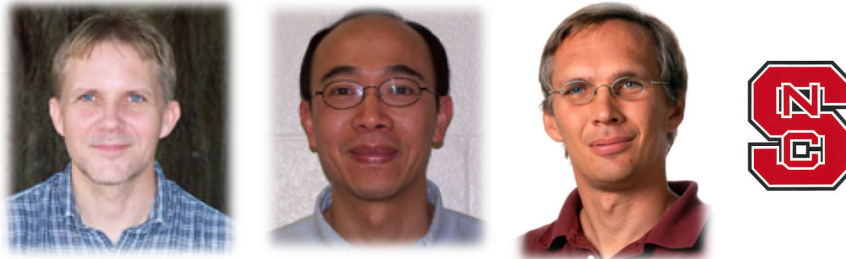


Synergistic Co-Design of Hardware and Software for Structured and Unstructured Grid Computations



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



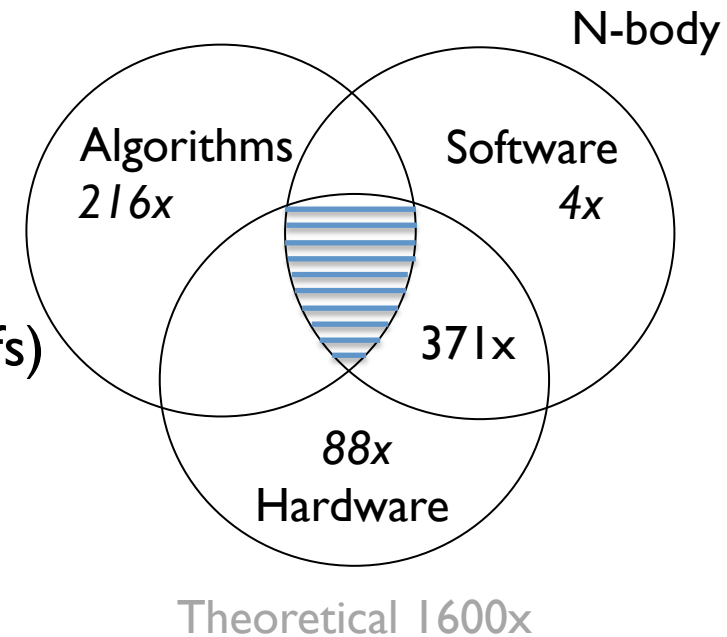
J. Edwards, H. Luo, F. Mueller

Air Force Office of Scientific Research (AFOSR) Basic Research Initiative
Transformational Computing via Co-Design of High-Performance Algorithms & HW
Computational Mathematics Program via Grant No. FA9550-12-1-0442
Program Manager: Fariba Fahroo



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 and maintainable **algorithms**
 - ... that can be mapped and optimized in **software**
 - ... onto the right type of processing core in **hardware**
 - ... at the right time
 - ... to deliver multiplicative speed-up that would *not* have possible by Moore's Law alone (e.g., 88x → 371x, as noted above)
 - Co-design feedback to vendors to guide future hardware design
 - Enabling of domain scientists and engineers to focus on their science and engineering rather than the *computer* science and engineering



Why Synergistic Co-Design?

- Exascale is coming ... why worry?

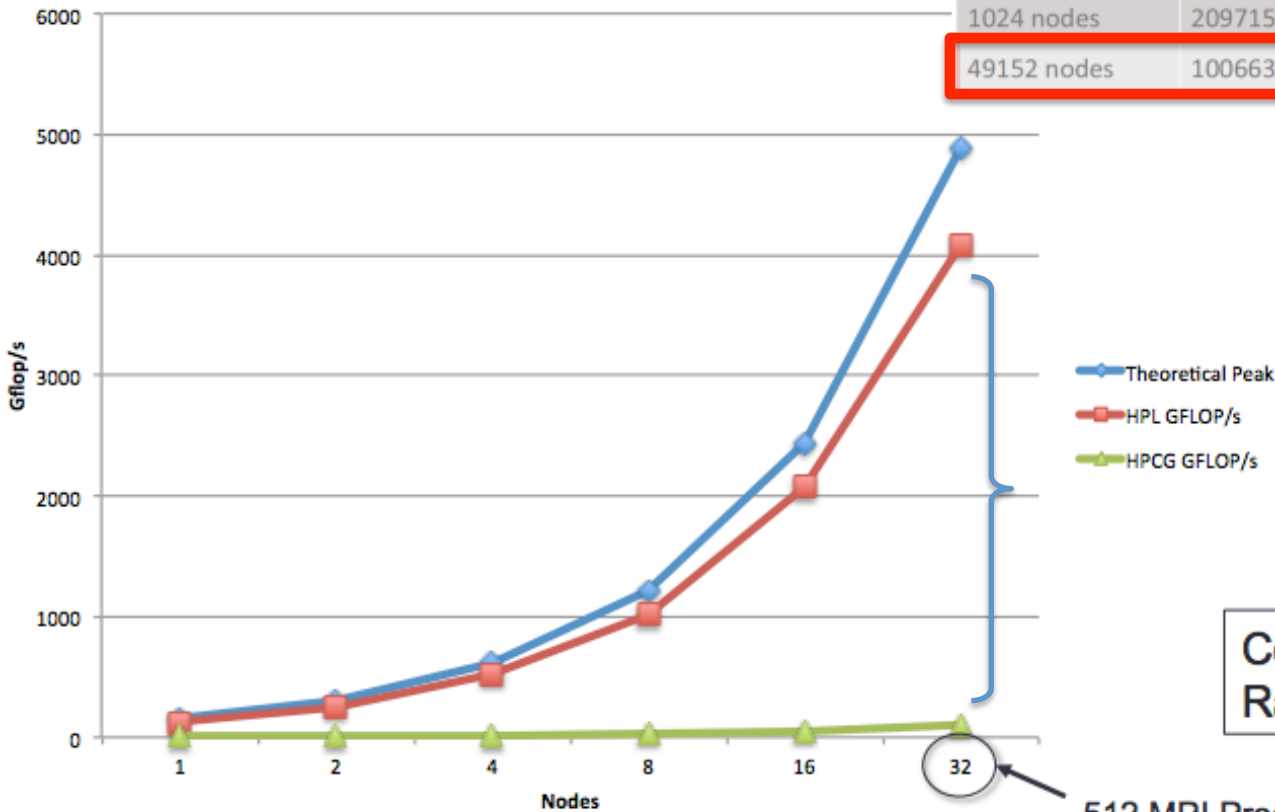
System Attributes	2010	“2015”		“2020”	
System peak	2 Petaflops	200-300 Petaflops		1 Exaflop = 1000 Petaflops	
Power	6 MW	15 MW		20-30 MW	
System memory	0.3 PB	5 PB		32-64 PB	
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW	25 GB/s	0.1 TB/sec	1 TB/sec	0.4 TB/sec	4 TB/sec
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	18,700	50,000	5,000	1,000,000	100,000
Total Node Interconnect BW	1.5 GB/s	20 GB/sec		200 GB/sec	
MTTI	days	O(1 day)		O(1 day)	

Source: Kathy Yelick (Lawrence Berkeley National Laboratory)

GFLOPS “Shock”

Mira Partition Size	Peak Gflops	Sustained Gflops	% of peak
64 nodes	13107.2	73.4	0.56%
128 nodes	26214.4	147.43	0.56%
256 nodes	52428.8	293.8	0.56%
512 nodes	104857.6	587.97	0.56%
1024 nodes	209715.2	1176.69	0.56%
49152 nodes	10066329.6	55177.6	0.55%

Results for Cielo
 Dual Socket AMD (8 core) Magny Cour
 Each node is 2*8 Cores 2.4 GHz = Total 153.6 Gflops/

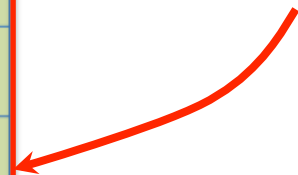


Courtesy Kalyan Kumaran, Argonne

Courtesy Mahesh Rajan, Sandia

512 MPI Processes

HPCG / HPL for TOP500 Supercomputers



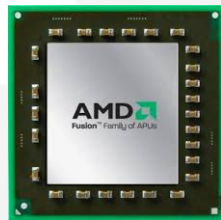
Site	Computer	Cores	HPL Rmax (Pflops)	HPL Rank	HPCG (Pflops)	HPCG/HPL
NSCC / Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom	3,120,000	33.9	1	.580	1.7%
RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx 8C + Custom	705,024	10.5	4	.427	4.1%
DOE/OS Oak Ridge Nat Lab	Titan, Cray XK7 AMD 16C + Nvidia Kepler GPU 14C + Custom	560,640	17.6	2	.322	1.8%
DOE/OS Argonne Nat Lab	Mira BlueGene/Q, Power BQC 16C 1.60GHz + Custom	786,432	8.59	5	.101#	1.2%
Swiss CSCS	Piz Daint, Cray XC30, Xeon 8C + Nvidia Kepler 14C + Custom	115,984	6.27	6	.099	1.6%
Leibniz Rechenzentrum	SuperMUC, Intel 8C + IB	147,456	2.90	12	.0833	2.9%
CEA/TGCC-GENCI	Curie tine nodes Bullx B510 Intel Xeon 8C 2.7 GHz + IB	79,504	1.36	26	.0491	3.6%
Exploration and Production Eni S.p.A.	HPC2, Intel Xeon 10C 2.8 GHz + Nvidia Kepler 14C + IB	62,640	3.00	11	.0489	1.6%
DOE/OS L Berkeley Nat Lab	Edison Cray XC30, Intel Xeon 12C 2.4GHz + Custom	132,840	1.65	18	.0439 #	2.7%
Texas Advanced Computing Center	Stampede, Dell Intel (8c) + Intel Xeon Phi (61c) + IB	78,848	.881*	7	.0161	1.8%
Meteo France	Beaufix Bullx B710 Intel Xeon 12C 2.7 GHz + IB	24,192	.469 (.467*)	79	.0110	2.4%
Meteo France	Prolix Bullx B710 Intel Xeon 2.7 GHz 12C + IB	23,760	.464 (.415*)	80	.00998	2.4%
U of Toulouse	CALMIP Bullx DLC Intel Xeon 10C 2.8 GHz + IB	12,240	.255	184	.00725	2.8%
Cambridge U	Wilkes, Intel Xeon 6C 2.6 GHz + Nvidia Kepler 14C + IB	3584	.240	201	.00385	1.6%
TiTech	TUSBAME-KFC Intel Xeon 6C 2.1 GHz + IB	2720	.150	436	.00370	2.5%

The Context for Synergistic Co-Design

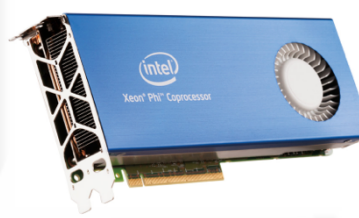
Massive parallelism w/ increasing heterogeneity in computing resources



Altera FPGA



AMD APU



Intel Xeon Phi (MIC)



NVIDIA GPU



TI DSP

... across a wide variety of environments



Tianhe-2



Performance via Parallelism

- 2,508 CPU cores
- 187,264 GPU cores
- ~ 6,000,000 threads

Debuted on **THE GREEN 500™**
as **GREENEST** commodity supercomputer in U.S.
(11/11)

Heterogeneous Systems in HPC

- Statistics

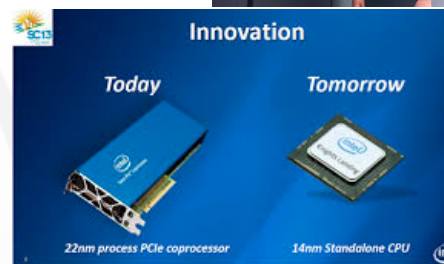
- Four of the top 10 systems
- Performance share in Top500 systems
5% (2009) → 35% (2014)

- HokieSpeed

- CPU+GPU heterogeneous supercomputer with large-scale visualization wall
- Debuted as the **GREENEST** commodity supercomputer in the U.S. in Nov. 2011



Tianhe-2



Top 10 Source: top500.org

- 1 **Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDI
- 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 **Cray XC30**, Intel Xeon E5-2697v2 12C 2.7GHz, Aries interconnect Cray Inc.

