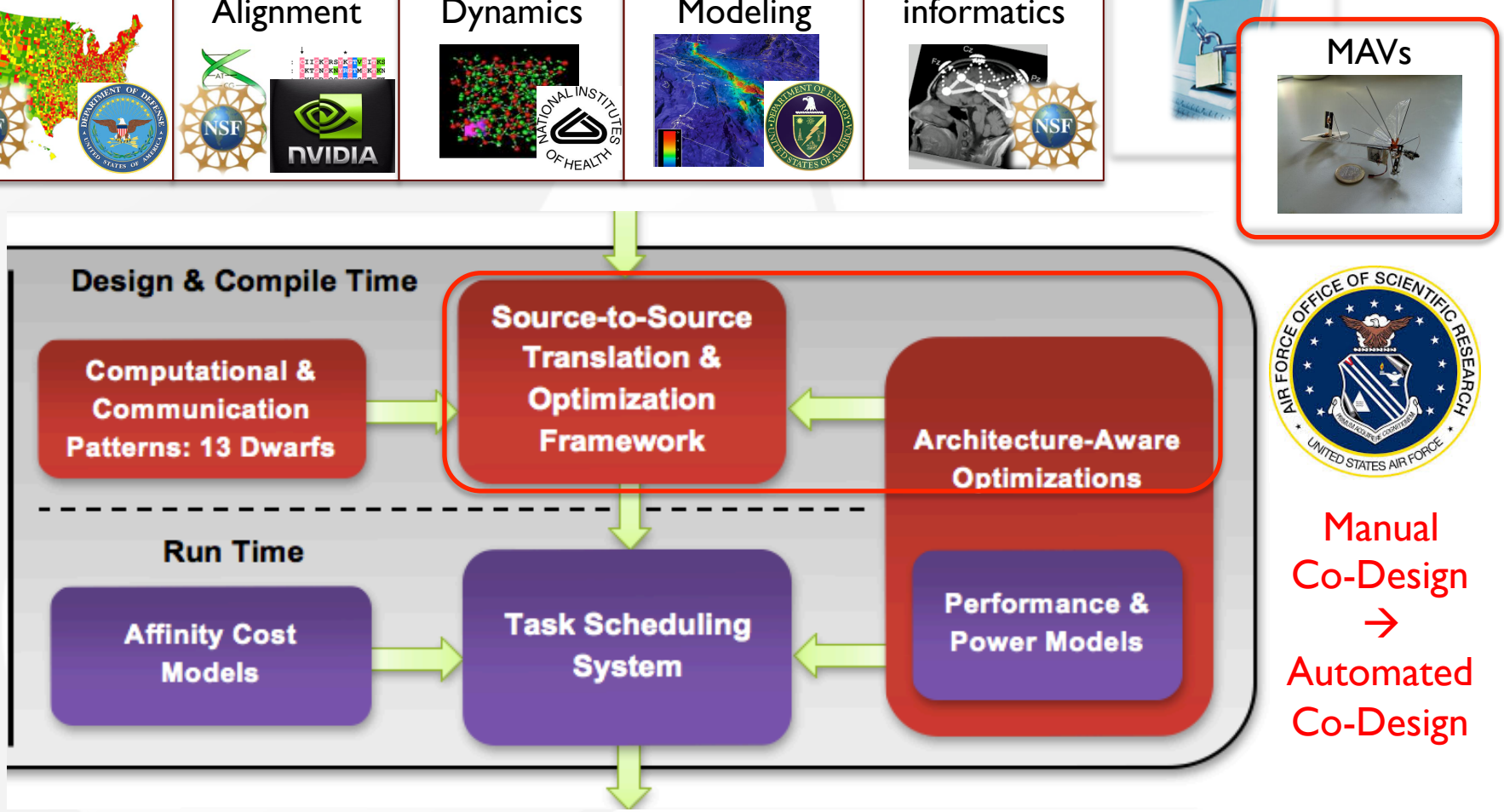


Intra-Node

Ecosystem for Heterogeneous Parallel Computing

MANUAL CO-DESIGN

Epidemiology | Sequence Alignment | Molecular Dynamics | Earthquake Modeling | Neuro-informatics | Cybersecurity | MAVs



Manual Co-Design → Automated Co-Design

Heterogeneous Parallel Computing Platform

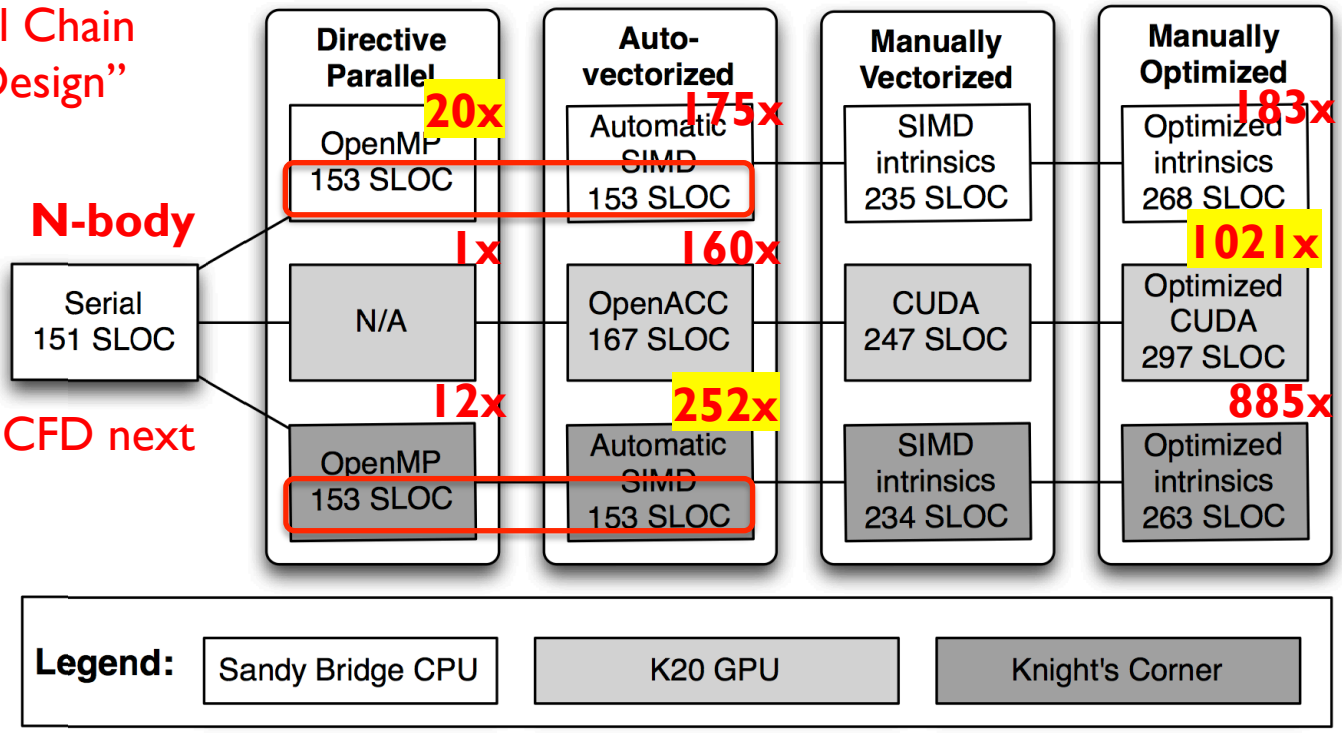
Co-Design Around Three P's:
Performance, Programmability, Portability

"Productivity = Performance + Programmability + Portability"

- Multi-dimensional optimization across two or more P's

… first *manual co-design* … then *automated co-design*

See "Tool Chain for Co-Design"



On the Programmability & Performance of Heterogeneous Platforms, *19th IEEE Int'l Conf. on Parallel and Distributed Systems*, Dec. 2013.

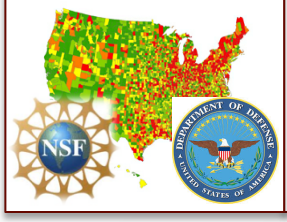
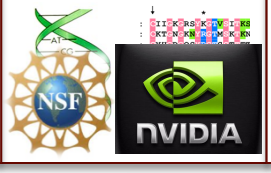
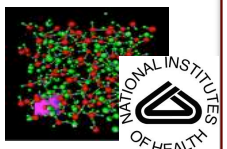
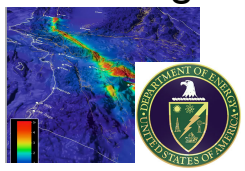
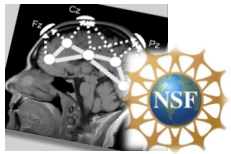
Acceleration Potential of Kernels

- Parallelism: Multicores and Accelerators (GPUs, Intel MIC)
- Problem: Existing Codes Sequential or Coarse Parallelism
  - Ad-hoc approach parallelization → unknown results
- Vision: Infer speedup potential before refactoring code
  1. Determine variable reuse for given architecture
  2. Estimate speedup for fine-grained parallelization
  3. Assess effects of manual code and data transformations
  4. Suggest (or auto-generate) code and data transformations

Intra-Node

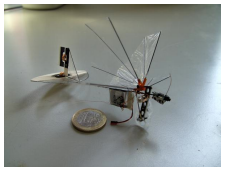
Ecosystem for Heterogeneous Parallel Computing

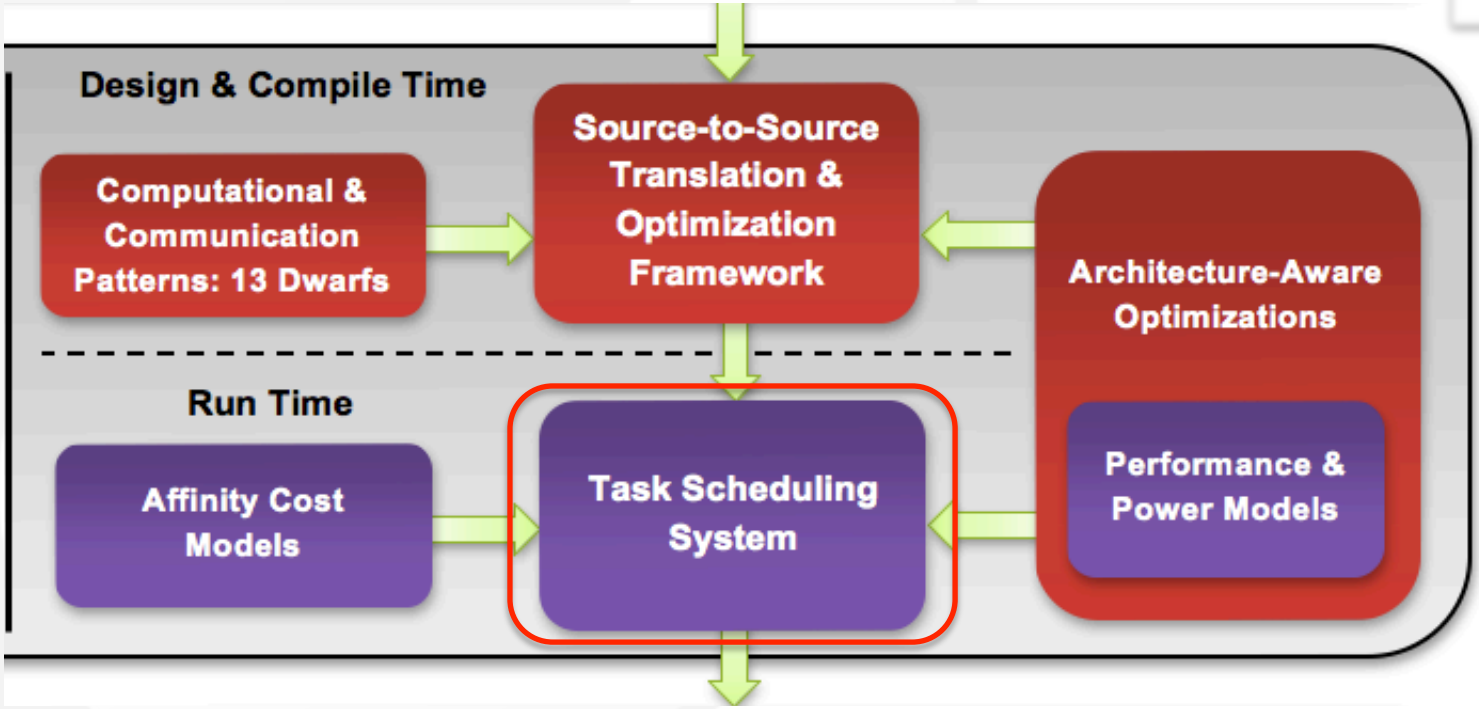
MANUAL CO-DESIGN

Epidemiology	Sequence Alignment	Molecular Dynamics	Earthquake Modeling	Neuro-informatics
				

Cybersecurity

MAVs

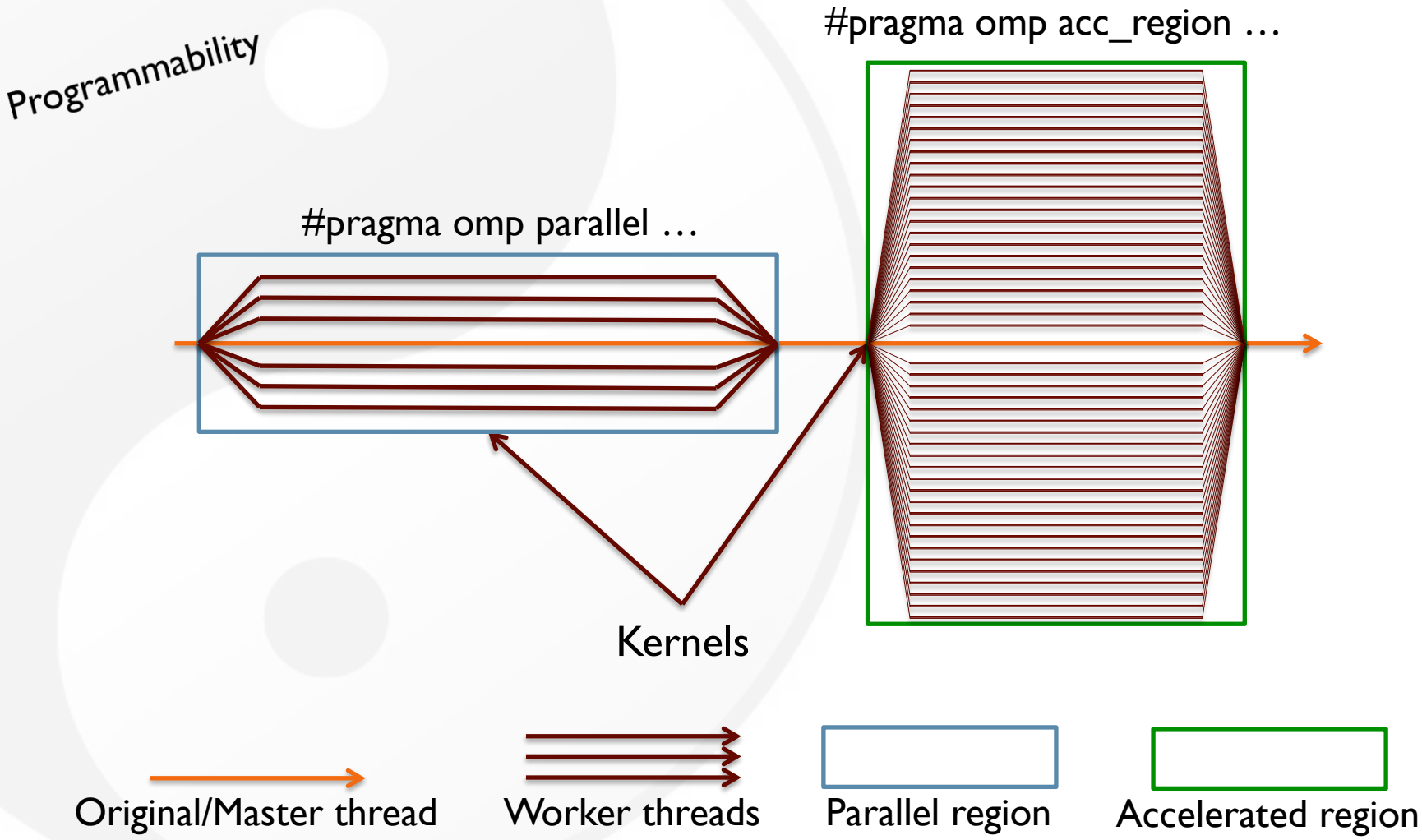




Manual Co-Design → Automated Co-Design

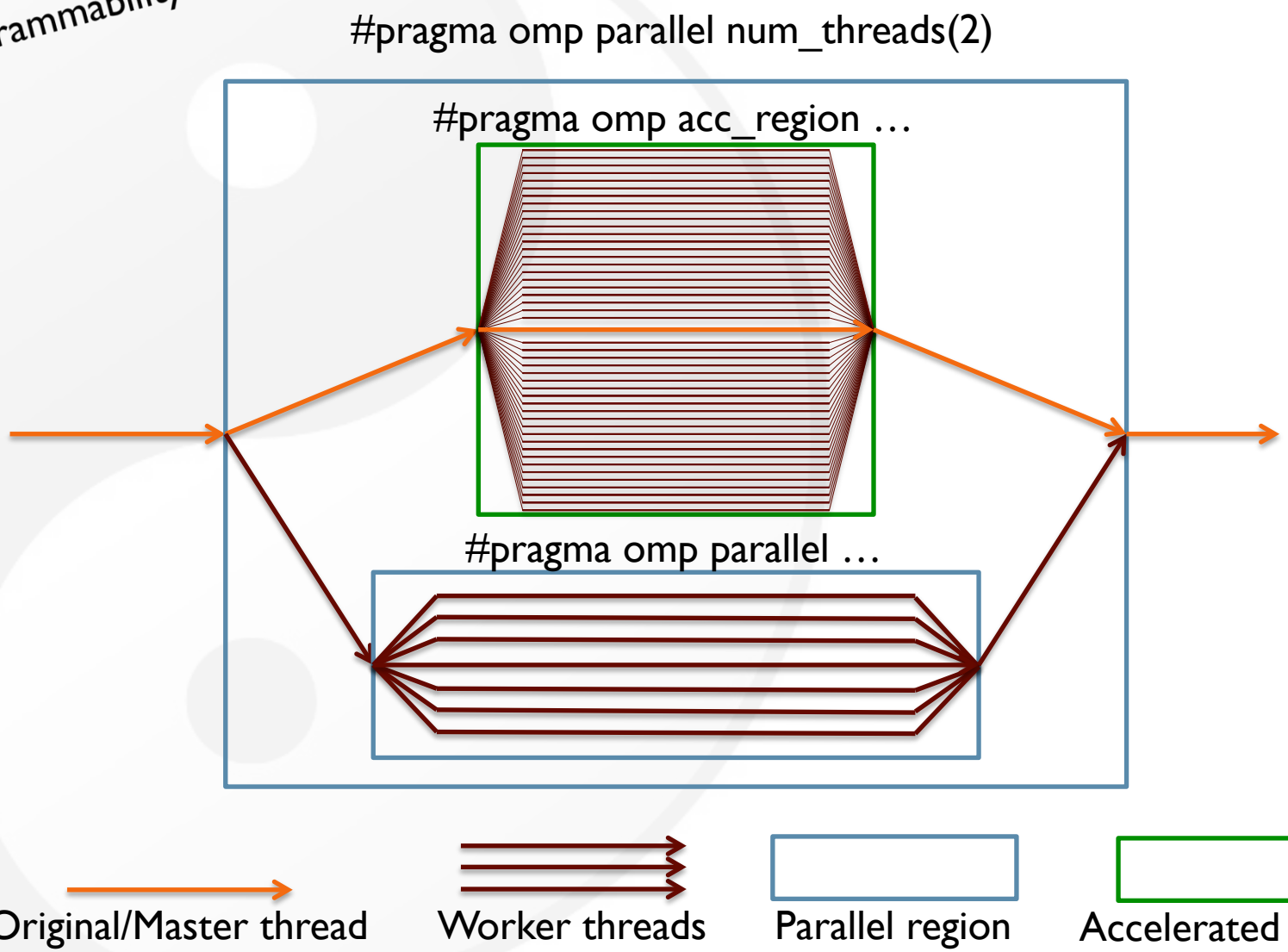
Heterogeneous Parallel Computing Platform

OpenMP Accelerator Behavior



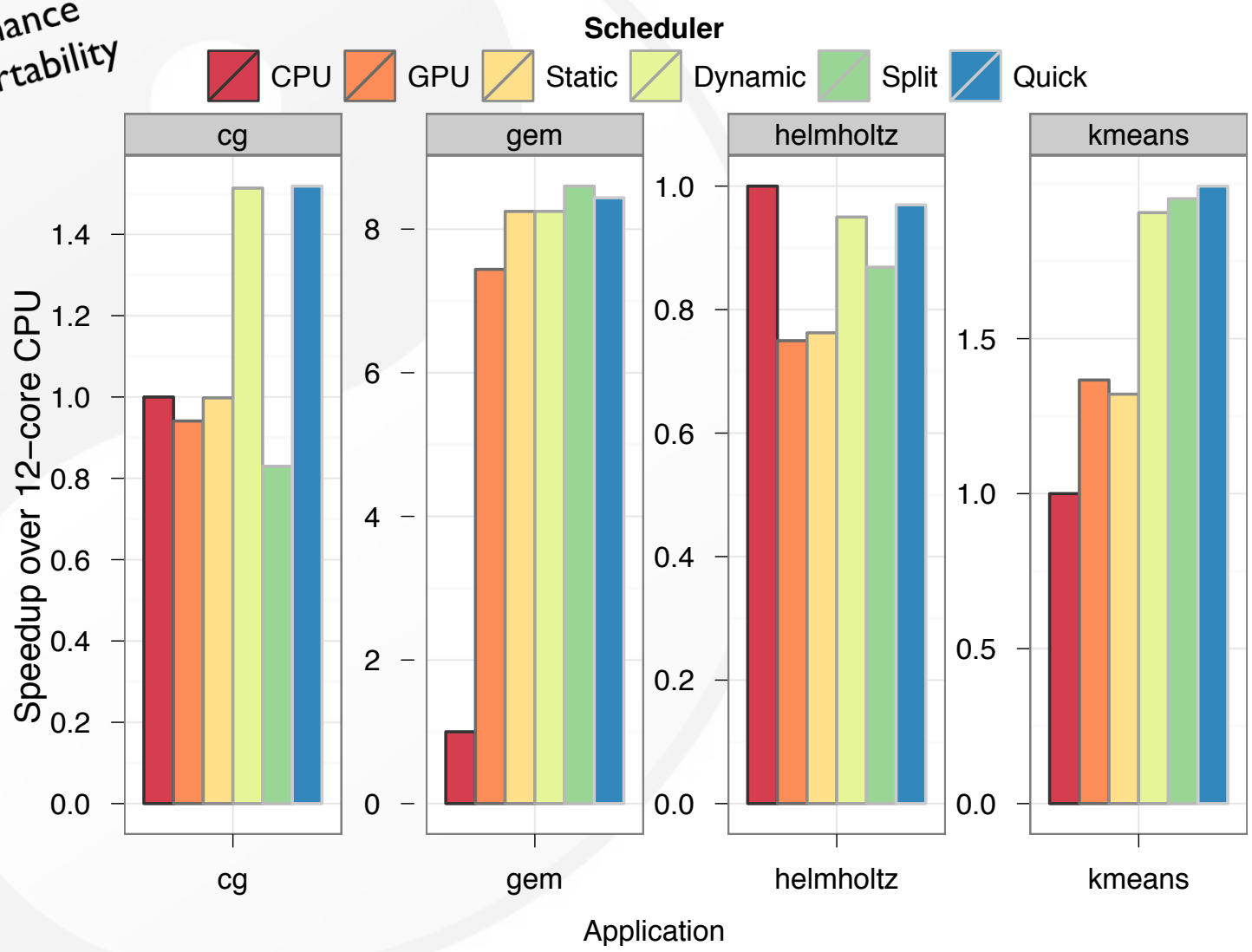
Our Version

Programmability



Preliminary Results of Automated Task Schedulers

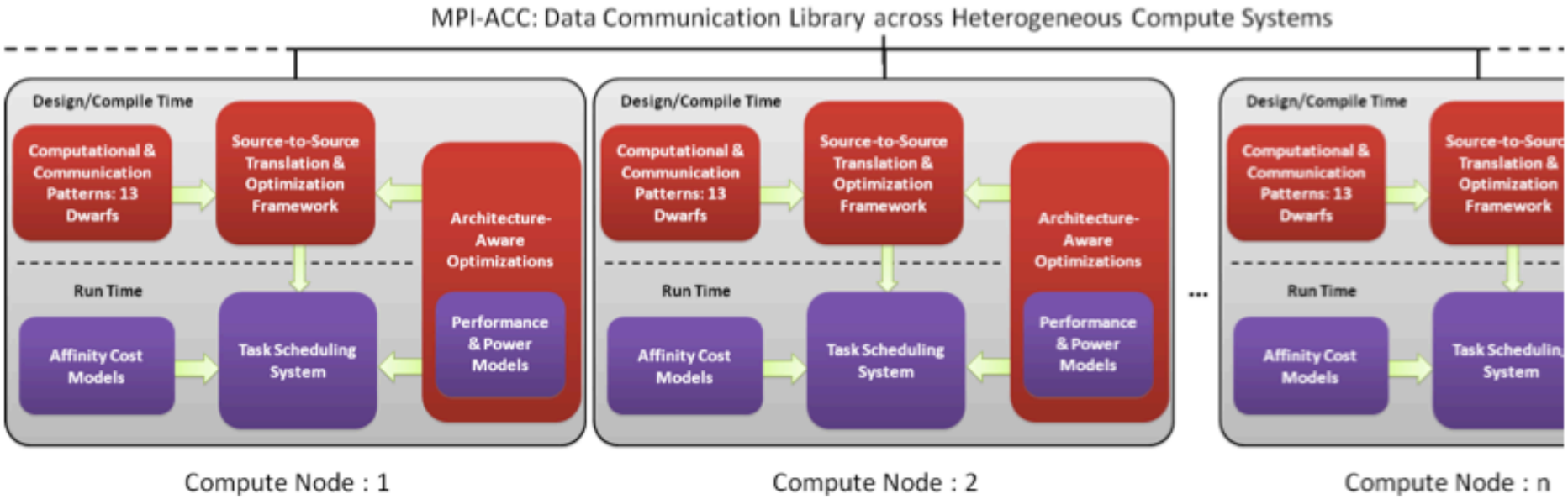
Performance and Portability



Inter-Node

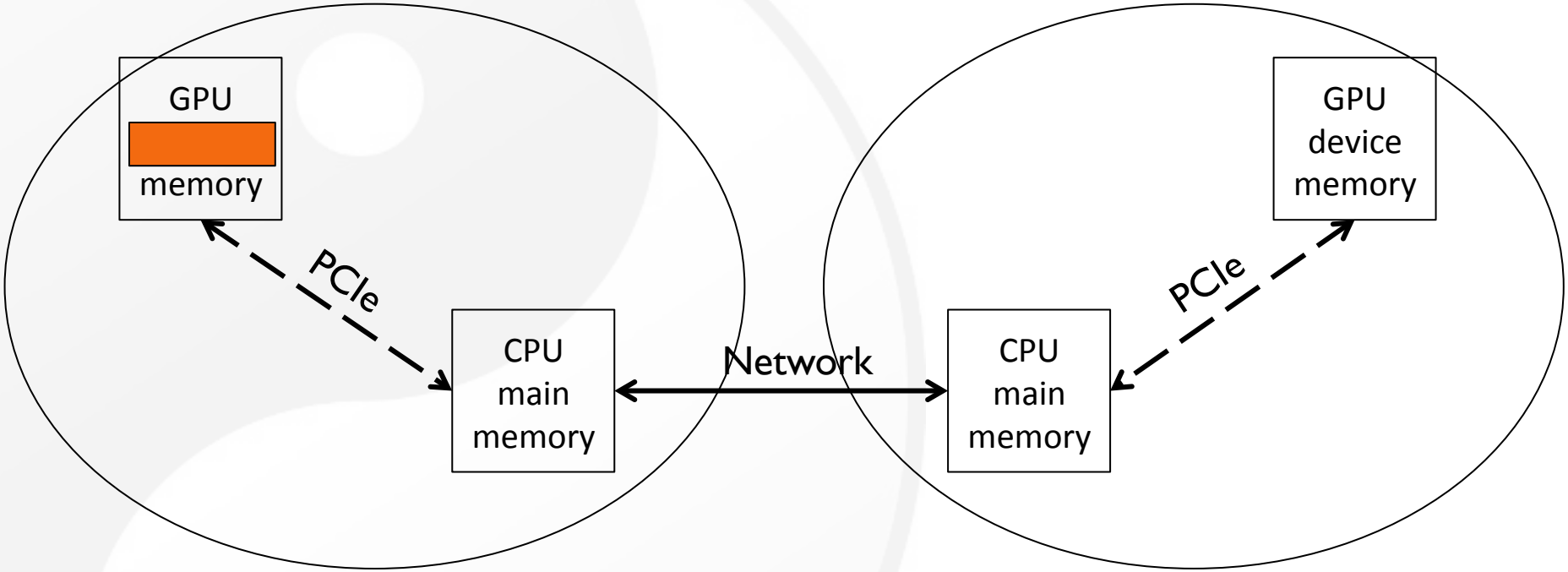
Ecosystem for Heterogeneous Parallel Computing

- Goal
  - Unified data movement library that hides all the hardware and system software details from the algorithm developer while supporting a multitude of environments



MPI-ACC: An Integrated and Extensible Approach to Data Movement in Accelerator-Based Systems by A. Aji, J. Dinan, D. Buntinas, P. Balaji, W. Feng, K. Bisset, R. Thakur. In Proc. 14th IEEE International Conference on High Performance Computing and Communications, Liverpool, UK, June 2012.