Heterogeneous Systems in HPC

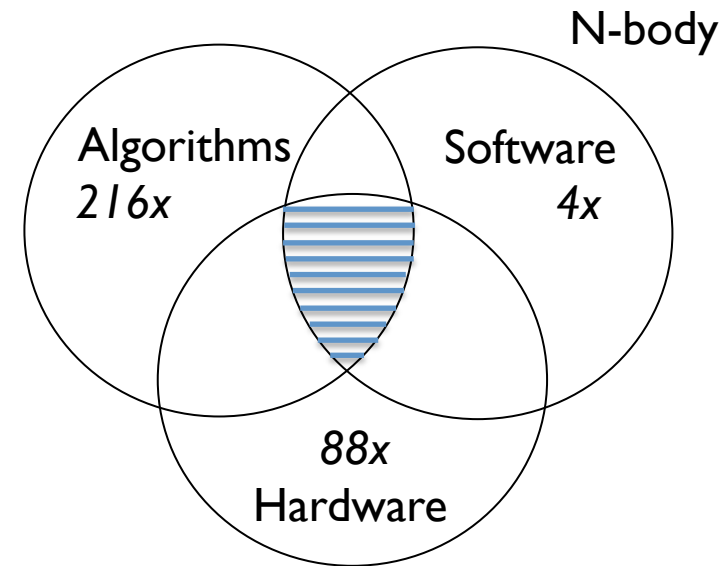


Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	4,503.17	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.100GHz, Infiniband FDR NVIDIA K20x	27.78
2	3,631.86	Cambridge University	Wilkes - Dell T620 Cluster, Intel Xeon E5-2630v2 6C 2.600GHz, Infiniband FDR NVIDIA K20	52.62
3	3,517.84	Center for Computational Sciences, University of Tsukuba	HA-PACS TCA - Cray 3623G4-SM Cluster, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband QDR NVIDIA K20x	78.77
4	3,185.91	Swiss National Supercomputing Centre (CSCS)	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect NVIDIA K20x Level 3 measurement data available	1,753.66
5	3,130.95	ROMEO HPC Center - Champagne-Ardenne	romeo - Bull R421-E3 Cluster, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR NVIDIA K20x	81.41
6	3,068.71	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.5 - Cluster Platform SL390s G7, Xeon X5670 6C 2.930GHz, Infiniband QDR NVIDIA K20x	922.54
7	2,702.16	University of Arizona	iDataPlex DX360M4, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR14 NVIDIA K20x	53.62
8	2,629.10	Max-Planck-Gesellschaft MPI/IPP	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband NVIDIA K20x	269.94
9	2,629.10	Financial Institution	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband NVIDIA K20x	55.62
10	2,358.69	CSIRO	CSIRO GPU Cluster - Nitro G16 3GPU, Xeon E5-2650 8C 2.000GHz, Infiniband FDR Nvidia K20m	71.01
11	2,351.10	King Abdulaziz City for Science and Technology	SANAM - Adtech, ASUS ESC4000/FDR G2, Xeon E5-2650 8C 2.000GHz, Infiniband FDR AMD FirePro S10000	179.15

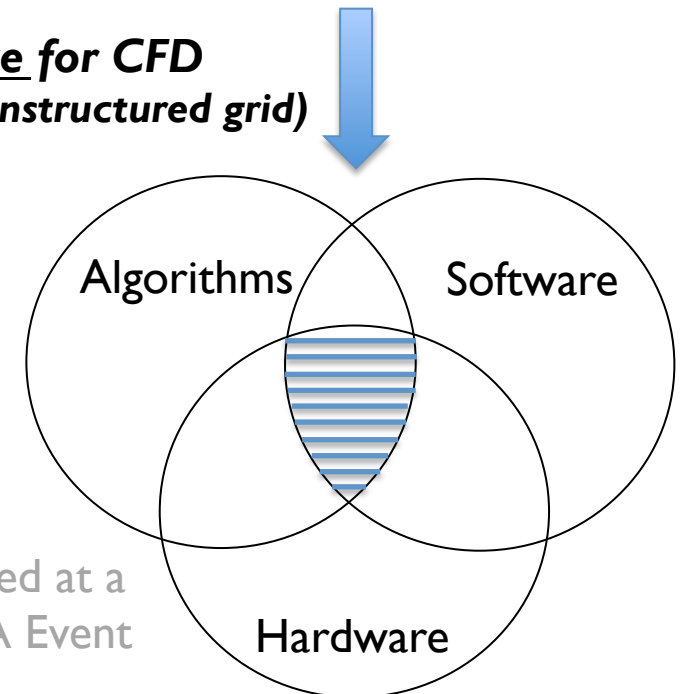
Roadmap

- Vision
- Motivation
- Approach: Synergistic Co-Design
- Artifacts
 - Optimized CFD Codes
 - CoreTSAR / AffinityTSAR
 - MetaMorph
 - MPI-ACC
 - VOCL:Virtual OpenCL
- Achievements & Publications
- Next Steps

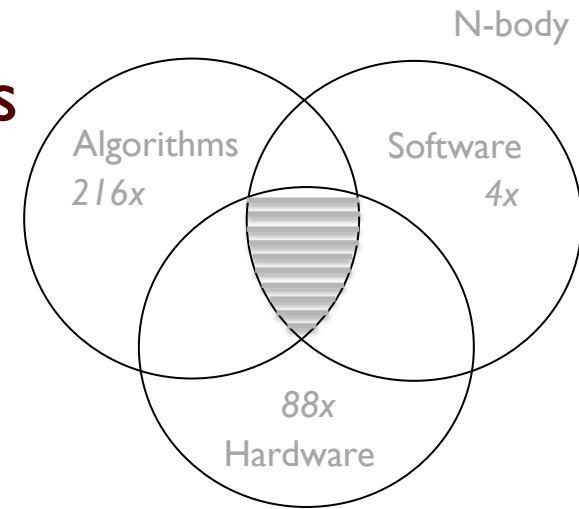
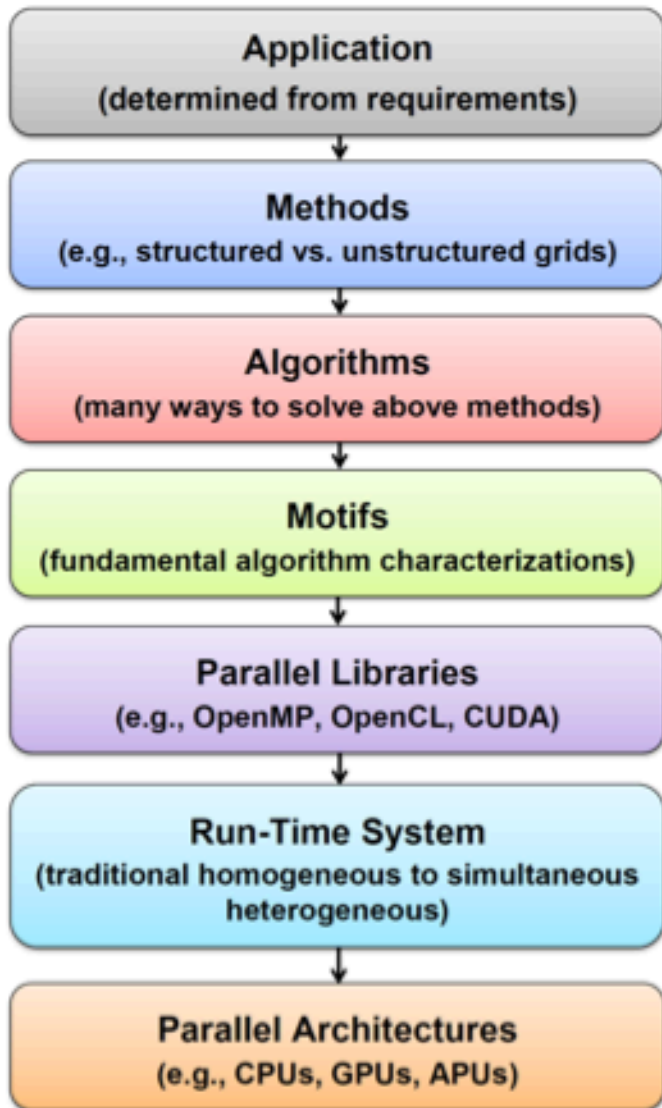
Generalize, as presented at a
White House BIGDATA Event
in May 2013



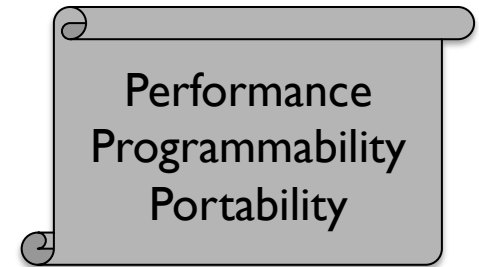
Formalize for CFD
(structured & unstructured grid)



Co-Design & Associated Research Areas



Performance



- Hardware always evolving.
- Application software should **NOT** have to change.
- *Co-designed hardware/software ecosystem should automatically adapt to new hardware*

Micro Air Vehicles (MAVs)

**Unsteady Aero
(A, B, C, D)**

Fluid/Structure Interaction
(Loosely Coupled)

**Structural
Dynamics (TBD)**

Artificial Compressibility (B, D) Projection Methods (A) Higher-Order Methods (C)

Structured Grid (A, B, D) Unstructured Grid (C)

Finite Difference (D) Finite Volume (A, B, D) Finite Element (C)

Explicit (A, B, C, D) Implicit (A, B, C, D)

Immersed Boundary Method (A, B) Arbitrary Lagrangian-Eulerian (A, B, C)

MPI SnucL

MPI OpenMP OpenACC OpenCL CUDA

Multicore CPU AMD GPU NVIDIA GPU AMD APU (CPU+GPU) Intel MIC

Incompressible Flow Method

Grid Type

Spatial Discretization

Temporal Discretization

Structural Motion

⋮

Internode Parallelism

Intranode Parallelism

Target Parallel Architectures

CFD Codes

A

GenIDLEST: Struct, Press Project, FVM

B

INCOMP3D: Struct, Art Compr, FVM

C

RDGFLO3D: Unstr, FEM


D

SENSEI: Struct, Art Comp, FVM

**Synergistic Co-Design
from 10,000 Feet:
Performance Perspective**

It's More than Co-Design for Performance ...

“Productivity = Performance + Programmability + Portability”