Data Movement in CPU-GPU Clusters



MPI Rank = 0

```
if(rank == 0)
{
  GPUMemcpy(host_buf, dev_buf, D2H)
  MPI_Send(host_buf, .. ..)
}
```

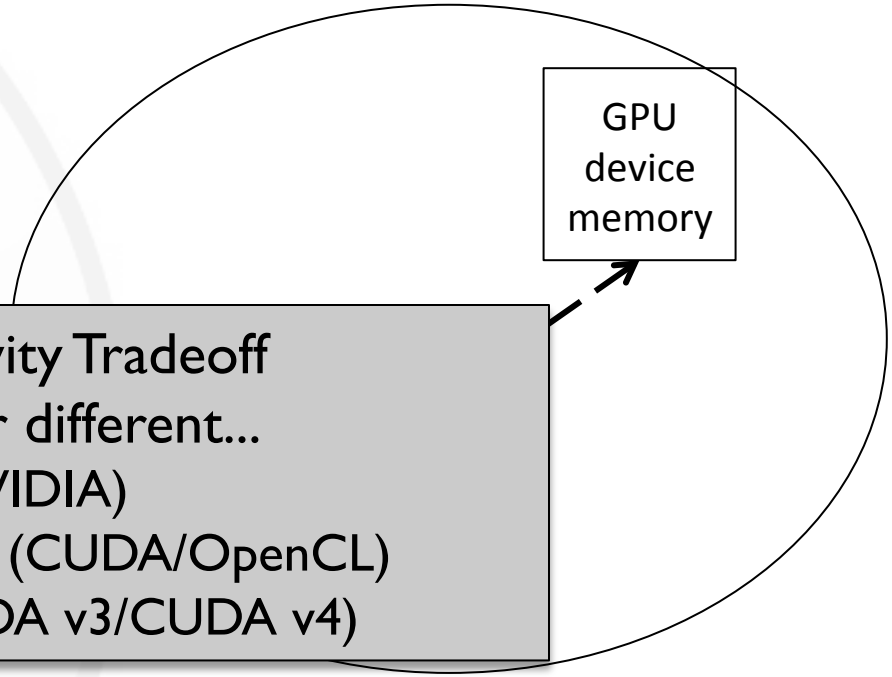
MPI Rank = 1

```
if(rank == 1)
{
  MPI_Recv(host_buf, .. ..)
  GPUMemcpy(dev_buf, host_buf, H2D)
}
```

Data Movement in CPU-GPU Clusters

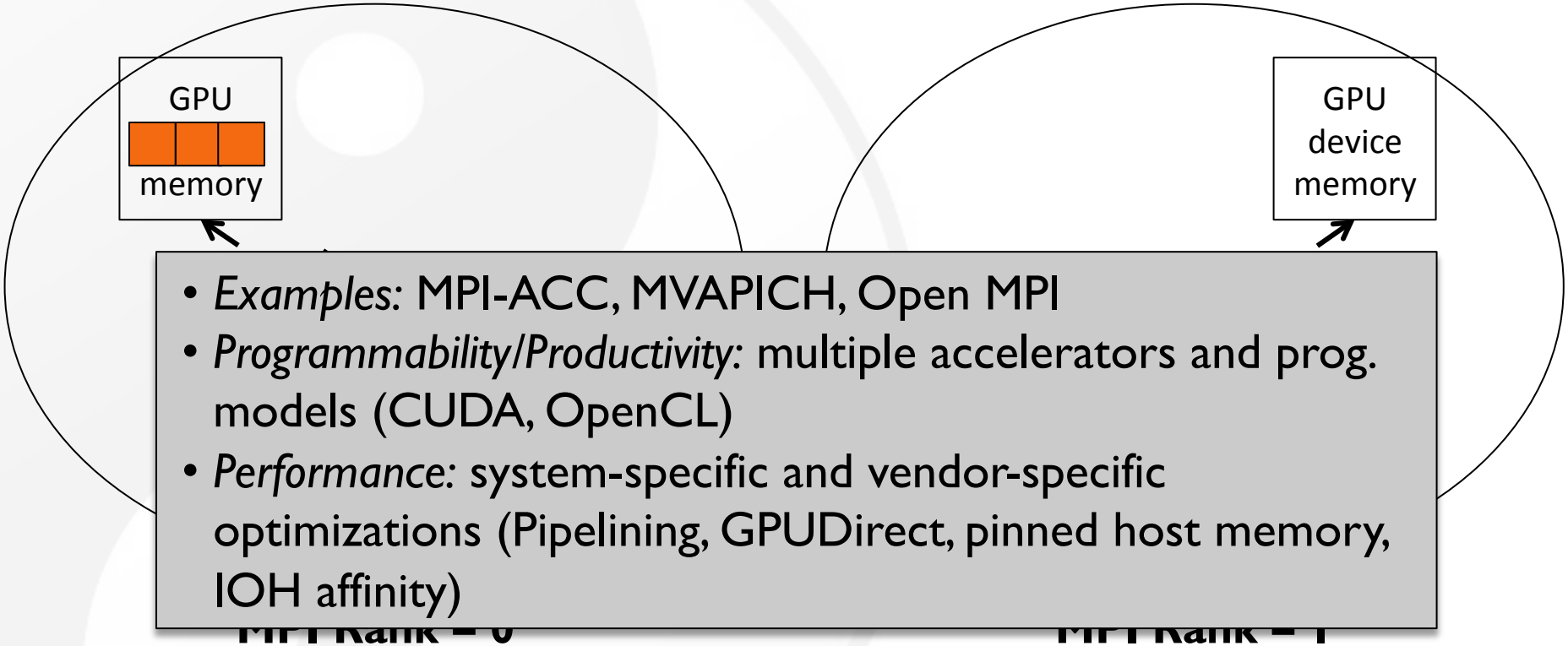
```
int processed[chunks] = {0};
for(j=0;j<chunks;j++) { /* Pipeline */
    cudaMemcpyAsync(host_buf+offset, gpu_buf
+offset,
            D2H, streams[j], ...);
}
numProcessed = 0; j = 0; flag = 1;
while (numProcessed < chunks) {
    if(cudaSt
cudaSuccess))
        /* st
        MPI_I
        numPr
    }
    MPI_Testa
    if(numPro
        while
            j=(j+1)%chunks; flag=processed[j];
        }
}
MPI_Waitall();
```

- Performance vs. Productivity Tradeoff
- Multiple code versions for different...
  - …GPUs (AMD/Intel/NVIDIA)
  - …programming models (CUDA/OpenCL)
  - …library versions (CUDA v3/CUDA v4)



MPI Rank = 1

GPU-Integrated MPI

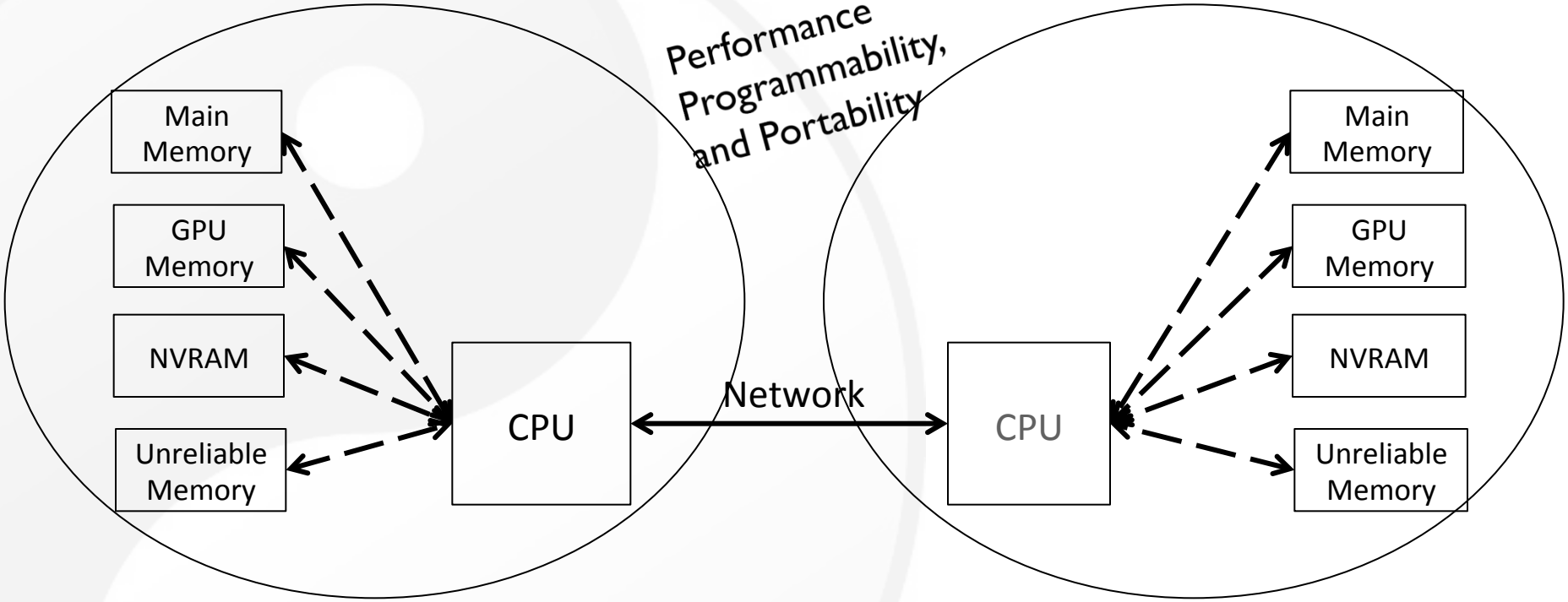


- *Examples:* MPI-ACC, MVAPICH, Open MPI
- *Programmability/Productivity:* multiple accelerators and prog. models (CUDA, OpenCL)
- *Performance:* system-specific and vendor-specific optimizations (Pipelining, GPUDirect, pinned host memory, IOH affinity)

```
if(rank == 0)
{
  MPI_Send(any_buf, .. ..)
}
```

```
if(rank == 1)
{
  MPI_Recv(any_buf, .. ..)
}
```

MPI-ACC: Generalized Runtime for Accelerator Systems



Rank = 0

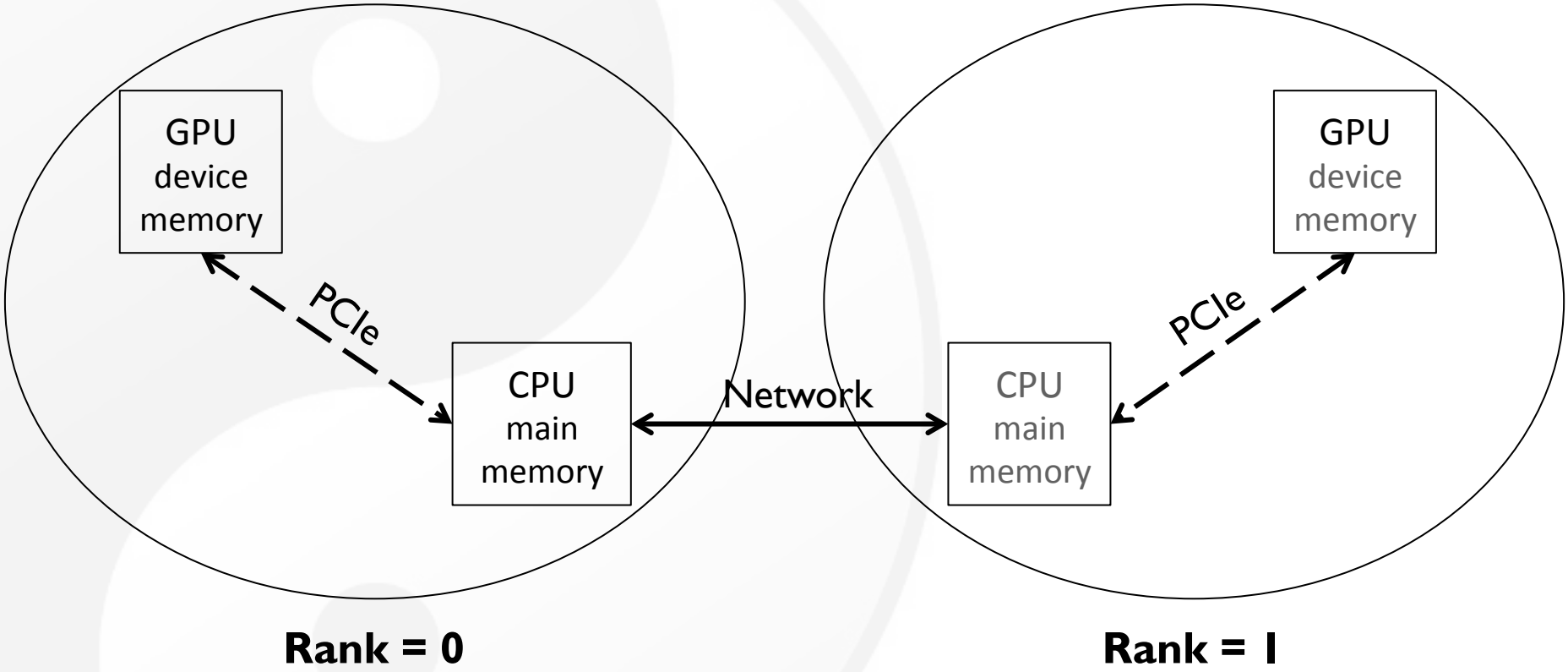
```
if (rank == 0)
{
    MPI_Send(s_buf, .. ..);
}
```

Rank = 1

```
if (rank == 1)
{
    MPI_Recv(r_buf, .. ..);
}
```

Goal of Programming CPU-GPU Clusters (MPI + Any Acc)

MPI-ACC: Generalized Runtime for Accelerator Systems



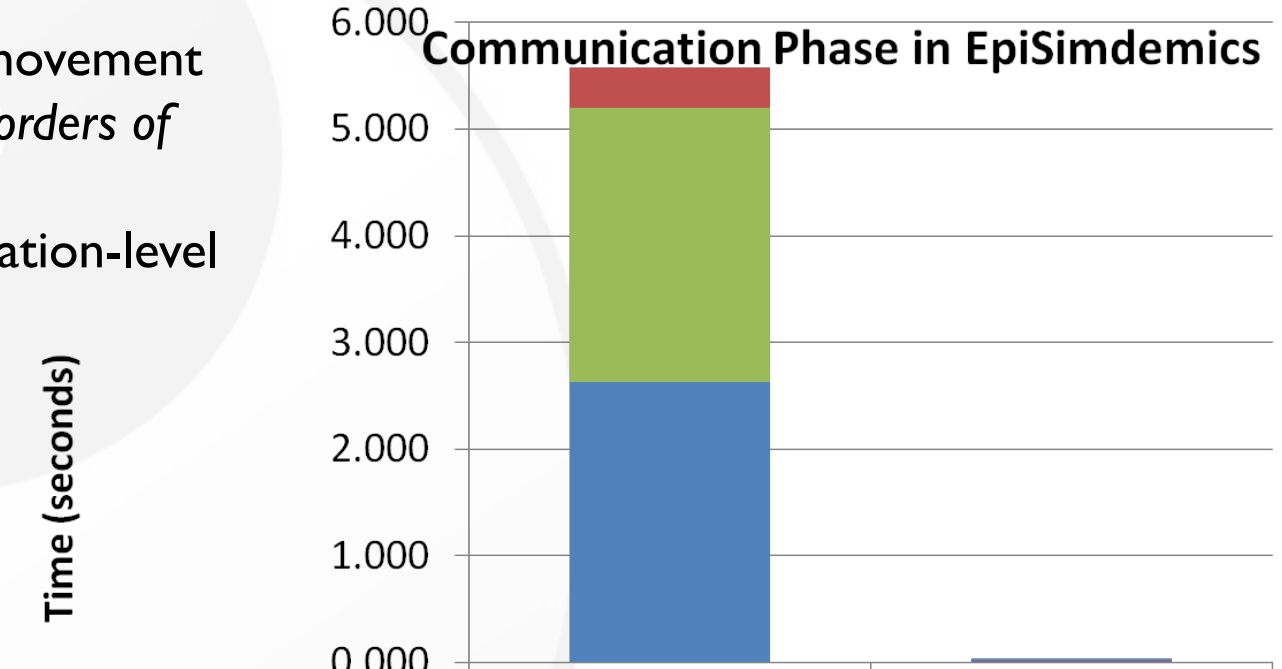

```
if(rank == 0)
{
  MPI_Send(any_buf, .. ..);
}
```

```
if(rank == 1)
{
  MPI_Recv(any_buf, .. ..);
}
```

MPI-ACC Performance Comparison

- Accelerates data movement operations *by two orders of magnitude*
- *Enables* new application-level optimizations

Performance Programmability, and Portability



	MPI + CUDA	MPI-ACC
D-D Copy (Packing)	0	0.003
GPU Receive Buffer Init	0	0.024
H-D Copy	0.382	0
H-H Copy (Packing)	2.570	0
CPU Receive Buffer Init	2.627	0

MPI-ACC runtime optimized

Year One: CS (Needs to be updated. Also less text.)

- Conduct R&D on optimizing and automating heterogeneous computing at the node level, e.g., automated run-time scheduling
- Develop, characterize, and optimize/re-factor dwarf abstractions (composition of dwarfs?)
- Interact with methods/algorithms and applications teams on mapping to heterogeneous systems. (Slightly more detail needed, e.g., tradeoffs on mapping explicit/implicit to GPU, data-transfer overhead of codes, interfacing to linear solvers like Trilinos, etc.)
- Support domain scientists on heterogeneous computing w/ GPUs
- Infrastructure: Tools for domain scientists and engineers, e.g., PGI Accelerator Suite (including OpenACC, PGI Fortran compiler, etc.)
- Hire CS postdoc and 2 GRAs (Adrian + Wu)
  - GRA Ross Glandon: Continue to educate him(self) on both time stepping algorithms and parallel computing
  - Other GRA still to be hired (from Kaixi, Sriram, Aniket, Lokendra: post-September 2013, Tom: post-September 2013)

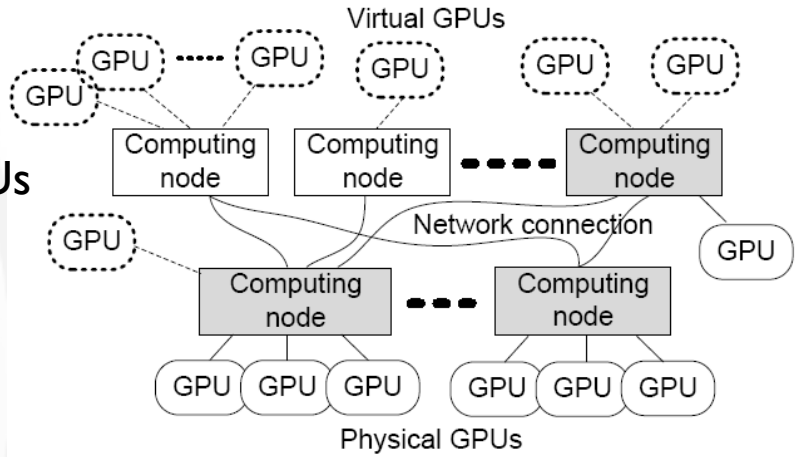
Next Steps

- Next steps
  - A list of (re-factored) deliverables and tasks for our program manager.

- Need a picture of hierarchical parallelism
  - Need a picture of MPI … domain scientists responsible for mapping/ decomposition
  - Need a picture of OpenMP/OpenCL … automated portion for supporting task scheduling

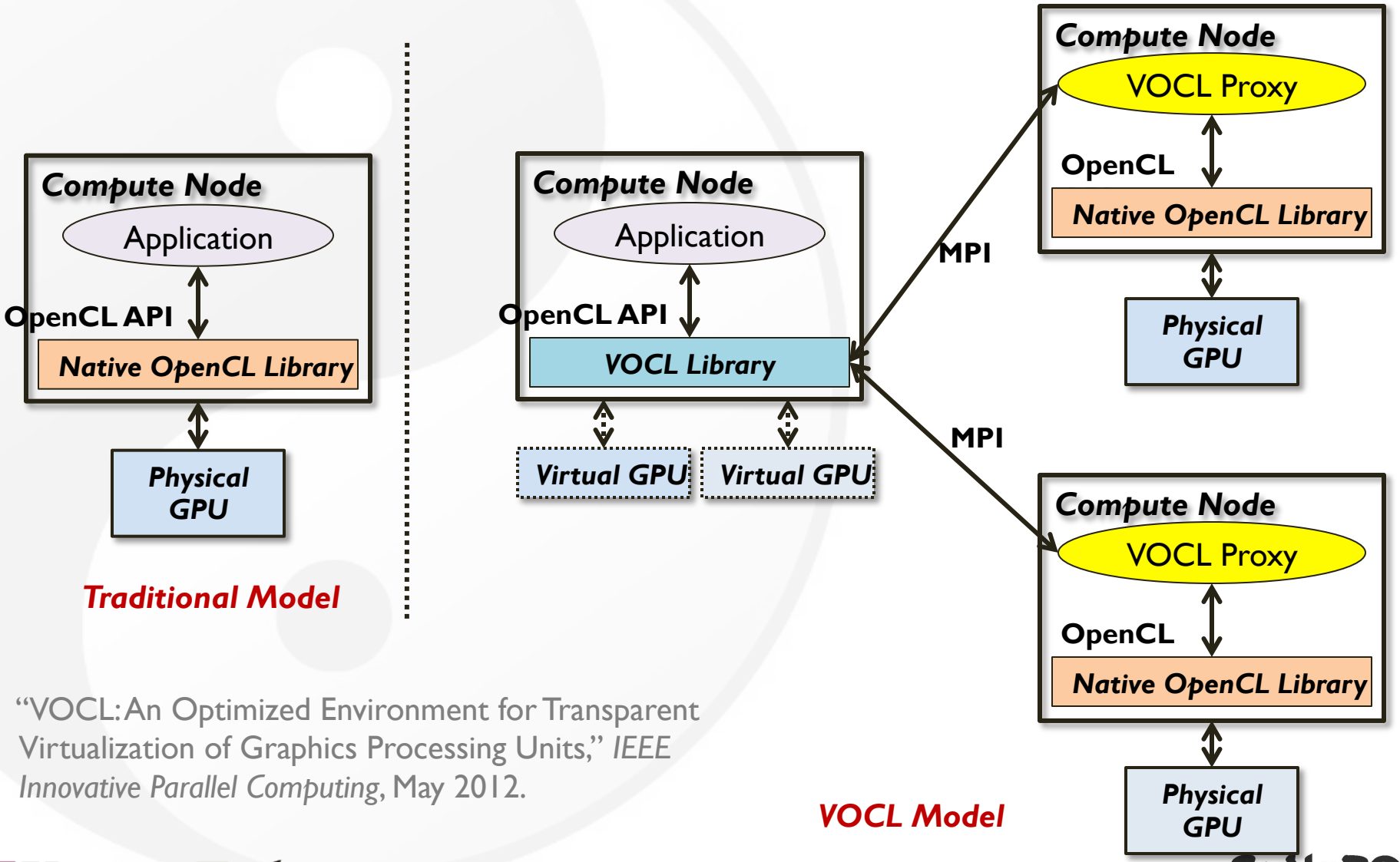
VOCL: A Virtual Implementation of OpenCL for Access and Management of Remote GPU Devices

- GPU Virtualization
  - Transparent utilization of remote GPUs
    - Remote GPUs look like local "virtual" GPUs
    - Applications can access them as if they are regular local GPUs
    - VOCL will automatically move data and computation
  - Efficient GPU resource management
    - Virtual GPUs can migrate from one physical GPU to another
    - If a system admin wants to add or remove a node, he/she can do that while applications are running (hot-swap capability)



- "VOCL: An Optimized Environment for Transparent Virtualization of Graphics Processing Units," *IEEE Innovative Parallel Computing*, May 2012.
- "Transparent Accelerator Migration in a Virtualized GPU Environment," *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2012.

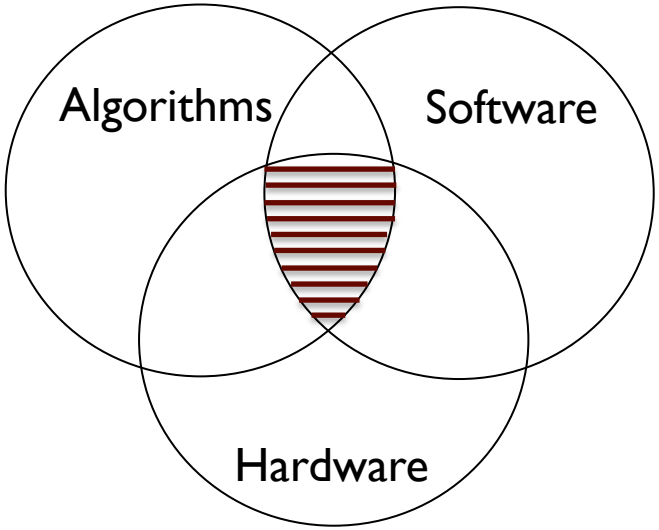
Virtual OpenCL (VOCL) Framework



"VOCL: An Optimized Environment for Transparent Virtualization of Graphics Processing Units," *IEEE Innovative Parallel Computing*, May 2012.

Roadmap

- Vision
- Team
- Approach
- Infrastructure
- Co-Design Research
  - Computer Science
  - CFD Codes (4)
  - Mathematics
- Achievements & Publications
- Next Steps



Overview of MAV Requirements

Algorithm Choice	Processing Req'ts	Communic. Req'ts	Memory Req'ts
Grid Type			
Structured			
Unstructured			
Spatial Discretization			
Finite Difference			
Finite Volume			
Finite Element			
Temporal Discretization			
Explicit			
Implicit (line)			
Implicit (plane)			
Implicit (volume)			

Overall Objective

- Extend and port our (four) CPU-based CFD methods to modern heterogeneous computing systems, particularly those with GPUs, via a synergistic hardware/software co-design approach that seeks to significantly improve performance, thus significantly enhancing the computational capabilities of current CFD tools.
  - Characterize and implement the following methods:
    - Projection and artificial compressibility methods for incompressible flows
    - Structured, unstructured, and Cartesian grids
    - Arbitrary Lagrangian-Eulerian (ALE) and immersed boundary methods (IBM) for boundary deformation
    - Finite volume and finite element methods based on the reconstructed discontinuous Galerkin (RDG) technique.
  - Co-design, test, verify, and assess the above methods for solving a variety of low Reynolds number incompressible flow problems of interest to the U.S. Air Force in general and for predicting MAV aerodynamics in particular.

Targeted CFD Codes

SENSEI (C. Roy, Virginia Tech)

- Structured, multiblock, 2nd order, finite-volume code
- Artificial compressibility method
- Arbitrary Lagrangian/Eulerian (ALE) 2nd or higher order spatial accuracy
- Artificial compressibility (AC) and immersed boundary (IB) methods

GenIDLEST (D. Tafti, Virginia Tech)

- Structured, multiblock, 2nd order, finite-volume code
- Pressure projection method
- ALE and immersed boundary methods (IBM)

RDGFLO (H. Luo, NCSU)

- Unstructured, discontinuous Galerkin (DG) method
- High-order solution of compressible flows

INCOMP3D (J. Edwards, NCSU)

- Structured, multiblock finite-volume code
- Second or higher order spatial accuracy
- ALE and IBM

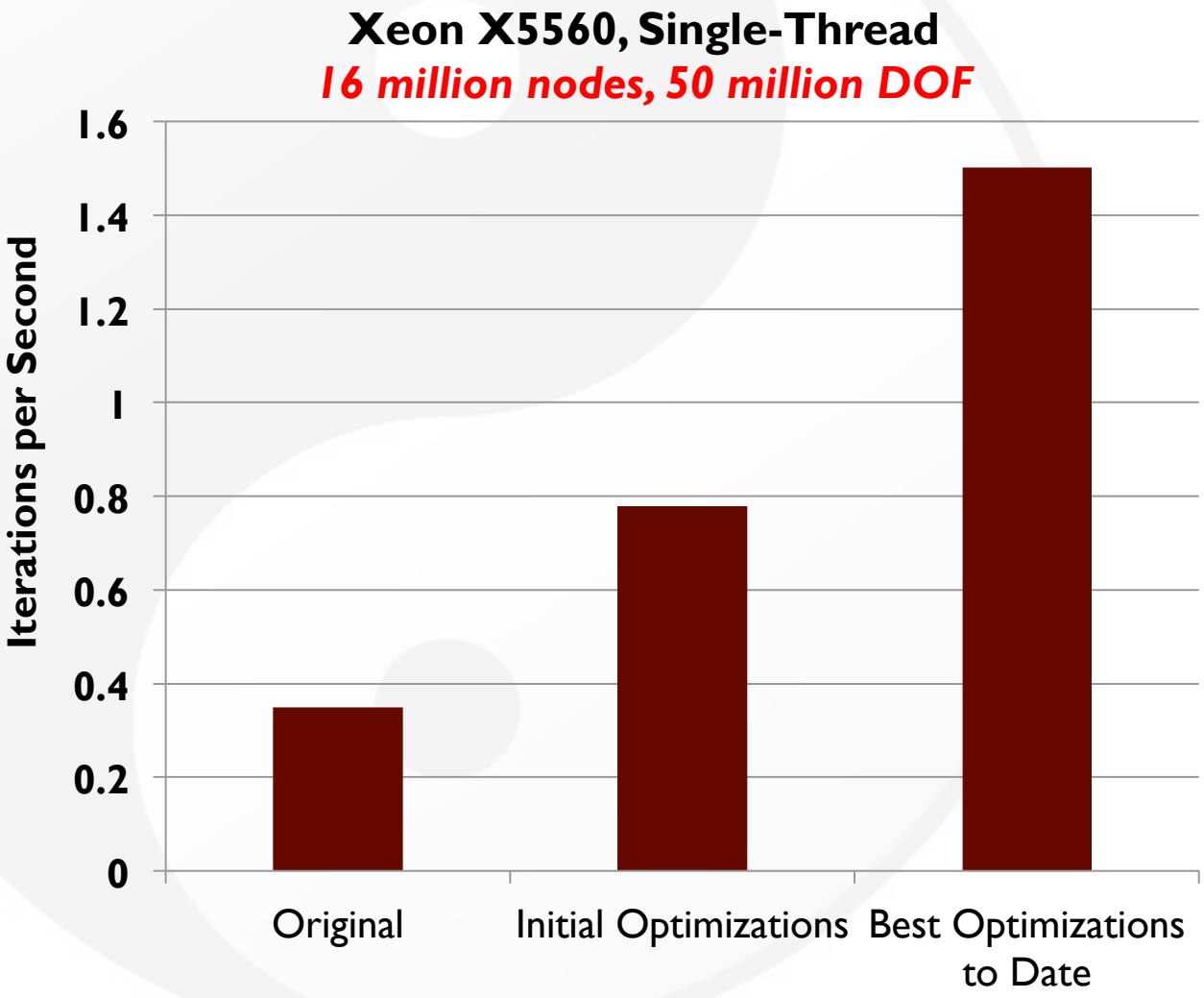
SENSEI: Overview

- SENSEI Lite (2D)
  - 2D, Cartesian, finite-difference code
  - Artificial compressibility
  - Explicit solver
  - CPU and GPU Optimization completed
- SENSEI (3D)
  - 3D, multiblock, finite-volume code
  - Artificial compressibility
  - Explicit and implicit solvers
  - Modern Fortran 2003/08 implementation
  - Challenge: Dynamically allocated arrays inside Fortran-derived types