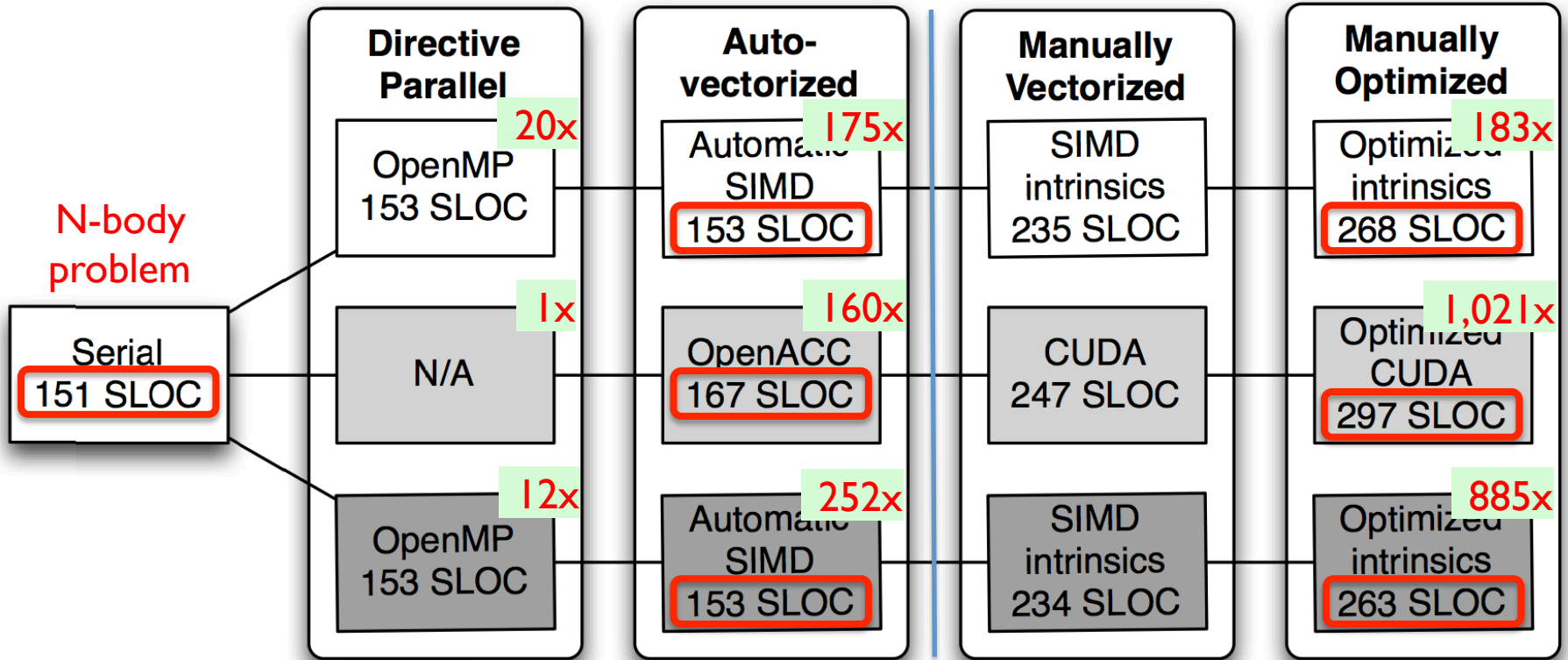
- Programmability and Portability? Who Cares?
 - Five years from now, you will *NOT* know what hardware will look like.
 - Re-write software if code is not portable. 
 - Case Study: Quantum Chemistry @ Stanford (~1,000,000 CUDA SLOC)*
 - What has Stanford been doing to get their code to run on other platforms?
- Performance in the Context of Programmability and Portability
 - Hardware/Software Co-Design (No Change in Algorithm)
 - Automated optimization (-O3) → advanced manual optimization → advanced automated optimization
 - Integrated Hardware/Software/Algorithm Co-Design
 - Manual co-design → automated co-design

* CU2CL: Automated CUDA-to-OpenCL Source-to-Source Translator could auto-translate the ~ 1,000,000 CUDA source lines of code (SLOC), which is part of our synergistic co-design ecosystem, but it only addresses functional portability for now. 

“Productivity = Performance + Programmability + Portability”

Least performance
Most programmable
Most portable

Most performance
Least programmable
Least portable



Legend:

Sandy Bridge CPU

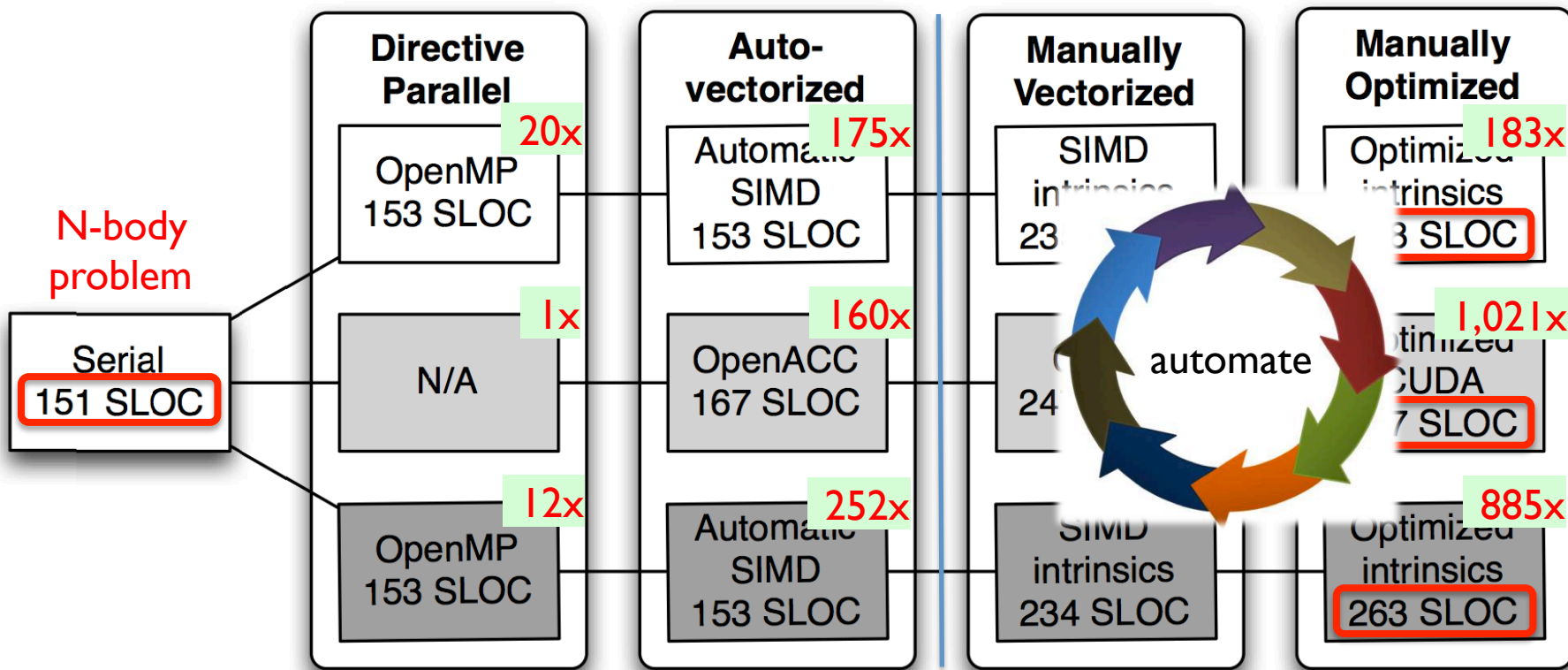
K20 GPU

Knight's Corner

“Productivity = Performance + Programmability + Portability”

Least performance
Most programmable
Most portable

Most performance
Least programmable
Least portable



Legend:

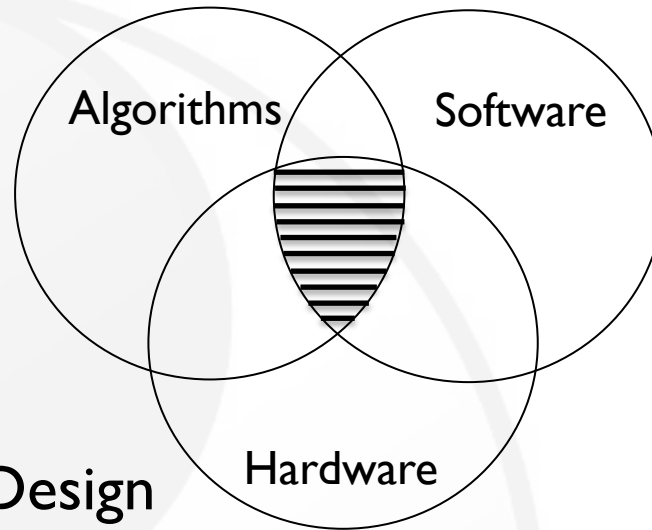
Sandy Bridge CPU

K20 GPU

Knight's Corner

Roadmap

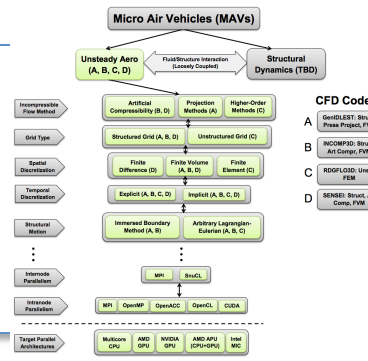
- Vision
- Motivation
- Approach:
Synergistic Co-Design
- Artifacts
 - Optimized CFD Codes
 - CoreTSAR / AffinityTSAR
 - MetaMorph
 - MPI-ACC
 - VOCL:Virtual OpenCL
- Achievements & Publications
- Next Steps



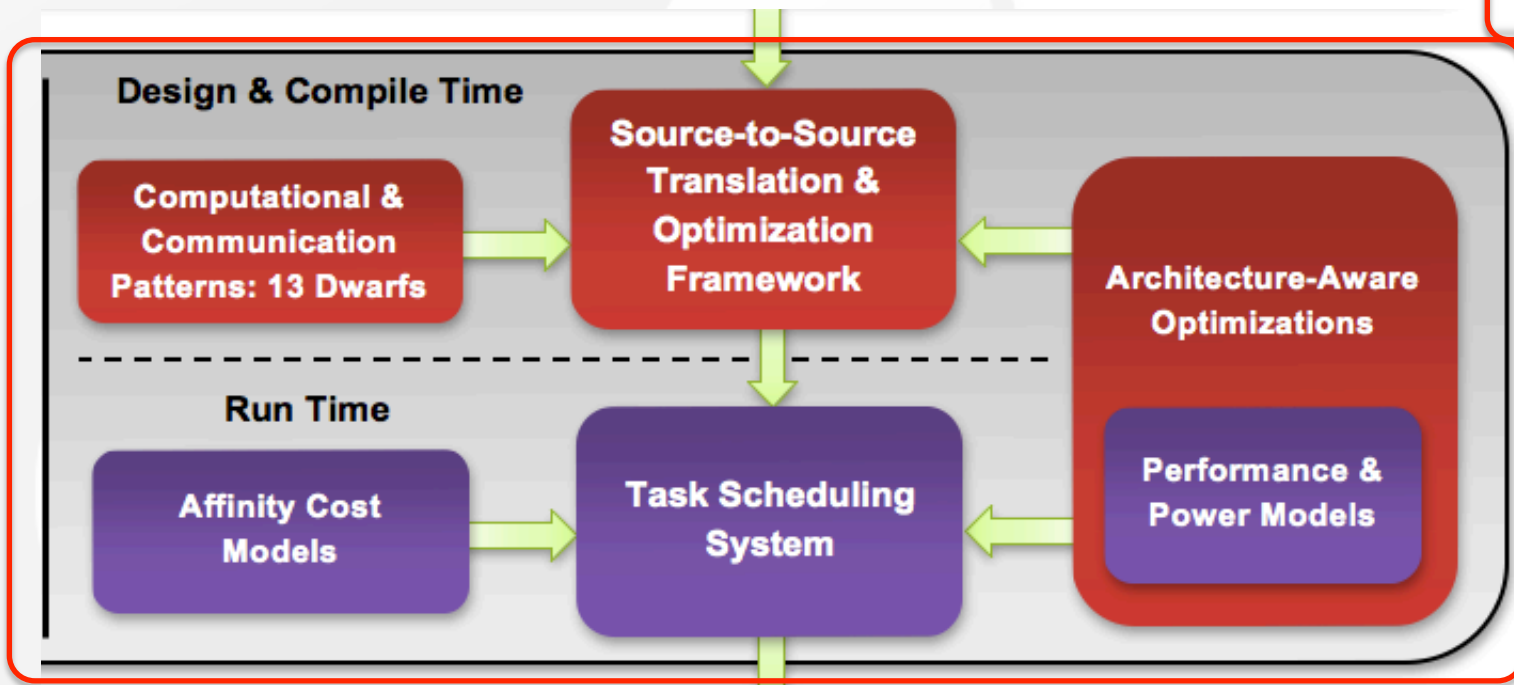
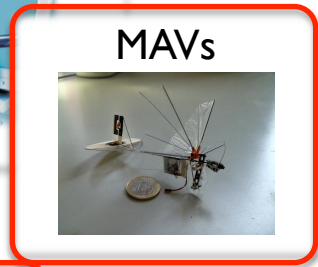
Ecosystem for Heterogeneous Parallel Computing

MANUAL CO-DESIGN

<p>Epidemiology</p>	<p>Sequence Alignment</p>	<p>Molecular Dynamics</p>	<p>Earthquake Modeling</p>
----------------------------	----------------------------------	----------------------------------	-----------------------------------



security



Manual Co-Design
→
Automated Co-Design

Heterogeneous Parallel Computing Platform

Targeted CFD Codes

SENSEI (C. Roy, Virginia Tech)

- Structured, multiblock, 2nd order, finite-volume code
- Artificial compressibility method
- 2nd-order spatial accuracy
- Artificial compressibility (AC)

GenIDLEST (D. Tafti, Virginia Tech)