Tradeoff between programmability, portability, and performance

Improving CPU Performance

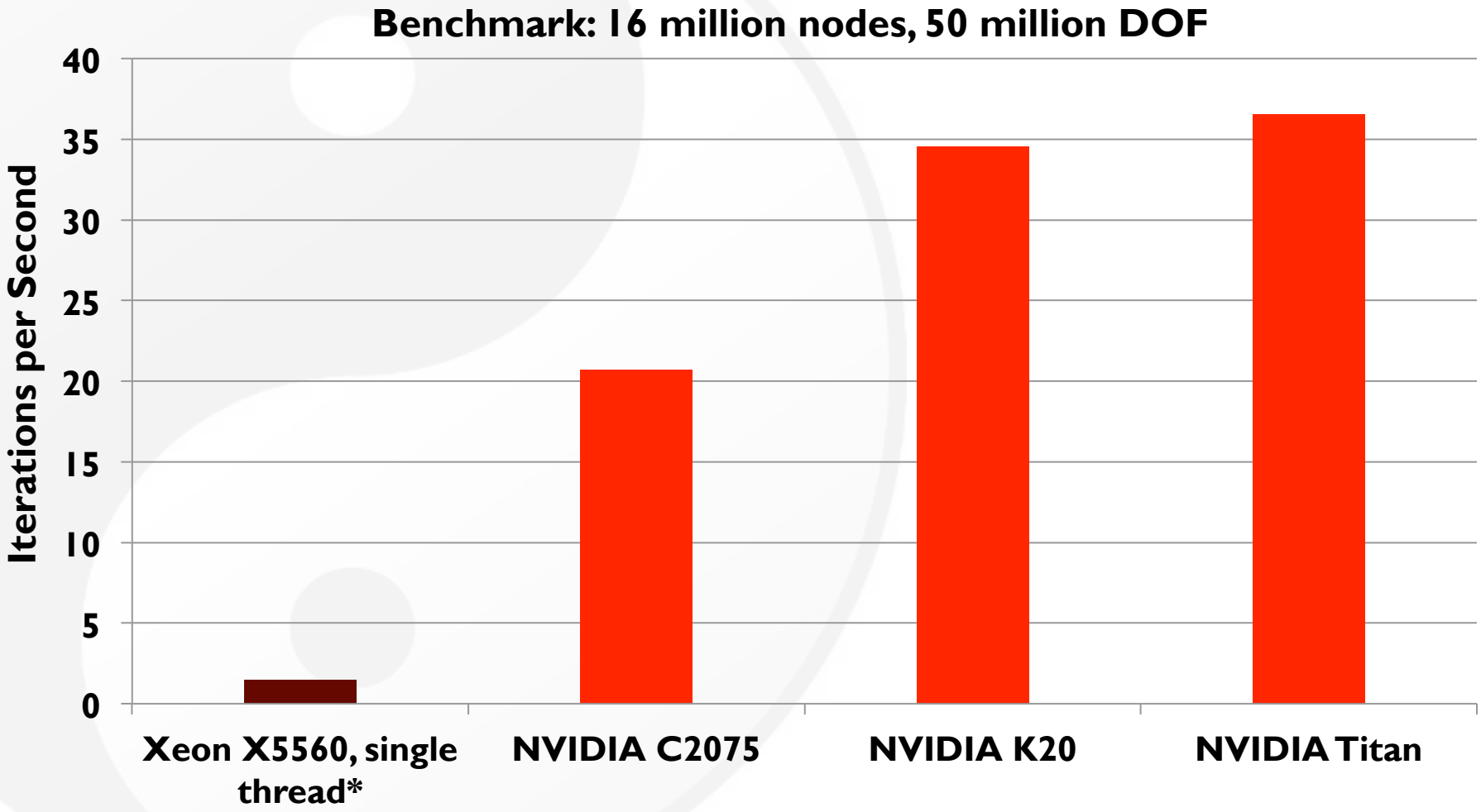
Prior to porting code to GPU, maximize performance on CPU



- **Original:** A verified finite-difference solver for the incompressible Navier-Stokes equations.
- **Initial Optimizations:** High-level code restructuring—eliminated temporary arrays, avoided a memory copy, fused multiple loops.
- **Best Optimizations to Date:** Includes SSE2 vectorization, data structure padding and loop blocking, in addition to previous optimizations.

Performance of SENSEI Lite

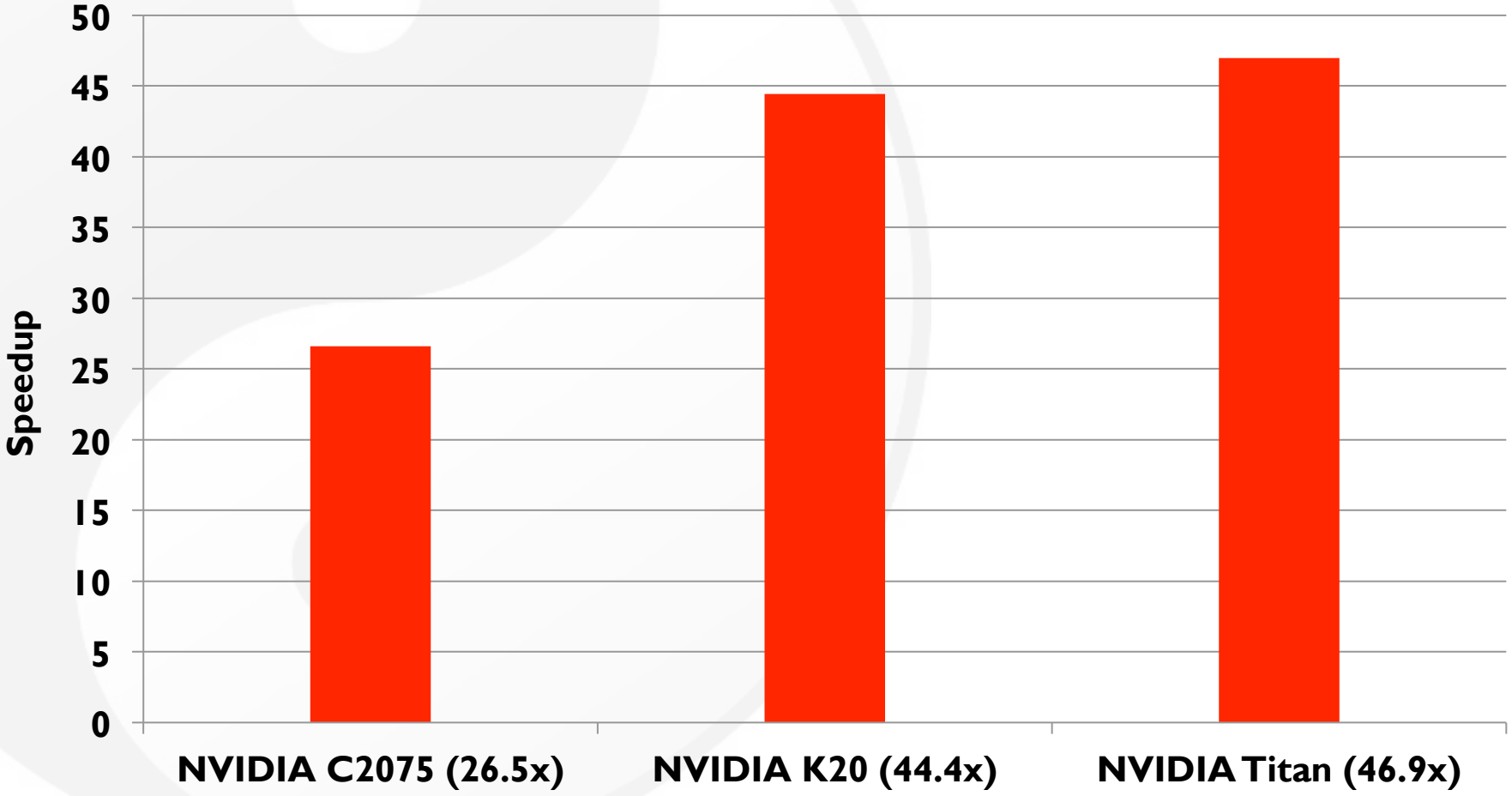
CPU vs Various GPU Architectures



***CPU uses the best performing vectorized code from the previous slide.**

Performance of SENSEI Lite

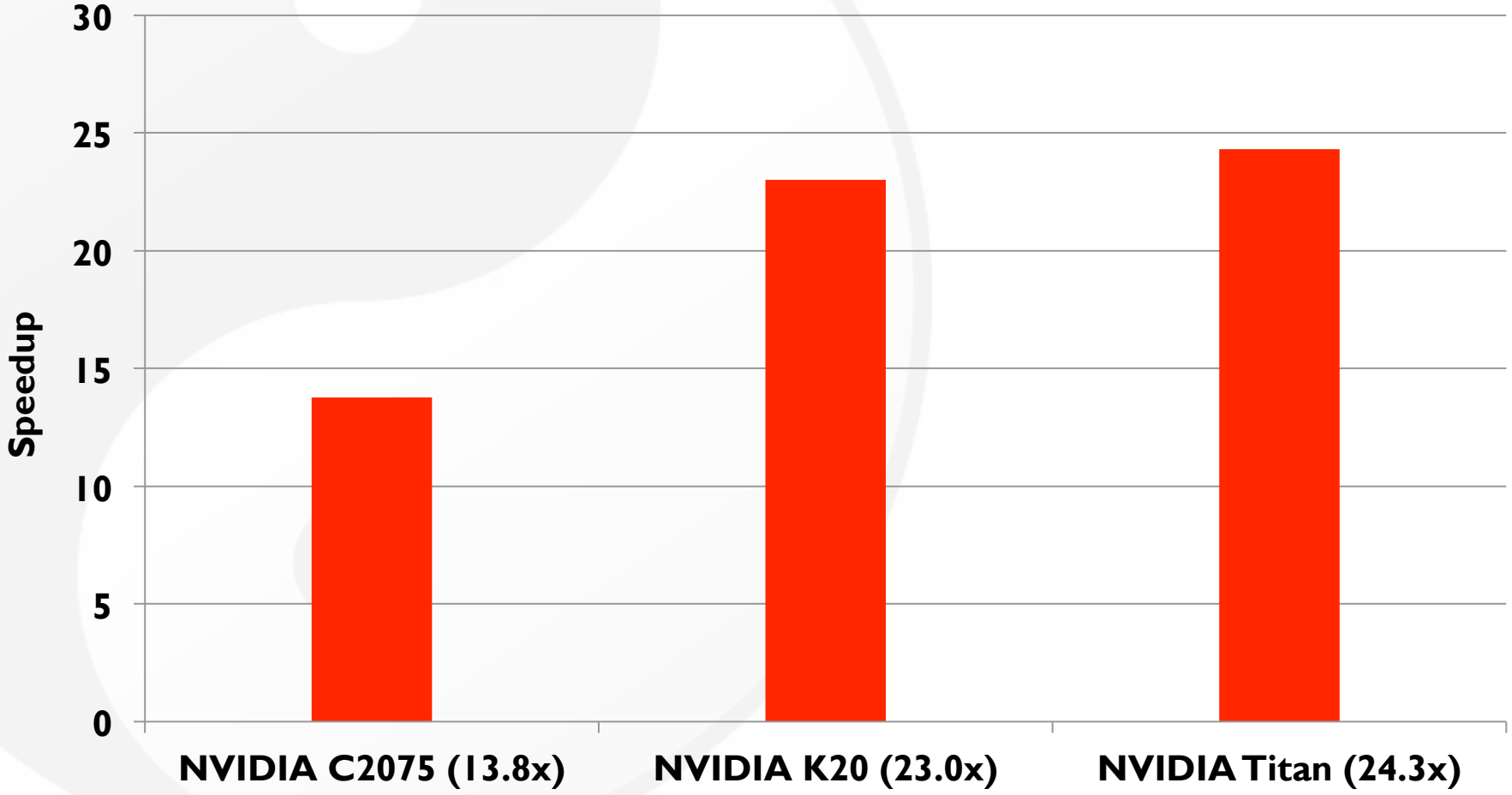
GPU speedup over *non-vectorized* single-thread CPU version
CPU = Xeon X5560 @2.8 GHz



Performance of SENSEI Lite

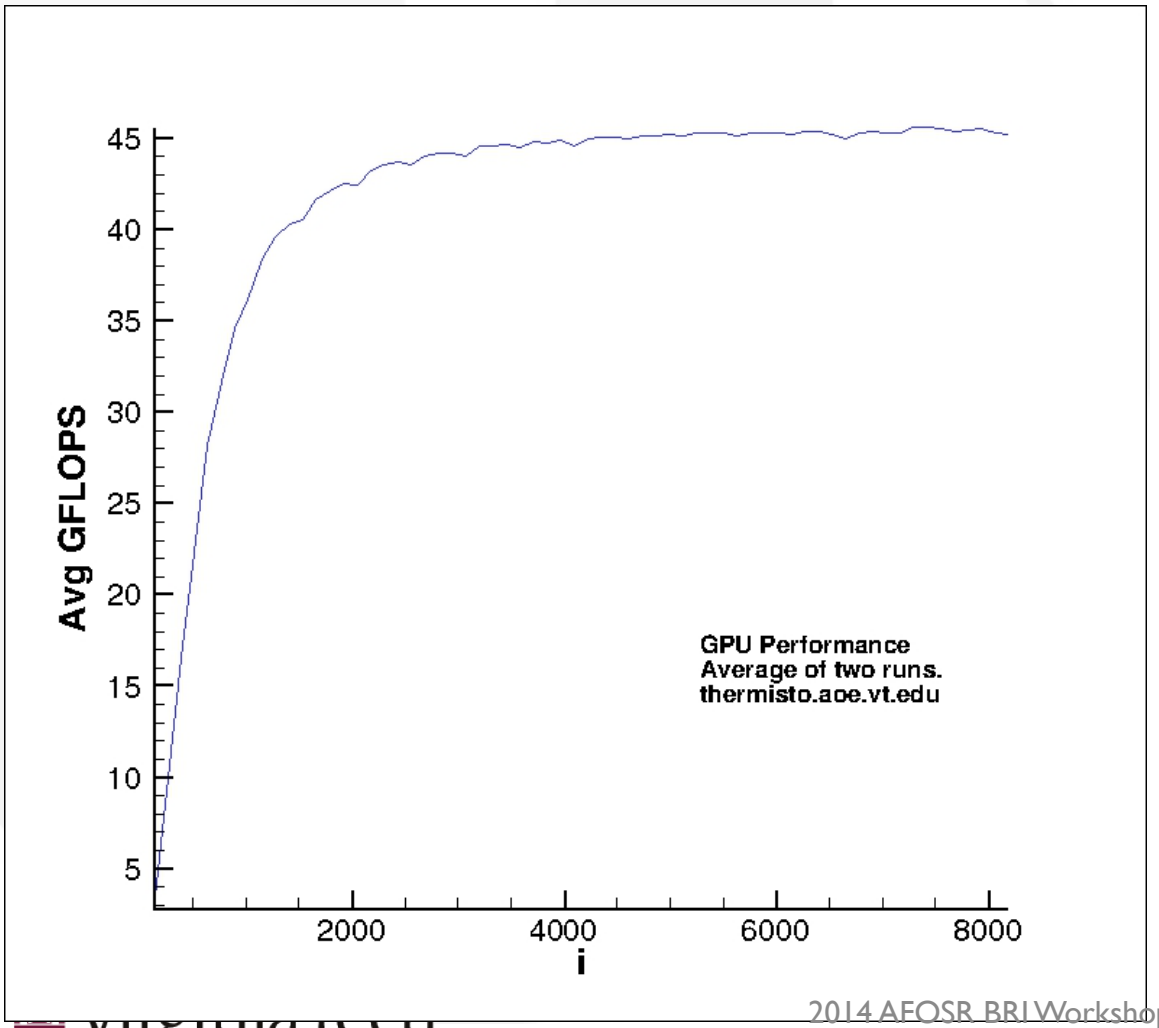
GPU speedup over *fastest (SSE vectorized)* single-thread CPU version