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

RDGFLO (H. Luo, NCSU)

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

INCOMP3D (J. Edwards, NCSU)

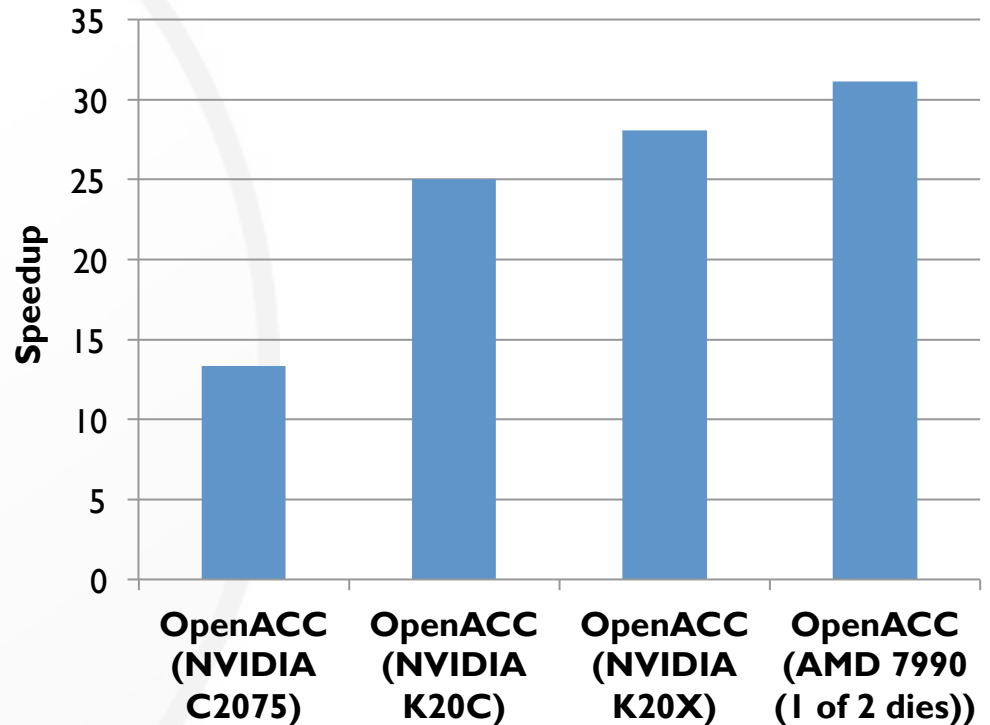
- Structured, multiblock finite-volume code
- Second or higher order spatial accuracy
- ALE and IBM

Testbed Codes

- Lid-Driven Cavity
- SENSEI Lite

Incompressible Navier-Stokes (INS) Finite-Difference Code

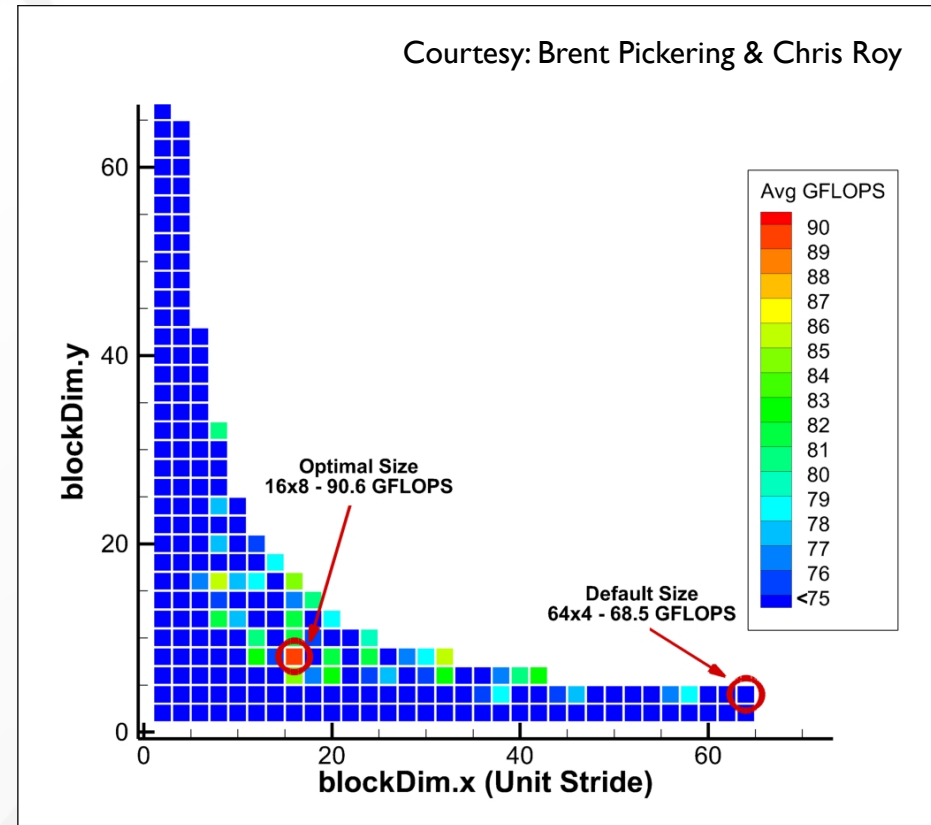
- Recap
 - 2D Cartesian grid FDM
 - Solves INS using artificial compressibility (lid-driven cavity benchmark case)
 - Ported from existing Fortran code to run on GPUs using OpenACC + PGI compiler
- Recent work focused on performance optimization of OpenACC code and running on multiple GPUs.
- Newer versions of PGI compiler (14.x) support more accelerator platforms, including AMD GPUs.



Speedup of INS code on several GPU platforms relative to a single CPU thread (SSE vectorized) running on a Xeon X5560.

Optimization of OpenACC using Gang/Worker/Vector Clauses

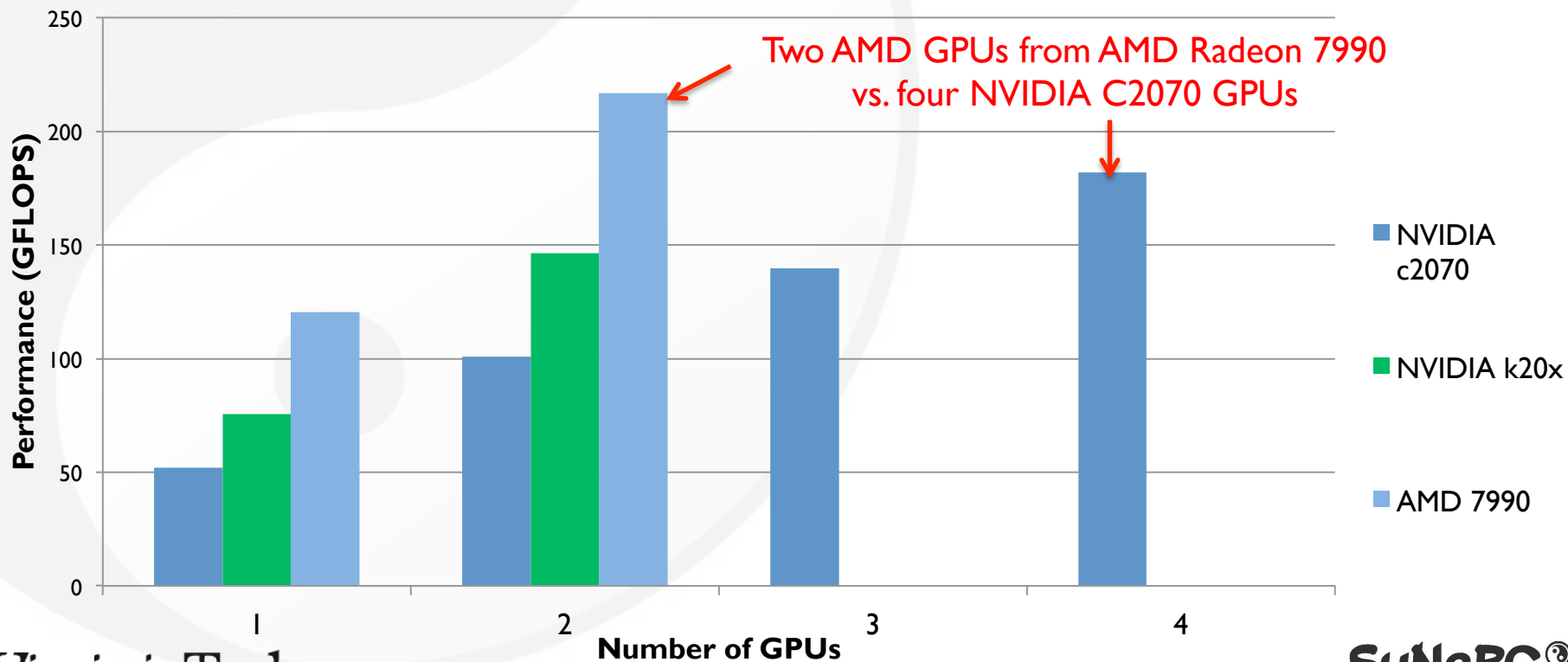
- Can use OpenACC clauses to control the kernel launch configuration on NVIDIA devices.
- Explored entire parameter space of possible 2D thread-block dimensions.
 - Tested both Fermi (C2075) and Kepler (K20) GPUs, using double- and single-precision arithmetic.
 - On all platforms, the default block size (when no vector clause was used) was observed to be 64x4.
 - Manual tuning showed performance increases ranging from 6-33% on the different GPUs. The compiler default was never found to be optimal.
- *Applying approaches to eliminate “brute force” search → Starchart / Stargazer*



Optimization results for K20c GPU using double precision.
Default: 64x4 threads/block → 68.5 GFLOPS
Optimal: 16x8 threads/block → 90.6 GFLOPS

Multi-GPU Scaling

- Near-linear performance scaling using multiple GPUs.
 - Domain decomposition, with each domain partition residing on one GPU for duration of simulation (only ghost cells had to be exchanged on each iteration).
 - One control CPU thread per GPU.
 - PGI 14.1 compiler can generate code for AMD GPUs in addition to NVIDIA.



Targeted CFD Codes

SENSEI (C. Roy, Virginia Tech)

- Structured, multiblock, 2nd order, finite-volume code
- Artificial compressibility method
- 2nd-order spatial accuracy
- Artificial compressibility (AC)

GenIDLEST (D. Tafti, Virginia Tech)

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

RDGFLO (H. Luo, NCSU)

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

INCOMP3D (J. Edwards, NCSU)