CPU = Xeon X5560 @2.8 GHz



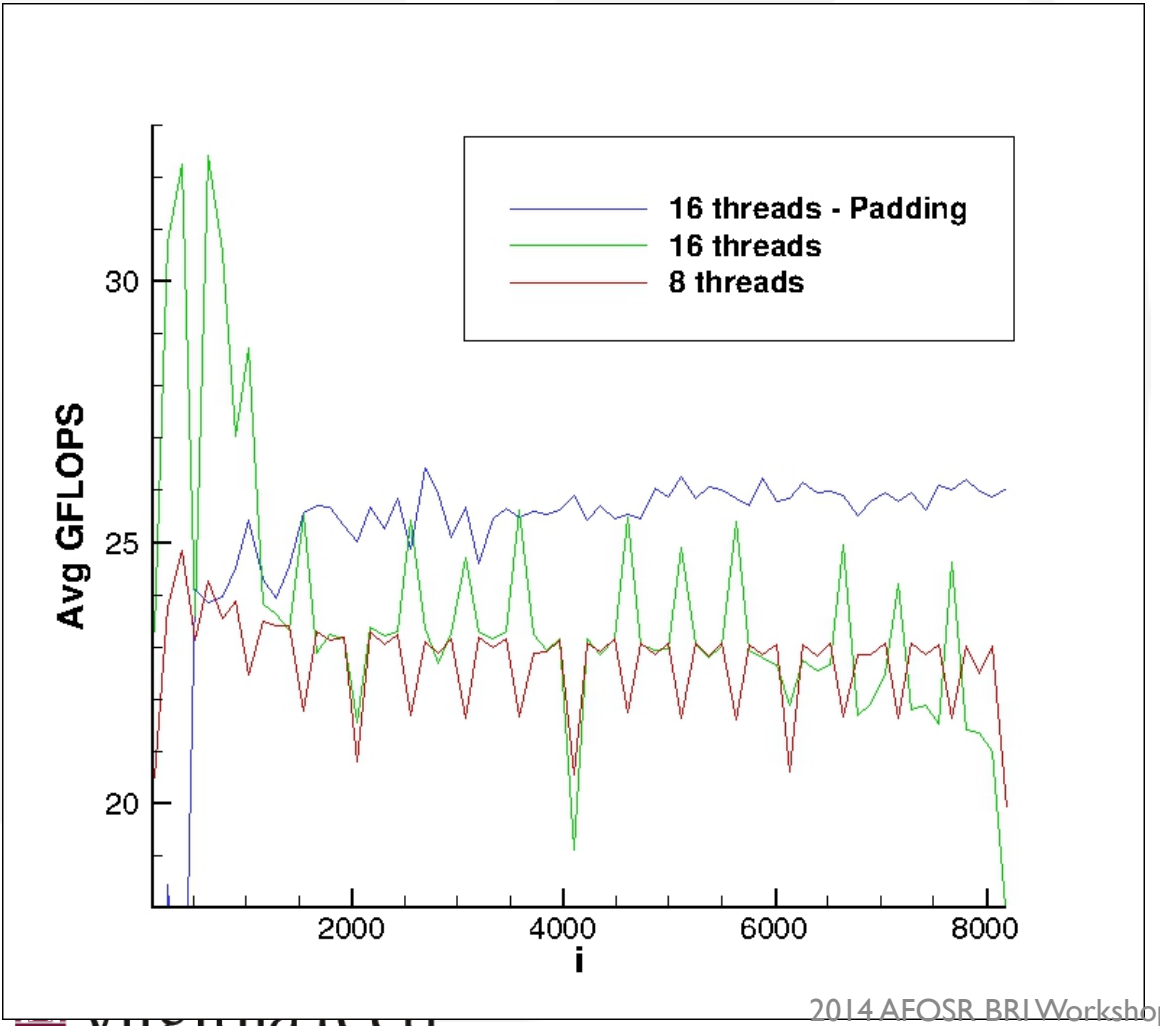
Performance of SENSEI Lite Code:

C2075 GFLOPS vs Grid Dimension



The performance on the GPU ramps up gradually as problem size is increased, with maximum throughput attained on grids larger than 2000x2000 (12 million DOF). Once the problem size is sufficiently large, the performance is steady around 45 GFLOPS.

Performance of LDC Code: X5560 GFLOPS vs Grid Dimension



Performance on the CPU can be very high for small problem sizes that fit entirely in the last-level cache, however once the solution data has increased beyond the L3 cache size the throughput is reduced. Periodic oscillations in the performance plot can be observed—we believe these are due in part to cache conflict misses. By padding the data structure it is possible to take advantage of the "peaks" and achieve a consistent higher performance level.

SENSEI Next Steps

Increasing Performance of LDC on GPU

- Specifying the gang/worker/vector parameters in the PGI Accelerator is possible—could be used for performance tuning of LDC code.

- We will use an existing script to modify and recompile source code parameters, automating the profiling process.

- Generate plots of performance vs block-size (and other parameters) for various GPU architectures (e.g., Fermi, Kepler) to determine optimum kernel launch conditions.

- Use CUDA profiler to ensure that PGI Accelerator is correctly caching stencil data in shared memory.

Targeted CFD Codes

SENSEI (C. Roy, Virginia Tech)

- Structured, multiblock, 2nd order, finite volume code
- Artificial compressibility method
- Arbitrary Lagrangian/Eulerian (ALE) 2nd or higher order spatial accuracy
- Artificial compressibility (AC) and immersed boundary (IB) methods

GenIDLEST (D. Tafti, Virginia Tech)

- Structured, multiblock, 2nd order, finite volume code
- Pressure projection method
- ALE and immersed boundary methods (IBM)

RDGFLO (H. Luo, NCSU)

- Unstructured, discontinuous Galerkin (DG) method
- High-order solution of compressible flows

INCOMP3D (J. Edwards, NCSU)

- Structured, multiblock finite volume code
- Second or higher order spatial accuracy
- ALE and IBM

GPU GenIDLEST: Preliminary Profiling

- Performance Breakdown

Time %	Component
44%	kernel_pc_jac_blk2_pc_ortho
19.3%	***GPU-CPU Data Transfers***
7.32%	kernel_pc_jac_glb2_ortho
7.13%	kernel_matxvec2_ortho
….	….

- Data marshaling performed on GPU, but …
  - Data transfers still ~ 20% of execution time
  - Maximum speedup achievable: 5x! (Amdahl's law)
    - Better overlapping of computations with transfers → MPI-ACC

GPU GenIDLEST: Issues

- Kernels operating at low occupancy
  - 52% @ 75% occupancy, 33% @ 25% occupancy, 15% @ < 25% occupancy
  - Manual optimization techniques: (1) Improved register usage, (2) optimize use of in-kernel resources, and (3) concurrent kernel execution
- Reductions performed on CPU
  - Entails transferring data back to CPU for reduction
  - Efficient reduction algorithms available for GPU → port reduction to GPU
- Linear algebra kernels – saxpy, daxpy, matrix-vector multiply
  - 15% of execution time
  - Benchmark performance against GPU BLAS libraries
- Sliced array data transfers in CUDA Fortran
  - Results into multiple cudaMemcpy calls for a single data transfer. OUCH!

Targeted CFD Codes

SENSEI (C. Roy, Virginia Tech)

- Structured, multiblock, 2nd order, finite volume code
- Artificial compressibility method
- Arbitrary Lagrangian/Eulerian (ALE) 2nd or higher order spatial accuracy
- Artificial compressibility (AC) and immersed boundary (IB) methods

GenIDLEST (D. Tafti, Virginia Tech)

- Structured, multiblock, 2nd order, finite volume code
- Pressure projection method
- ALE and immersed boundary methods (IBM)

RDGFLO (H. Luo, NCSU)

- Unstructured, discontinuous Galerkin (DG) method
- High-order solution of compressible flows

INCOMP3D (J. Edwards, NCSU)

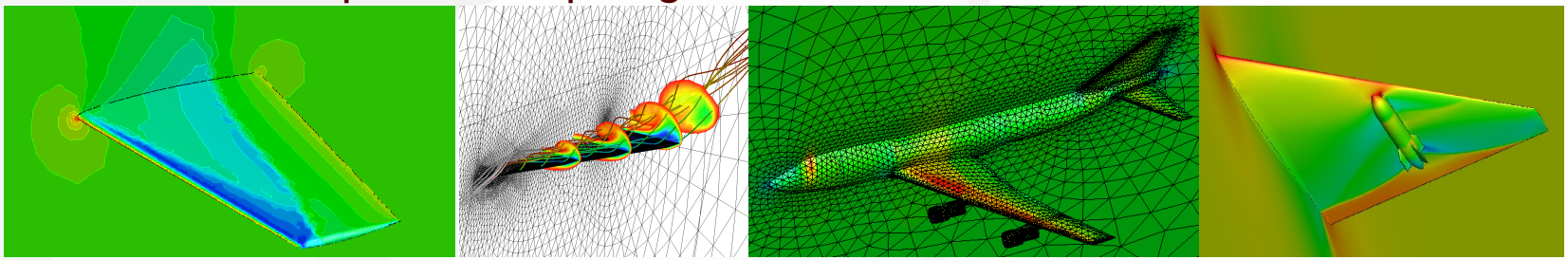
- Structured, multiblock finite volume code
- Second or higher order spatial accuracy
- ALE and IBM

RDGFLO: Overview

Reconstructed Discontinuous Galerkin Flow Solver

Key Features

- Compressible Navier-Stokes / Euler equations.
- Third-order reconstructed discontinuous Galerkin (DG) finite element method.
- Unstructured hybrid grids, i.e., tetrahedron, prism, pyramid, hexahedron.
- Time-accurate and steady-state solution schemes.
- MPI-based parallel computing on CPU clusters.



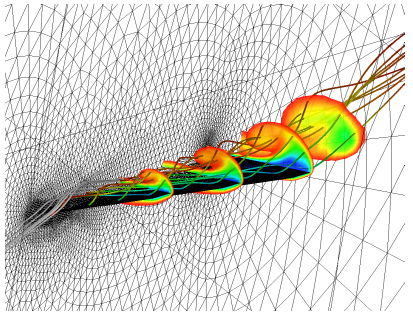
Need GPU acceleration for RDGFLO because …

High-order methods are expensive for large-scale problems in terms of computing time!

Rewriting a huge legacy code using CUDA is too costly. Alternatively, …

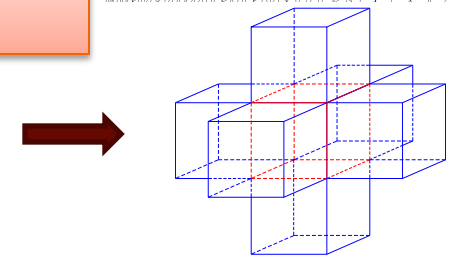
OpenACC does not require much change of data structures and algorithms in a legacy code.

RDGFLO: GPU Parallelization

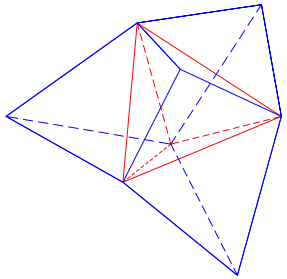


Race condition occurs in GPU parallelization in loops over faces when data writes to their left and right cell arrays.

Example: the elemental residual array for the cell (red) can be overwritten simultaneously in multiple GPU cores in loops over its faces by its face-neighboring cells (blue).



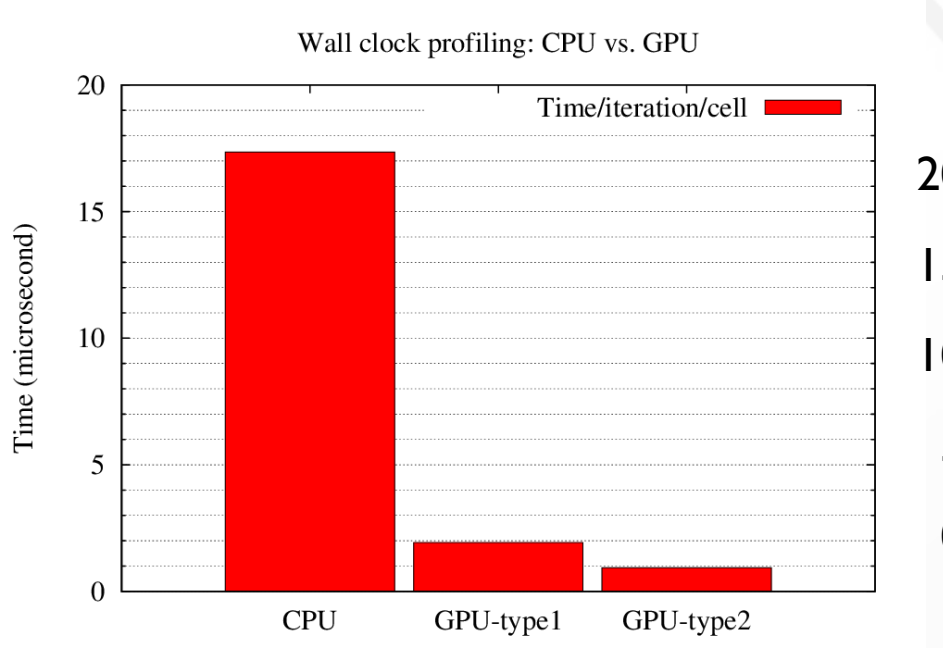
Coloring algorithm: reorder face indices and pack them in groups; Criterion: faces that share common elements do not reside in the same group (the same strategy in the case of OpenMP).

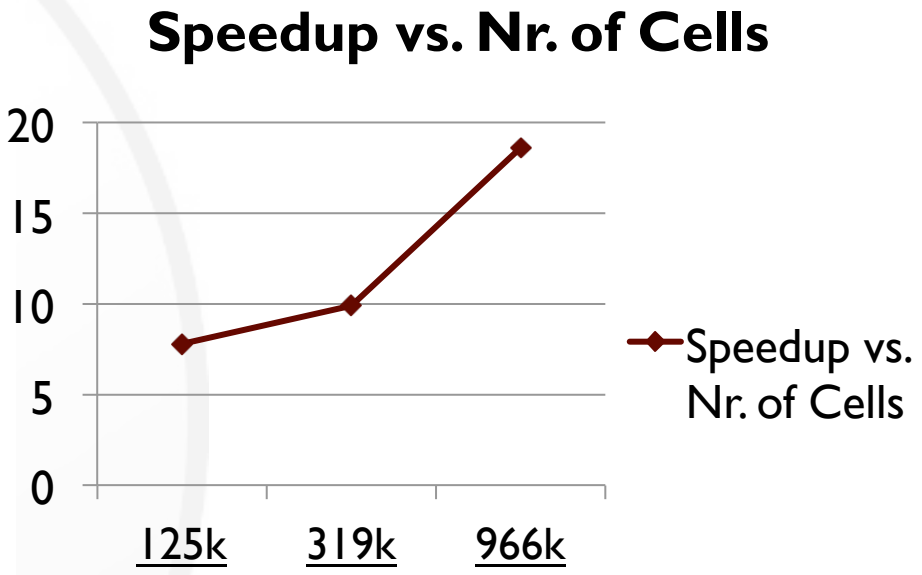


Example: an ACC sequential loop over the groups is nested outside the ACC parallel loop over the faces (the original loop is untouched).

All geometric and solution arrays are copied from host to device only once. For steady-state problems, solution arrays are only copied back to host memory at the end of time iterations and dumped in files.

RDGFLO: Weak Scaling Tests





Unit timings on the grid of 966k cells
-- CPU: AMD Opteron Processor 6128
-- GPU-type1: nVidia Tesla C2050
-- GPU-type2: nVidia Tesla K20c

Preliminary Result: A speedup factor of 20x (or more) is achievable.