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

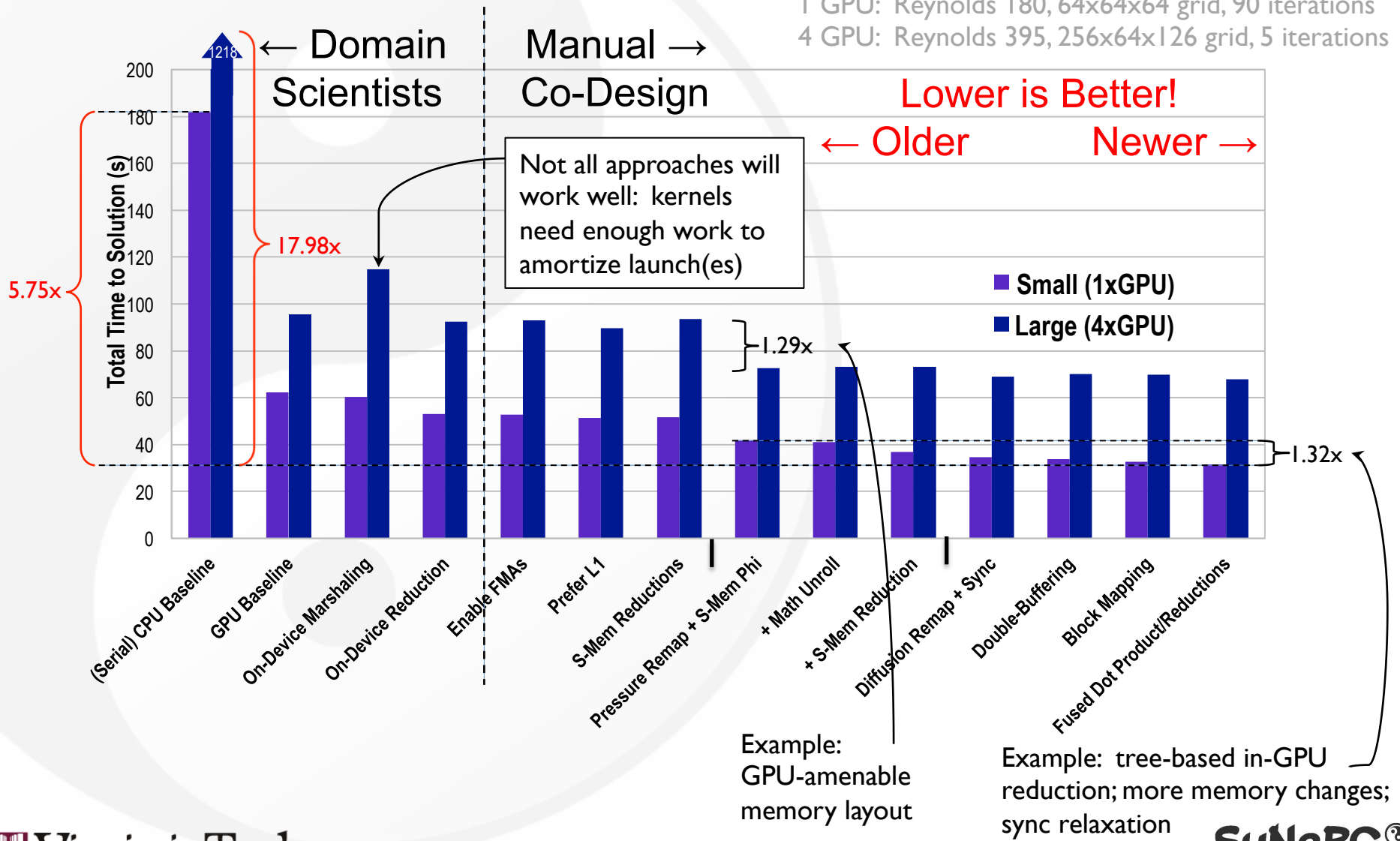
Testbed Codes

- Lid-Driven Cavity
- SENSEI Lite

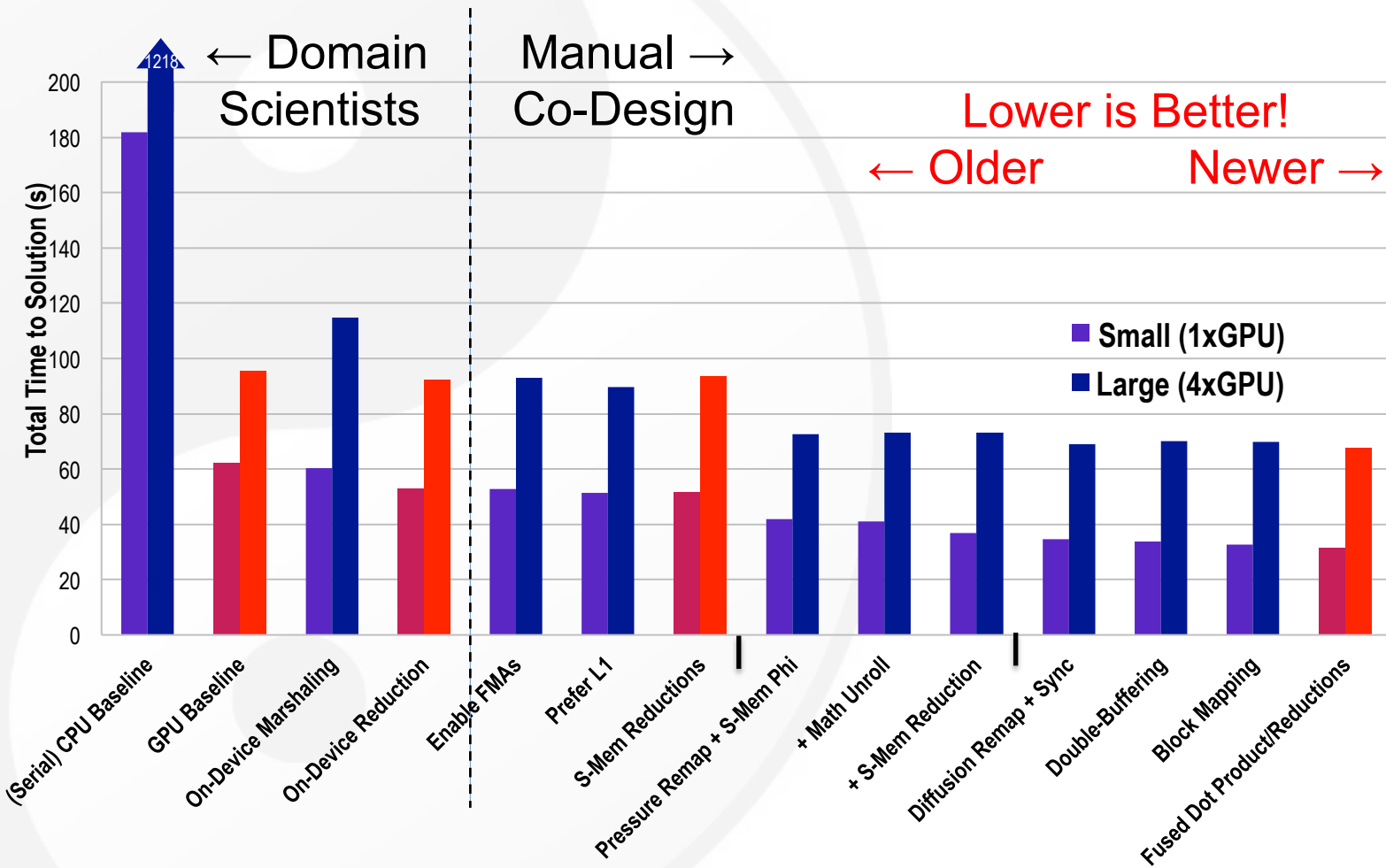
Status of these codes
relative to co-design?

GenIDLEST: Manual GPU Optimization

1 GPU: Reynolds 180, 64x64x64 grid, 90 iterations
 4 GPU: Reynolds 395, 256x64x126 grid, 5 iterations



GenIDLEST: GPU Dot-Product Modifications



Dot Product: Serial on CPU

HOST

Serial C

```
1. void dotProd(double *a,  
2.             double *b,  
3.             double *ret,  
4.             int n_elem)  
5. {  
6.     int i;  
7.     *ret = 0.0;  
8.     for (i = 0; i < n_elem; i++)  
9.         ret += a[i] * b[i];  
10. }
```

10

DEVICE

Dot Product: Parallel on GPU Device using CUDA

HOST

```

1. int main(int argc, char ** argv)
2. {
3.     int n_elem;
4.     double *a, *b, *ret, sum = 0.0;
5.     //allocate and initialize a, b, and ret
6.     ...
7.     double *dev_a, *dev_b, *dev_r;
8.     size_t size = n_elem * sizeof(double);
9.     //allocate device buffers
10.    cudaMalloc((void**)&dev_a, size);
11.    cudaMalloc((void**)&dev_b, size);
12.    cudaMalloc((void**)&dev_r, size);

13.    //initialize device buffers
14.    cudaMemcpy(dev_a, a, size, cudaMemcpyHostToDevice);
15.    cudaMemcpy(dev_b, b, size, cudaMemcpyHostToDevice);

16.    //set grid/block size
17.    ...
18.    //multiply elements
19.    dotProd-vmul<<<grid, block>>>(dev_a, dev_b, dev_r,
20.        n_elem);

21.    //partial result
22.    cudaMemcpy(ret, dev_r, size, cudaMemcpyDeviceToHost);

23.    //CPU sum
24.    int i;
25.    for (i = 0; i < n_elem; i++)
26.        sum += ret[i];
27. }
```

27

DEVICE (GPU)

```

1. __global void dotProd-vmul(
2.     double *a, double *b,
3.     double *ret, int n_elem)
4. {
5.     int i;
6.     int tid = blockIdx.x *
7.         blockDim.x + threadIdx.x;
8.     int n_threads = blockDim.x *
9.         gridDim.x;
10.    for (i=tid; i<n_elem; i+=n_threads)
11.        ret[tid] = a[tid] * b[tid];
12. }
```

12

Dot Product: Optimized Parallel on GPU Device

```

1. int main(int argc, char **argv)
2. {
3.     //global and block size
4.     int ni, nj, nk, tx, ty, tz;
5.     //pick a mode and zeroth device
6.     cudaSetDevice(0)
7.     //declare host memory
8.     double *a, *b, *ret, zero=0.0;
9.     //allocate and initialize it
10.    ...

11.    //Allocate device buffers
12.    size_t size = sizeof(double)*ni*nj*nk;
13.    double *dev_a, *dev_b, *dev_r;
14.    cudaMalloc(&dev_a, size);
15.    cudaMalloc(&dev_b, size);
16.    cudaMalloc(&dev_r, sizeof(double));
17.    //initialize them
18.    cudaMemcpy(dev_a, a, size,
19.        cudaMemcpyHostToDevice);
20.    cudaMemcpy(dev_b, a, size,
21.        cudaMemcpyHostToDevice);
22.    cudaMemcpy(dev_r, &zero, sizeof(double),
23.        cudaMemcpyHostToDevice);

24.    //set computation shape
25.    dim3 grid, block, shape, start, end;
26.    block = dim3(tx, ty, tz);
27.    grid = dim3((ni+tx-1)/ni, (nj+ty-1)/ty, 1);
28.    shape = dim3(ni, nj, nk);
29.    start = dim3(0, 0, 0);
30.    end = dim3(ni-1, nj-1, nk-1);

31.    //run the kernel
32.    dotProd<<<grid, block, tx*ty*tz*sizeof(double)>>>
33.        (dev_a, dev_b, shape.x, shape.y, shape.z,
34.         start.x, start.y, start.z, end.x, end.y, end.z,
35.         (nk+tz-1)/tz, dev_r, tx*ty*tz);

36.    //bring the dot product back
37.    cudaMemcpy (ret, dev_r, sizeof(double),
38.        cudaMemcpyDeviceToHost);
39. }
    
```

39

```

1. __device__ void block_reduction(double *psum,
2.     int tid, int len_)
3. {
4.     int stride = len_ >> 1;
5.     while (stride > 0) {
6.         if (tid < stride)
7.             psum[tid] += psum[tid+stride];
8.         __syncthreads();
9.         stride >>= 1;
10.    }
11. }

12. //Implementation of double atomicAdd
13. ...

14. __global__ void kernel_dotProd(double *phi1, double
15.     *phi2, int i, int j, int k, int sx,
16.     int sy, int sz, int ex, int ey, int ez,
17.     int gz, T * reduction, int len_)
18. {
19.     extern shared psum[];
20.     int tid, x, y, z, itr;
21.     bool boundx, boundy, boundz;
22.     tid = threadIdx.x + (threadIdx.y * blockDim.x
23.         + (threadIdx.z * (blockDim.x * blockDim.y));
24.     x = (blockIdx.x)*blockDim.x+threadIdx.x+sx;
25.     y = (blockIdx.y)*blockDim.y+threadIdx.y+sy;

26.     psum[tid] = 0;
27.     boundy = ((y >= sy) && (y <= ey));
28.     boundx = ((x >= sx) && (x <= ex));

29.     for (itr = 0; itr < gz; itr++) {
30.         z = itr*blockDim.z+threadIdx.z +sz;
31.         boundz = ((z >= sz) && (z <= ez));
32.         if (boundx && boundy && boundz)
33.             psum[tid] += phi1[x+y*i+z*i*j] *
34.                 phi2[x+y*i+z*i*j];
35.     }
36.     __syncthreads();
37.     block_reduction(psum, tid, len_);
38.     __syncthreads();

39.     if(tid == 0)
40.         atomicAdd(reduction, psum[0]);
    
```