Targeted CFD Codes

SENSEI (C. Roy, Virginia Tech)

- Structured, multiblock, 2nd order, finite volume code
- Artificial compressibility method
- Arbitrary Lagrangian/Eulerian (ALE) 2nd or higher order spatial accuracy
- Artificial compressibility (AC) and immersed boundary (IB) methods

GenIDLEST (D. Tafti, Virginia Tech)

- Structured, multiblock, 2nd order, finite volume code
- Pressure projection method
- ALE and immersed boundary methods (IBM)

RDGFLO (H. Luo, NCSU)

- Unstructured, discontinuous Galerkin (DG) method
- High-order solution of compressible flows

INCOMP3D (J. Edwards, NCSU)

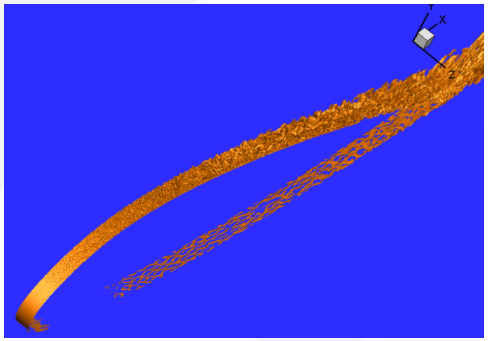
- Structured, multiblock finite volume code
- Second or higher order spatial accuracy
- ALE and IBM

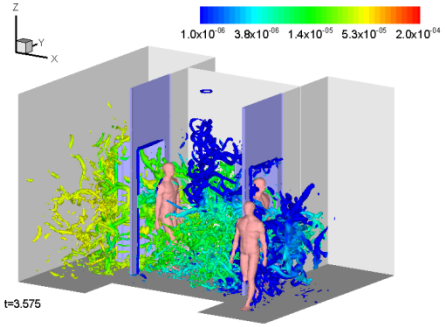
INCOMP3D: Overview

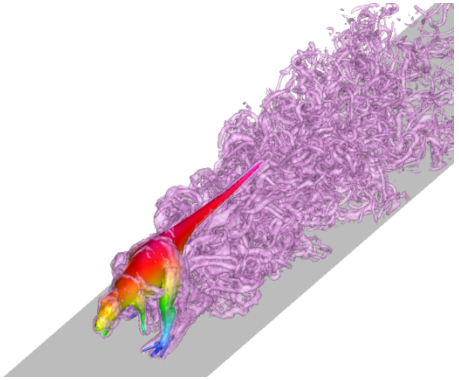
Multi-block Incompressible Navier-Stokes Solver for Large Eddy Simulation

Key Features

- Higher order PPM / central-difference schemes
- Fully implicit time evolution using dual-time stepping methodology
- Multi-block structured meshes (MPI parallelism)
- Immersed boundary methods for complex motion events

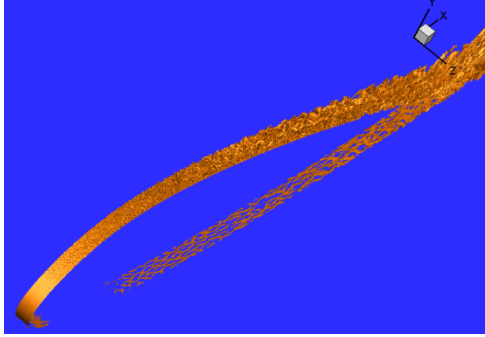






Need GPU acceleration for INCOMP3D to reduce costs associated with large-eddy simulation at high Reynolds numbers

INCOMP3D: GPU Parallelization

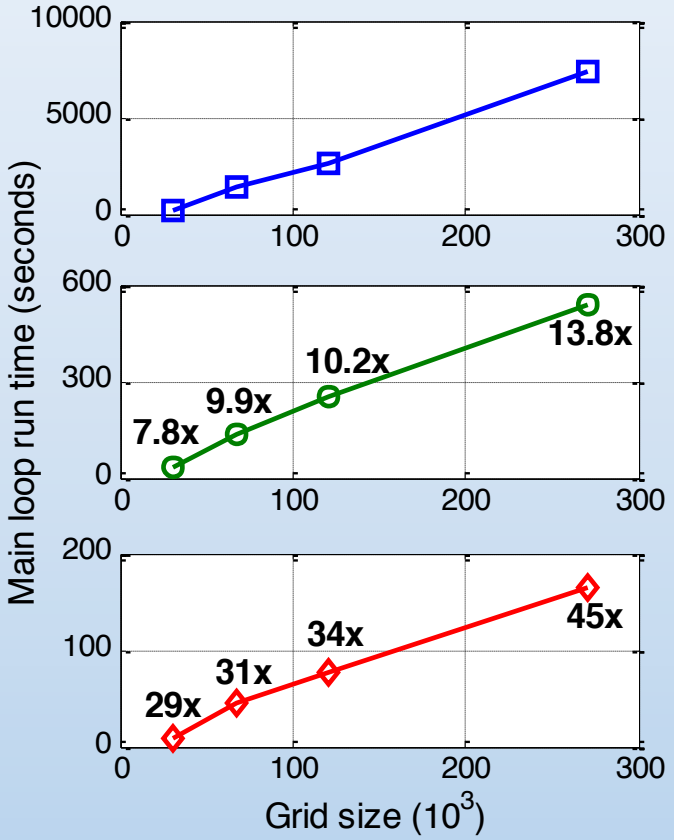


- Two scaled-down versions realized
- OpenACC (ACC) port
  - Main loop is carried out on GPU only. All essential arrays remain on GPU main memory. Temporary data arrays are created directly on GPU.
  - One code can be compiled into pure CPU or GPU-accelerated versions, using different compiler options. This facilitates long-term maintenance.
- CUDA Fortran (CUF) port
  - Each time step are carried out by one monolithic kernel, which includes residual calculation and time marching.
  - Residual array is directly created in shared memory; time step is local to each thread. Memory requirement is greatly reduced.
  - Overlapping blocks are used, due to inter-thread data dependency.
  - Residual calculation involves flux grouping by direction (i and j directions), to avoid memory contingency.
  - CUDA Fortran array overhead identified and solved using global variables.

INCOMP3D: Initial Findings

- ACC and CUF both achieved significant speedup over CPU
- CUF achieved better performance, but requires much more effort to port and maintain
  - Direct access of shared memory allows greater flexibility on algorithms.
  - Easier to make mistakes; harder to debug.
- ACC provides a good compromise between CPU and CUDA
  - Good speedup (~10x), with minimal to moderate effort on porting.
  - Easier to debug and maintain.

Example: steady-state flow inside a channel with 3 circular obstacles, Re=200. All results (in seconds) are obtained using nVidia c2050.

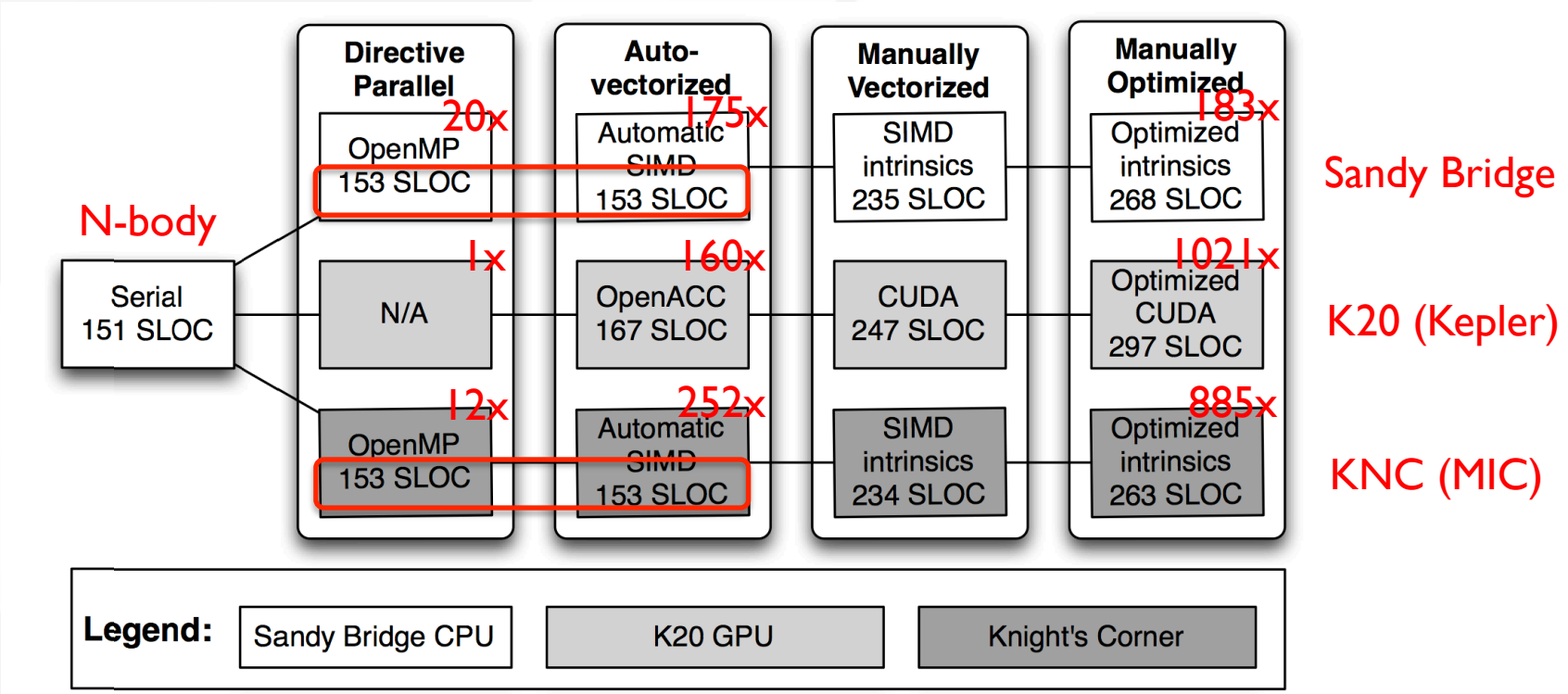


Co-Design Around Three P's: Performance, Programmability, Portability

"Productivity = Performance + Programmability + Portability"

- Multi-dimensional optimization across two or more P's

  … first *manual co-design* … then *automated co-design*



RDGFLO and INCOMP3D: Next Steps

- Further Porting of RDGFLO
  - Hierarchical WENO reconstruction, implicit time integration and turbulence model (LES)
- Porting of the full version of INCOMP3D
  - 3D LDFSS, implicit time integration, full IB support and turbulence model (LES)
- Biggest hurdle: multi-GPU MPI communication
  - Avoid explicit GPU-CPU transfer during MPI data exchanges.
  - A CUDA-aware MPI implementation (currently MVAPICH2) is used, which can take advantage of the best implementation available (GPUDirect, hardware RDMA in CUDA5).
  - Current MPI Fortran interface does not support operations on OpenACC variables directly. Manual data packing on GPU using explicit CUDA programming is still required in OpenACC codes.

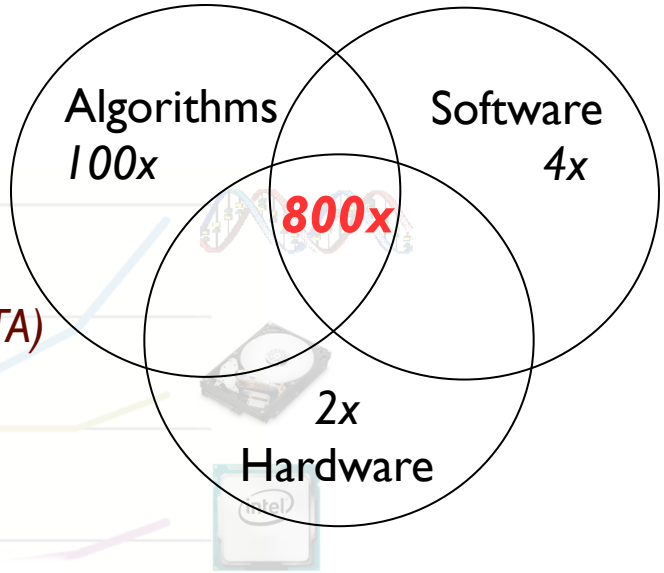
Year One: Synergistic Co-Design

- A slide that is an adaptation of the next slide (presented at White House for "BIG DATA") but customized for structured and unstructured grid codes? See next slide
- Too early?

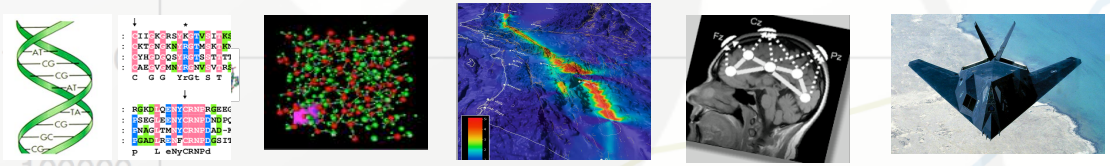
Synergistic Co-Design for BIG DATA

Prof. Wu Feng, Virginia Tech

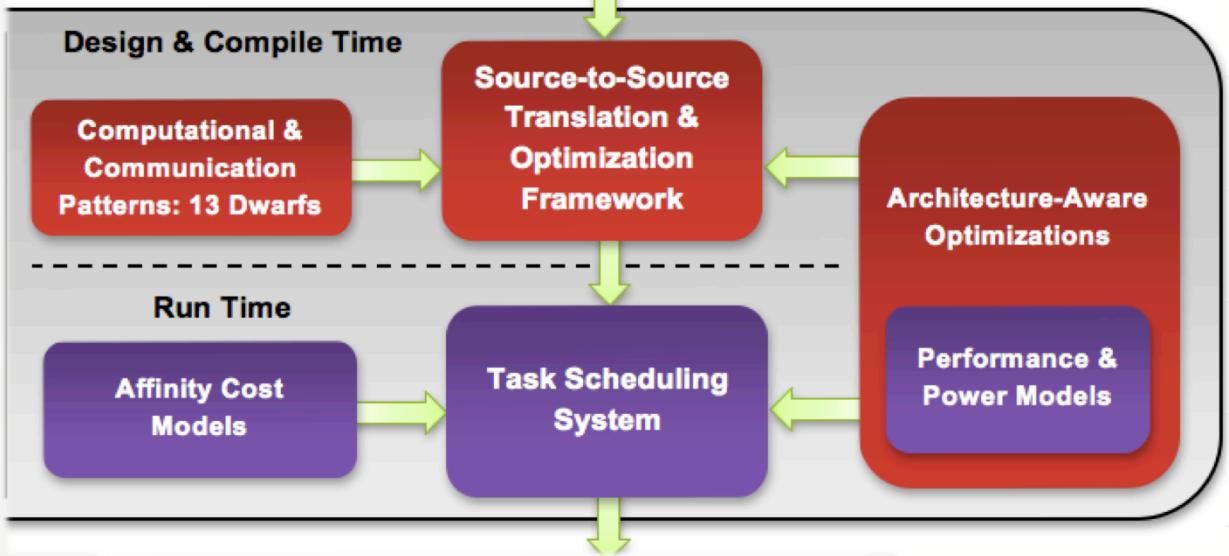
- Changing Landscape
  - … from *FLOPS ("old HPC") to bytes ("new HPC" → BIG DATA)*



Apps



Software Ecosystem



Hardware

Co-Design Exemplars

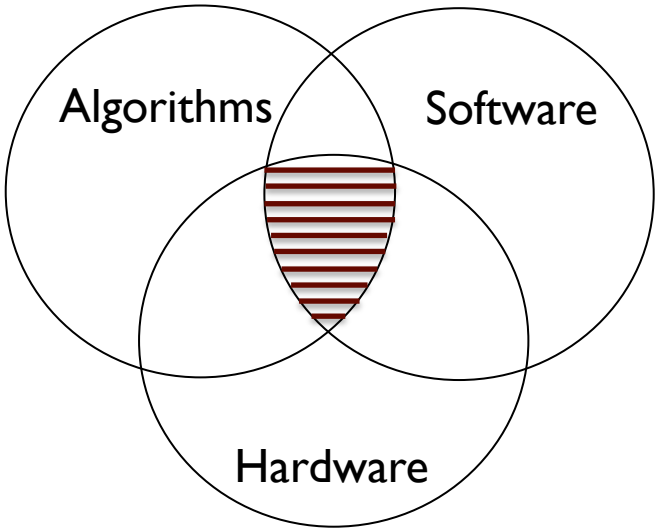
1. *ParaMEDIC: Parallel Metadata Environment for Distributed I/O & Computing (yrs → mins)*
   → Find missing genes
   http://archive.isgtw.org/?pid=1000811

2. *Molecular Modeling*
   → Rational drug design
   http://www.youtube.com/watch?v=zPBFenYg2Zk

3. *Temporal Data Mining of Brain*
   http://synergy.cs.vt.edu/pubs/papers/feng-temporal-data-mining-gpu-gems-2011.pdf

Roadmap

- Vision
- Team
- Approach
- Infrastructure
- Co-Design Research
  - Computer Science
  - CFD Codes (4)
  - Mathematics (de Sturler, Sandu)
- Achievements & Publications
- Next Steps

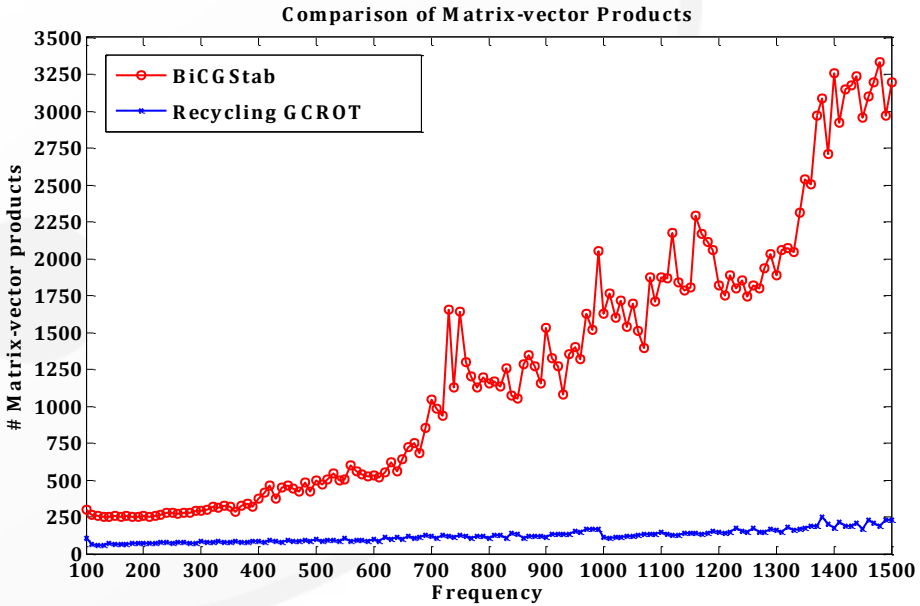


Solvers on GPUs and Multicore Processors

- Goal: Minimum Total Execution Time (vs Accuracy)
  - Fewer *expensive* iterations vs more *cheap* iterations
- Current General Effort
  - Efficient implementation of each kernel (data layout)
  - Efficient implementation of standard preconditioners
  - Better performance by varying parameters/preconditioners, memory-intensive algorithms have poor cache performance
- Future Efforts
  - Combine kernels/iterations, latency hiding, more arithmetic vs. data movement
  - Vary precision, analyze accuracy, and convergence
  - Alternative preconditioners
    - Faster convergence, better hardware utilization, data storage/access
    - Modify algorithms (solvers/preconditioners) – need to maintain good convergence

New(er) Solvers with Better Trade-offs

- Faster convergence (fewer iterations) for additional orthogonalizations and upfront matvecs, but all at once
  - Higher level BLAS/more work per data movement – trade off with sparse matvec
  - Better convergence allows cheaper preconditioner
  - Only one sync per extra vector in space
  - Especially useful for sequences of linear systems



Acoustics problem: Tire noise

Collaboration Jan Bierman, BMW

Preconditioner Effort

- Develop preconditioners with good GPU performance and fast convergence
- Avoid triangular solves (in ILU type prec.s)? Or improve performance by alternative storage schemes?
- Prec.s that use SpMV - Sparse Approximate Inverses
  - Improve convergence by multilevel correction
  - Multigrid-like preconditioners (AMG)
  - Domain decomposition preconditioners? (but without local direct solve)
- Computing preconditioners is expensive
  - Update preconditioners for sequence of problems

New Rosenbrock-Krylov Matrix-Free Integrators Using Minimal Implicitness

- New implicit matrix-free Rosenbrock-Krylov time integration methods avoid solution "to convergence" and guarantee order of consistency with small Krylov spaces (e.g., four-dimensional)
- Full-space linear algebra parallelized using cuBLAS. Arnoldi iteration parallelized with a custom CUDA kernel for the orthogonalization step.
- Tested on a parallel implementation of the shallow water equations using different grid sizes and tolerances.

Rosenbrock-Krylov Methods
Suited for Parallelization and Acceleration

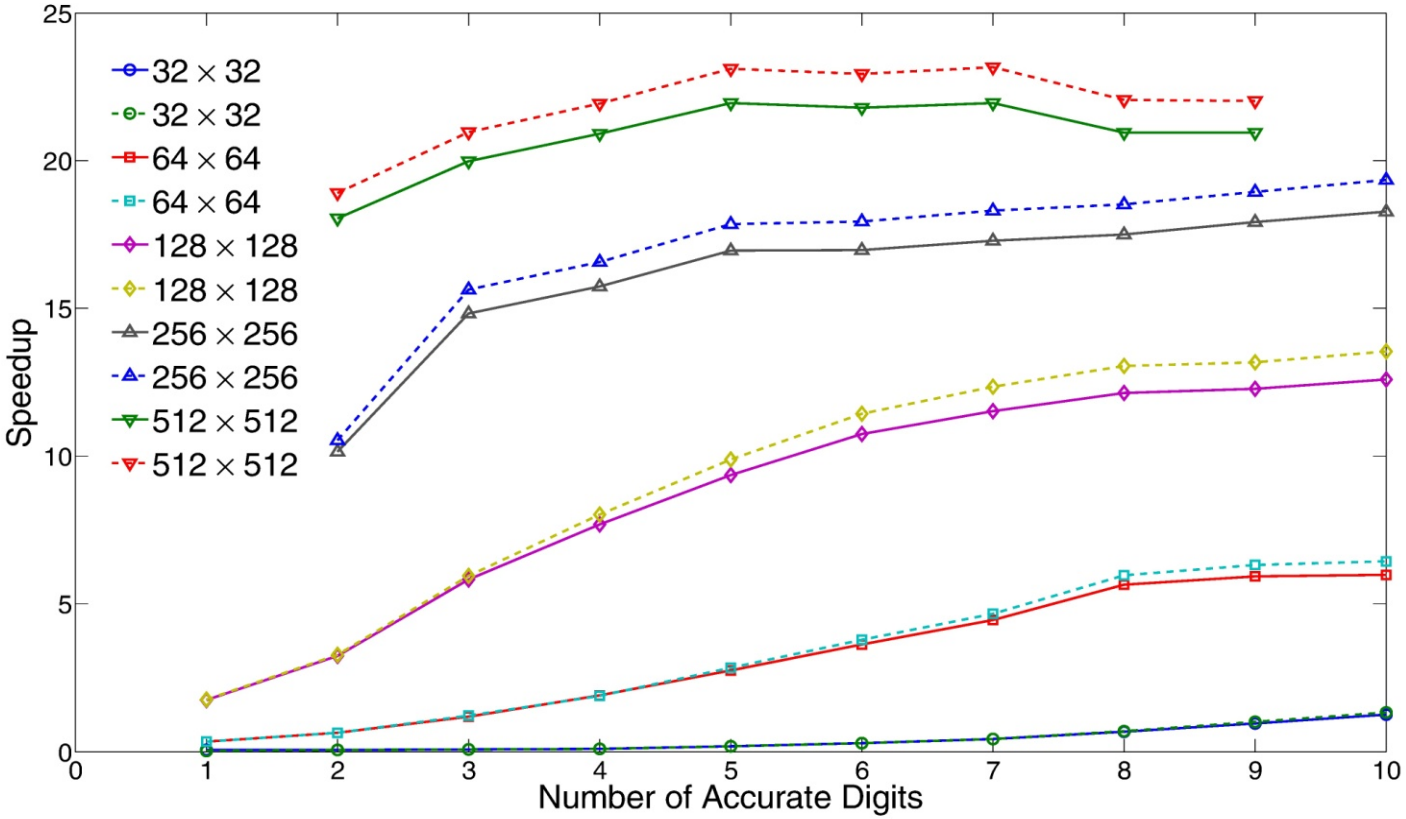
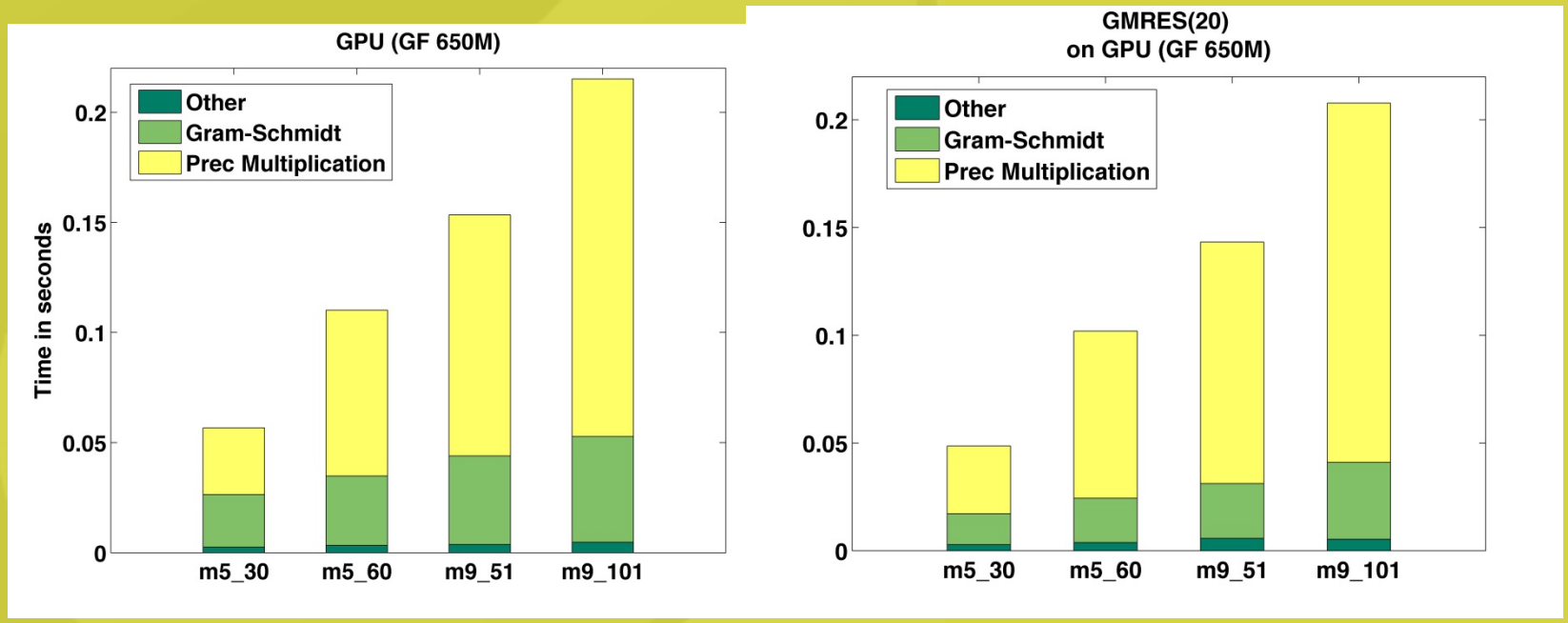


Figure: Speedup of GPU accelerated ROK-SWE over serial.
Dashed lines use a custom kernel for the Arnoldi, while solid use cuBLAS.
(GPU=NVIDIA Quadro 4000, CPU=Intel Xeon 2.5GHz).

Performance Issues – SENSEI Lite

- With good preconditioner (convergence) GMRES(20) and GMRES(40) converge about equally fast
- GMRES(20) much lower cost in orthogonalizations
- However, preconditioner cost dominates
- Note huge difference SpMV (in other) and Prec (BILUT)



GMRES(40) GMRES(20)

Achievements

- CS
  - Compile/Design Time: Infer speedup potential before refactoring code
    - Memory trace compression algorithms to estimate parallelization benefits
  - At Run Time: Initial evaluation of our automated run-time system prototype (for Accelerated OpenMP and OpenACC) → vendors
  - Identified commonality for library for CFD codes: generalized GPU-to-GPU communication (via MPI), ghost cell exchange between GPUs, …
- CFD
  - Preliminary GPU parallelization of prototype CFD codes with OpenACC and CUDA
  - Up to 40x speed-up over a single CPU core (25x over SSE vectorized)
- Math
  - Initial GPU-parallelized Rosenbrock-Krylov method
  - Integrated initial CUDA-parallelized solvers with CFD

Publications

- P. Tranquilli, A. Sandu, "Rosenbrock-Krylov Methods for Large Systems of Differential Equations" http://arxiv.org/abs/1305.5481, May 2013.
- J. M. Derlaga, T. S. Phillips, C. J. Roy, "SENSEI Computational Fluid Dynamics Code: A Case Study in Modern Fortran Software Development," AIAA Paper 2013-2450, *21st AIAA Computational Fluid Dynamics Conf.*, June 2013.
- S. R. Glandon, P. Tranquilli, A. Sandu, " Acceleration of matrix-free time integration methods", *Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA) at SC13*, November 2013.
- B. P. Pickering, C. J. Roy, T. R. W. Scogland, W. Feng, "Directive-Based GPU Programming for Computational Fluid Dynamics," *52nd AIAA Aerospace Sciences Meeting*, January 2014.
- Y. Xia, L. Luo, H. Luo, J. Edwards, F. Mueller, 'GPU Acceleration of a Reconstructed Discontinuous Galerkin Method for Compressible Flows on Unstructured Grids", *52nd AIAA Aerospace Sciences Meeting*, January 2014.
- L. Luo, J. R. Edwards, H. Luo, F. Mueller, "Performance Assessment of Multi-block LES Simulations using Directive-based GPU Computation in a Cluster Environment," *AIAA SciTech 2014*, January 2014.

What's Next?

- Platforms
  - AMD & Intel CPU, AMD APU, AMD & NVIDIA GPUs, Intel MIC
- Towards Ease of Use and Automation (for Performance, Programmability, and Portability)
  - Web resource for *tenets of synergistic co-design* … between algorithms, software, and hardware → automation (long term)
  - Towards a CFD library for heterogeneous computing systems
    - GPU-integrated MPI vs. GPUDirect, ghost cell exchange, bounds checking, …
  - Code repositories for production codes
- GPU-Integrated MPI Evaluation
  - Experimental platforms (MIC and next-generation APU w/ "infinite memory")
- GPU mixed-precision solvers, GPU-efficient preconditioners
- GPU-efficient accurate and stable high-order time stepping

Ongoing and Upcoming Tasks

- Further Porting of RDGFLO
  - Hierarchical WENO reconstruction; Implicit time integration
  - Large eddy simulation
- Porting of the full version of INCOMP3D
  - 3D LDFSS, implicit time integration
  - Full IB support, turbulence model (LES) and multi-phase, reacting fluids
- Multiple GPUs with MPI communication
  - Avoid explicit GPU-CPU transfer during MPI data exchanges.
  - A CUDA-aware MPI implementation (currently MVAPICH2) is used, which can take advantage of the best implementation available (GPUDirect hardware RDMA in CUDA5).
  - Mix OpenACC with CUDA-aware MPI calls. Current MPI interfaces does not support OpenACC variables directly.

Agenda

10:30-11:00 Opening Remarks and Overview of Project:
Co-Design of Hardware/Software for Predicting MAV Aerodynamics

11:00-11:30 Development of a Portable, GPU-Accelerated High-Order Discontinuous Galerkin CFD Code for Compressible Flows on Hybrid Grids

11:30-12:00 Performance Assessment of a Multi-block Incompressible Navier-Stokes Solver using Directive-based GPU Programming in a Cluster Environment

12:00-12:30 Cache Performance Prediction

12:30-13:30 Working Lunch

13:30-14:00 GPU Acceleration of the SENSEI CFD Code Suite

14:00-14:30 GENIDLEST Co-Design

14:30-14:45 Break

14:45-15:15 Accelerated Solvers for CFD

15:15-15:45 Co-design of Time-Stepping Algorithms for Large Aerodynamics Simulations

15:45-16:00 Break

16:00-16:30 Codifying and Applying a Methodology for Manual Co-Design and Developing an Accelerated CFD Library (Co-Design: CFD/Math/CS)

16:30-17:00 Tool Chain for Co-Design (Co-Design: CS/Tools)

17:00-17:30 Discussion

Acknowledgements

- This work was funded by the Air Force Office of Scientific Research (AFOSR) Computational Mathematics Program
  - Program Manager: Fariba Fahroo
  - Grant No. FA9550-12-1-0442