40



Dot Product via our MetaMoph Library

```

1. int main(int argc, char **argv)
2. {
3.     //global and block sizes
4.     int ni, nj, nk, tx, ty, tz;
5.     //pick a mode and zeroth device
6.     choose_accel(0, accelModePreferCUDA);
7.     //declare host memory
8.     double *a, *b, *ret, zero = 0.0;
9.     //allocate and initialize it
10.    ...
11.    //Allocate device buffers
12.    size_t size = sizeof(double)*ni*nj*nk;
13.    double *dev_a, *dev_b, *dev_r;
14.    accel_alloc(&dev_a, size);
15.    accel_alloc(&dev_b, size);
16.    accel_alloc(&dev_r, sizeof(double));
17.    //initialize them
18.    accel_copy_h2d(dev_a, a, size, true);
19.    accel_copy_h2d(dev_b, b, size, true);
20.    accel_copy_h2d(dev_r, &zero, sizeof(double),
21.        true);
22.    //set computation shape
23.    a_dim3 grid, block, shape, start, end;
24.    block[0] = tx, block[1] = ty, block[2] = tz;
25.    grid[0] = (ni+tx-1)/ni, grid[1] = (nj+ty-1)/ty,
26.        grid[2] = (nk+tz-1)/tz;
27.    shape[0] = ni, shape[1] = nj, shape[2] = nk;
28.    start[0] = 0, start[1] = 0, start[2] = 0;
29.    end[0] = ni-1, end[1] = nj-1, end[2] = nk-1;
30.    //run the kernel
31.    accel_dotProd(&grid, &block, dev_a, dev_b,
32.        &shape, &start, &end, dev_r, a_db, true);
33.    //bring the dot product back
34.    accel_copy_d2h(ret, dev_r, sizeof(double),
35.        false);
36.}
    
```

36

also support:

accelModePreferOpenCL
for AMD/MIC/CPU

all kernels/copies can be
asynchronous with
flag = true, else blocking

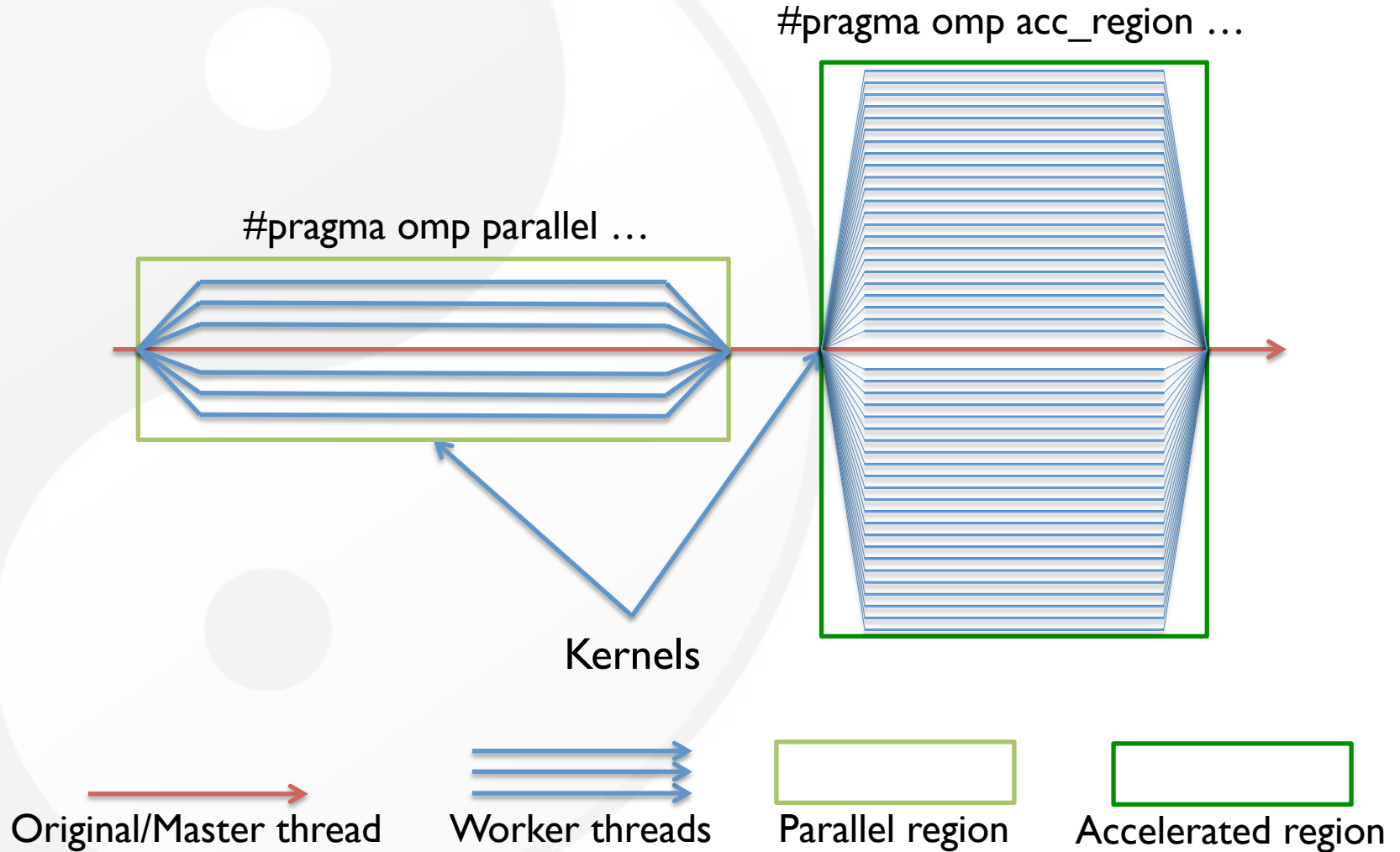
grid & block specify
thread organization a
la CUDA/OpenCL

shape, start, and end allow dot
product on arbitrary
subregions of 3D space

Automated Task Scheduling (with CoreTSAR)

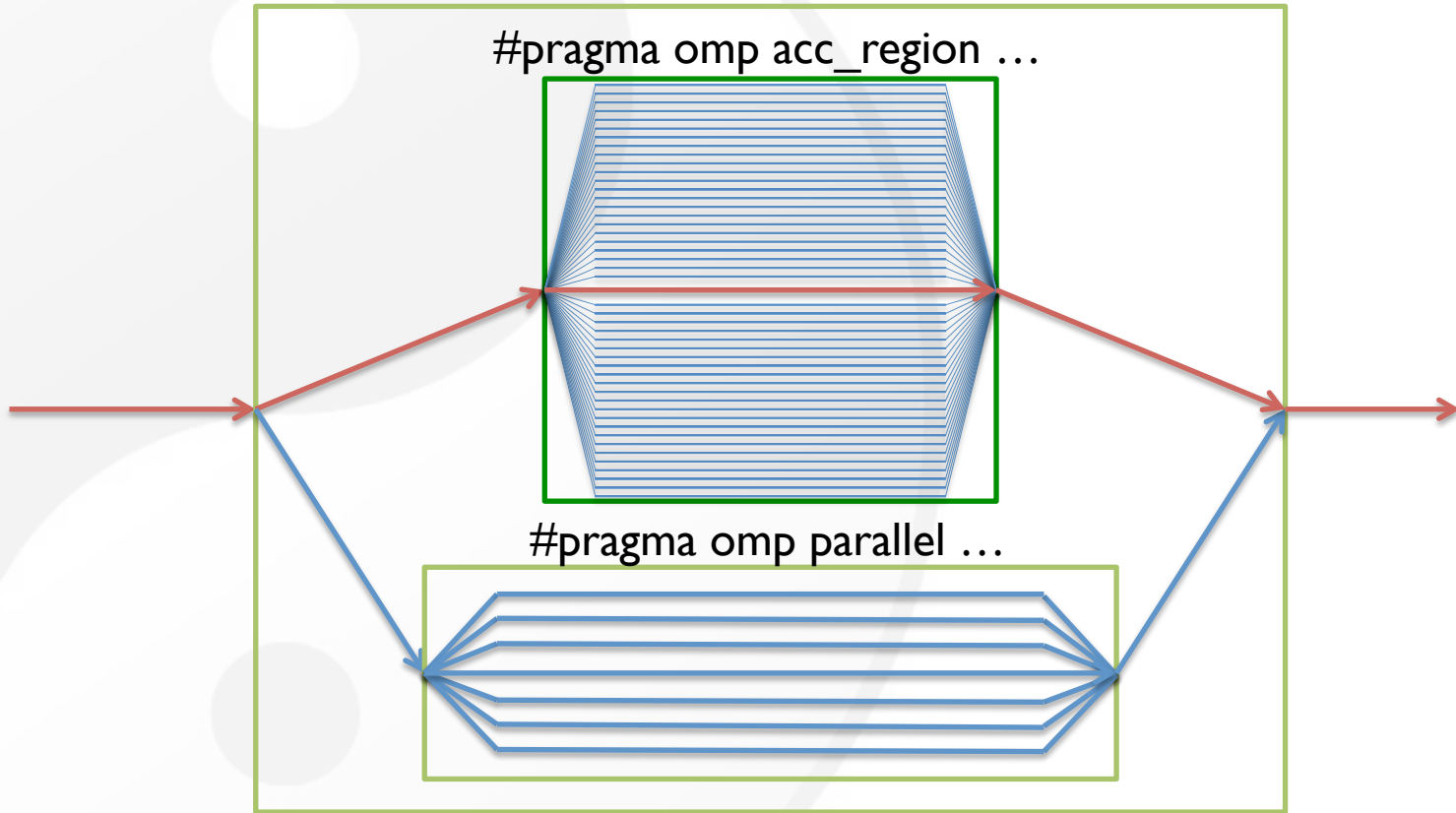
- Automatically dividing parallel tasks across arbitrary heterogeneous compute resources *simultaneously* for *functional portability*
 - CPUs, GPUs, APUs, Co-processors, FPGAs, ...
- Intelligent runtime task scheduling for *performance portability*
 - Accelerators add *physical heterogeneity and distributed memory*
 - GPUs, FPGAs, Co-processors (Intel MIC, Tileria Tile64, etc.)
 - System topology adds *heterogeneity through locality imbalance*
 - Hierarchical partially-shared caches
 - Non-uniform memory access (NUMA) memory systems
 - Operating system imbalance, work unevenly distributed to cores
 - Non-uniform latency to peripheral devices

OpenMP Accelerator Behavior



Our Version → OpenACC

```
#pragma omp parallel num_threads(2)
```



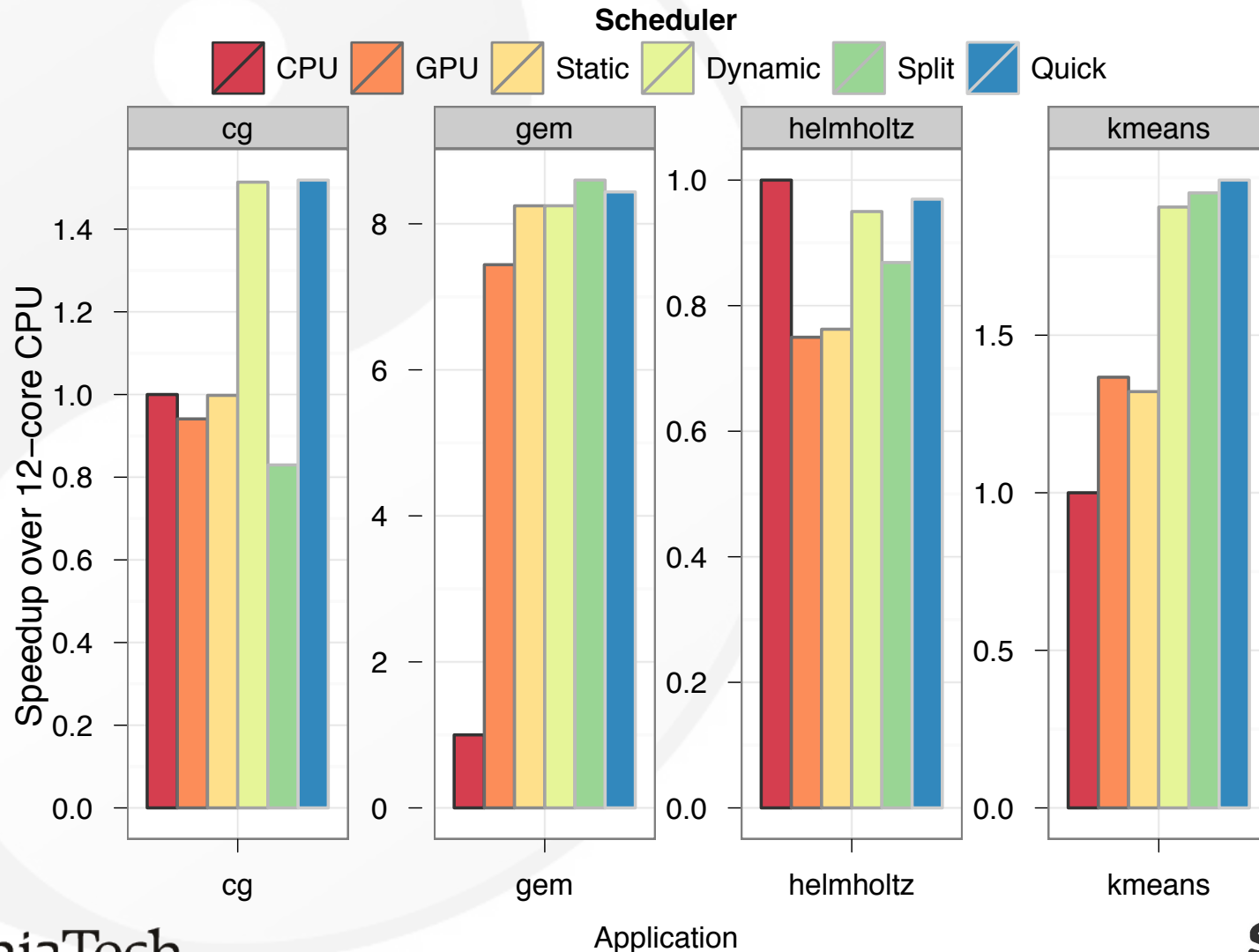
Original/Master thread

Worker threads

Parallel region

Accelerated region

Our Automated Task Schedulers via Co-Design vs. 12-Core CPU



Dot Product with OpenACC

Serial C

```
1. void dotProd(double *a,
2.             double *b,
3.             double *ret,
4.             int n_elem)
5. {
6.     int i;
7.     *ret = 0.0;
8.     for (i = 0; i < n_elem; i++)
9.         ret += a[i] * b[i];
10.}
```

OpenACC

```
1. void dotProd(double *a,
2.             double *b,
3.             double *ret,
4.             int n_elem)
5. {
6.     int i;
7.     double summer = 0;
8.     #pragma acc kernels for independent \
9.         copyin(a[n_elem],b[n_elem]) \
10.        reduction(+:summer)
11.     for (i = 0; i < n_elem; i++)
12.         summer += a[i] * b[i];
13.     *ret = summer;
14.}
```

14

Just **one** pragma, target one of GPU/CPU/MIC/...

Dot Product Code Example

Serial C

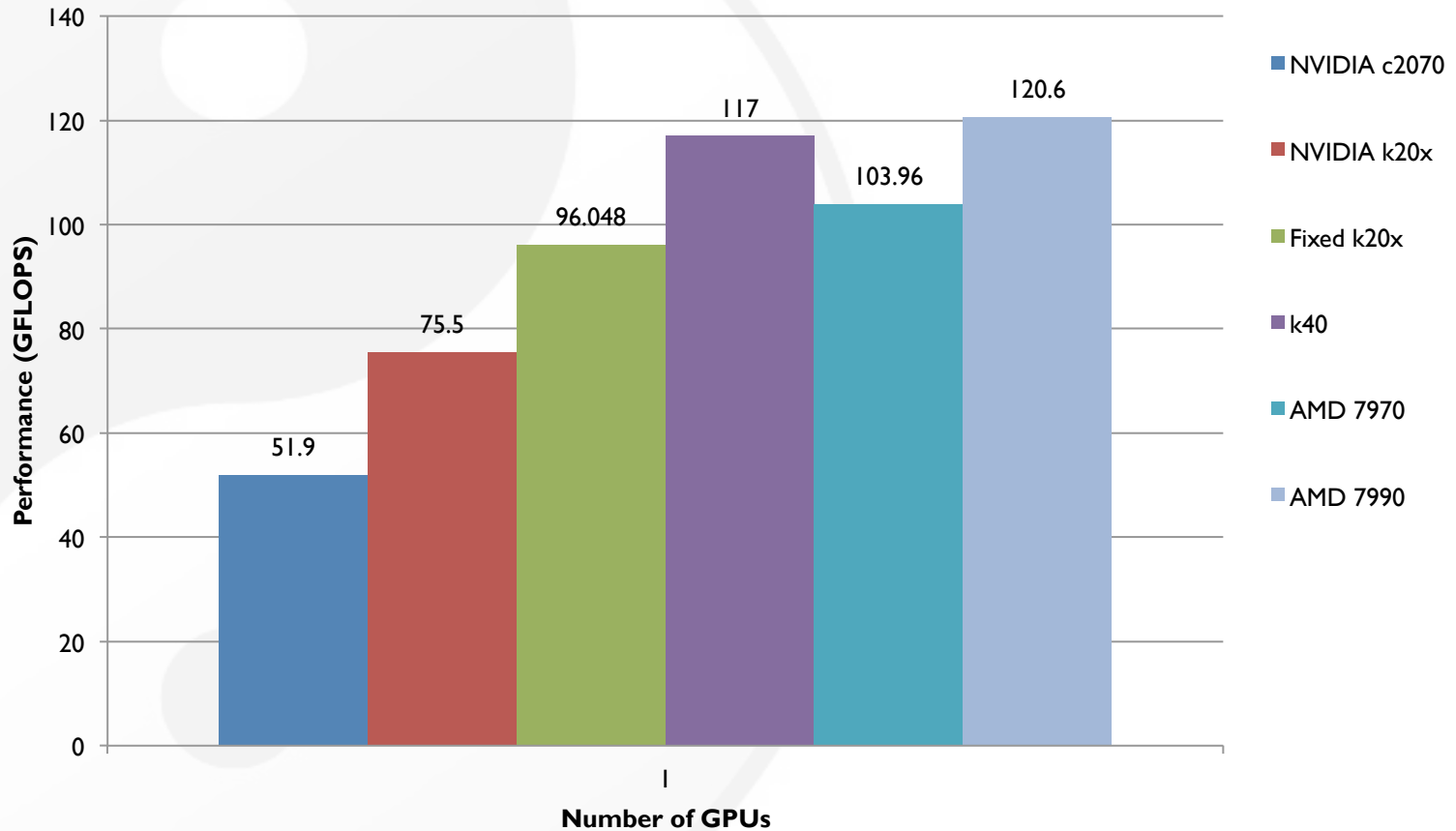
```
1. void dotProd(double *a,  
2.             double *b,  
3.             double *ret,  
4.             int n_elem)  
5. {  
6.     int i;  
7.     *ret = 0.0;  
8.     for (i = 0; i < n_elem; i++)  
9.         ret += a[i] * b[i];  
10. }
```

OpenACC

```
1. void dotProd(double *a,  
2.             double *b,  
3.             double *ret,  
4.             int n_elem)  
5. {  
6.     int i;  
7.     double summer = 0;  
8.     #pragma acc kernels for independent \  
9.         copyin(a[n_elem],b[n_elem]) \  
10.        reduction(+:summer)  
11.     for (i = 0; i < n_elem; i++)  
12.         summer += a[i] * b[i];  
13.     *ret = summer;  
14. }
```

Just **one** pragma, target one
of GPU/CPU/MIC/...

Evaluating OpenACC with LDC



CoreTSAR Extended OpenACC

CoreTSAR

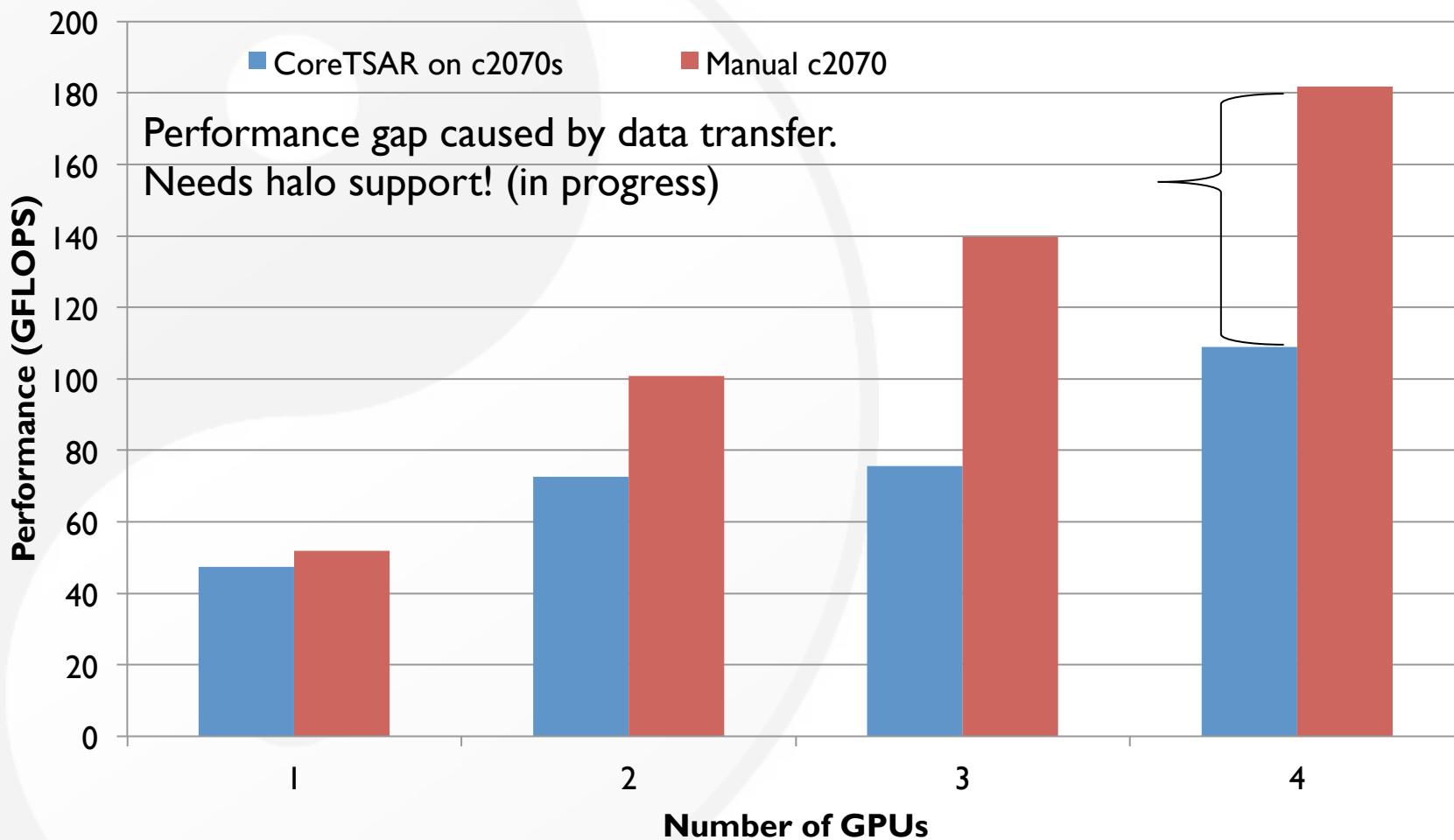
```
1. void dotProd(double *a,  
2.             double *b,  
3.             double *ret,  
4.             int n_elem)  
5. {  
6.     int i;  
7.     double summer = 0;  
8.     #pragma acc kernels for independent hetero(true) \  
9.         copyin(a[true:n_elem],b[true:n_elem]) \  
10.        reduction(+:summer)  
11.     for (i = 0; i < n_elem; i++)  
12.         summer += a[i] * b[i];  
13.     *ret = summer;  
14. }
```

OpenACC

```
1. void dotProd(double *a,  
2.             double *b,  
3.             double *ret,  
4.             int n_elem)  
5. {  
6.     int i;  
7.     double summer = 0;  
8.     #pragma acc kernels for independent \  
9.         copyin(a[n_elem],b[n_elem]) \  
10.        reduction(+:summer)  
11.     for (i = 0; i < n_elem; i++)  
12.         summer += a[i] * b[i];  
13.     *ret = summer;  
14. }
```

Three small changes, target
all devices with
coscheduling!

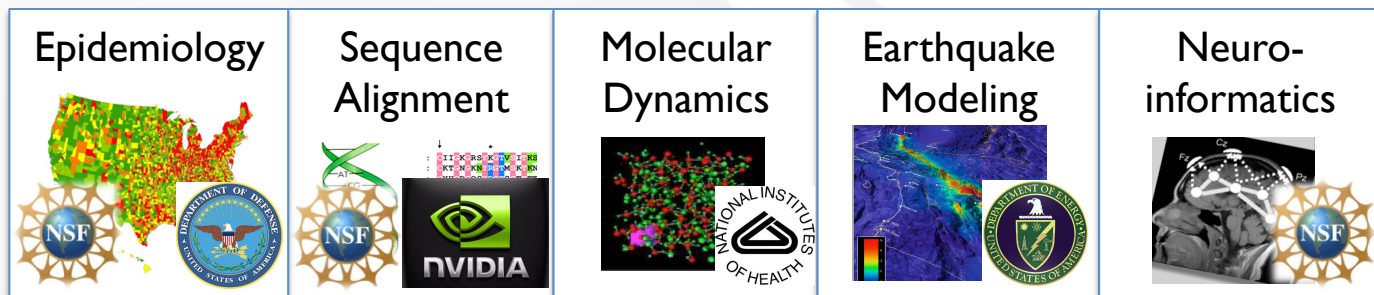
Multi-GPU LDC with CoreTSAR



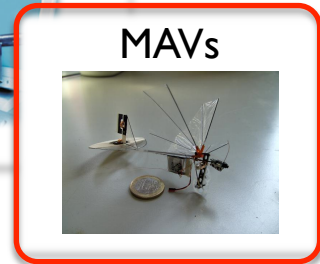
Intra-Node

Ecosystem for Heterogeneous Parallel Computing

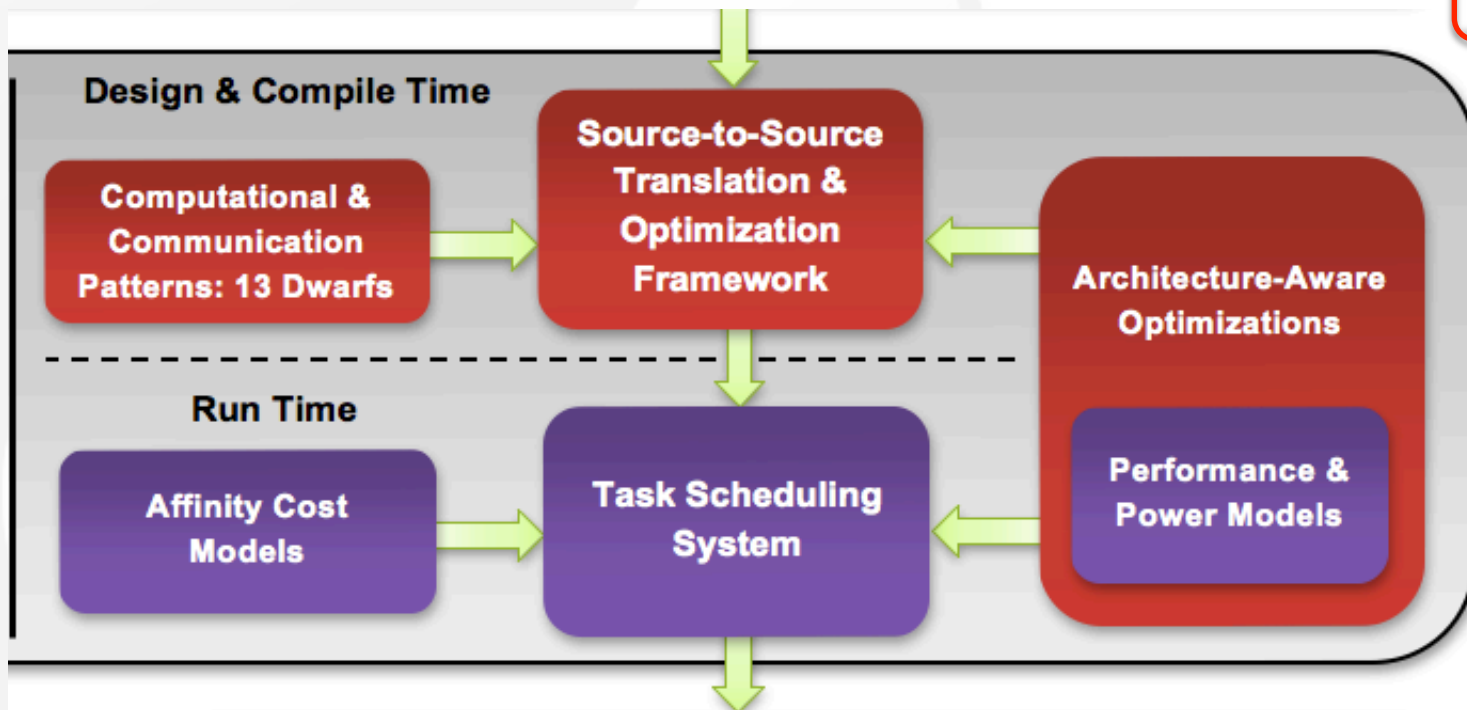
MANUAL CO-DESIGN



Cybersecurity



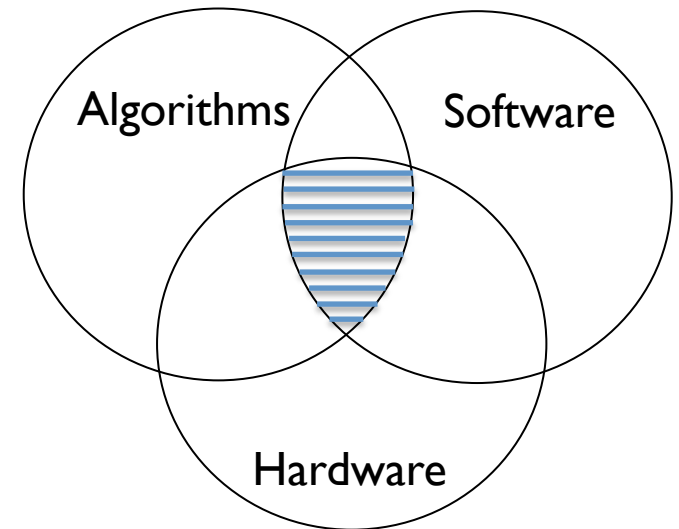
Manual Co-Design
→
Automated Co-Design



Heterogeneous Parallel Computing Platform

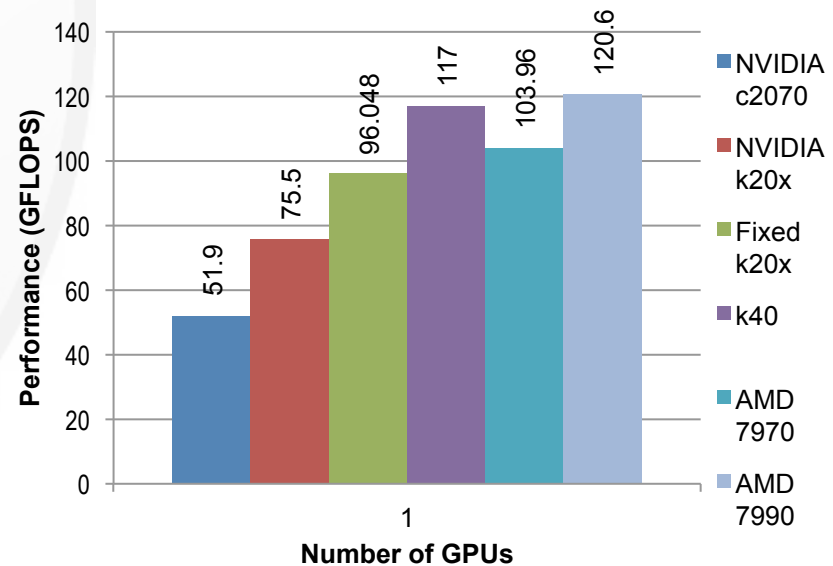
Roadmap

- Vision
- Team
- Approach: Synergistic Co-Design
- **Artifacts**
 - **Optimized CFD Codes**
 - **CoreTSAR / AffinityTSAR**
 - **MetaMorph**
 - MPI-ACC
 - VOCL:Virtual OpenCL
- **Achievements & Publications**
- **Next Steps**



Vision: Abstraction Library → *MetaMorph*

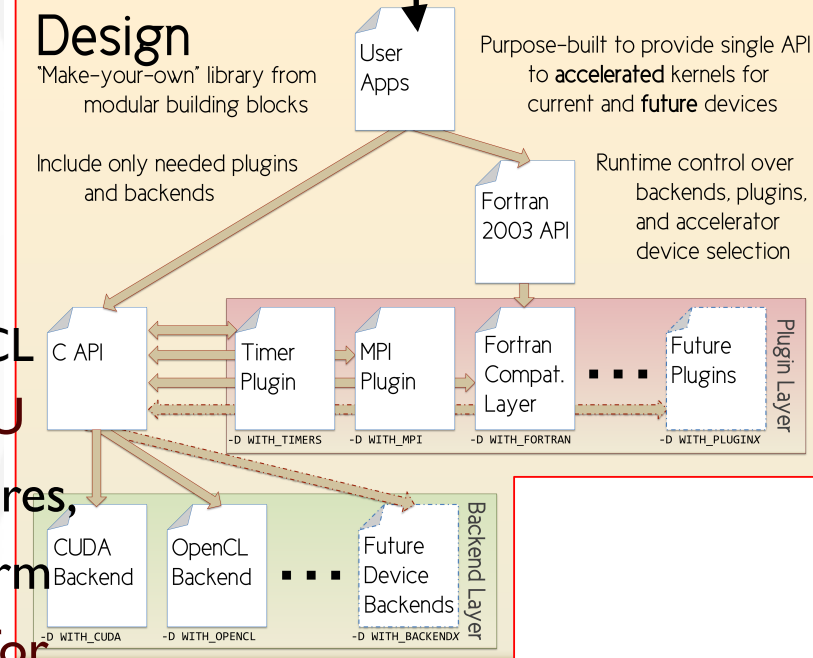
- Motivation
 - Architectures changing rapidly → scientists should *not* have to rewrite codes!
- Goal
 - *Community-driven* development of accelerated back-ends and plugins
 - Support for distributing workload onto the right device(s) with minimal user intervention, i.e.
 - Automatic load balancing at runtime to deliver better performance and resource utilization (via CoreTSAR)
- Desired Features
 - Usable by non-architecture experts (via drop-in replacement functions)
 - How? Hide accelerator-centric languages & optimizations behind a standard interface



MetaMorph: Updates

- **Poster session at SC'14**

- *MetaMorph: A Modular Library for Democratizing the Acceleration of Parallel Computing across Heterogeneous Devices*
- Strong interest in an HSA-native backend
 - AMD et. al
- Moderate interest in an OpenMP backend
 - target Intel MIC and CPU natively
 - Intel might be considering dropping OpenCL
- Several papers on implementing SpMV GPU operations on CSR and similar sparse structures, efficiently and without expanding to dense form
- Paper on communication-avoiding Krylov for CPU/GPU hybrid clusters



MetaMorph: Updates

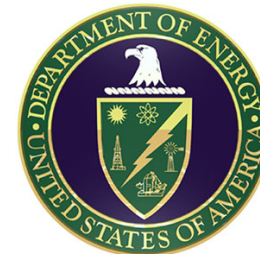
- **Data Marshalling & Face Transform Survey**
 - Three replies: GenIDLEST, INCOMP3D, and SENSEI
 - Needed operations
 - *Transpose, Rotate (90/180/270), Mirror;*
 - GenIDLEST also needs *Interpolate*.
 - **SENSEI needs Z-ordering inversal of XY planes**
- **Potential use in a Smooth Particle Hydrodynamics (SPH) code**
 - Essentially a particle-in-cell (PIC) code w/ sorted array of particles indexed by cell
 - Needs MPI capability to transfer “extents” of contiguous memory containing particles for a given cell

MetaMorph: Future Work

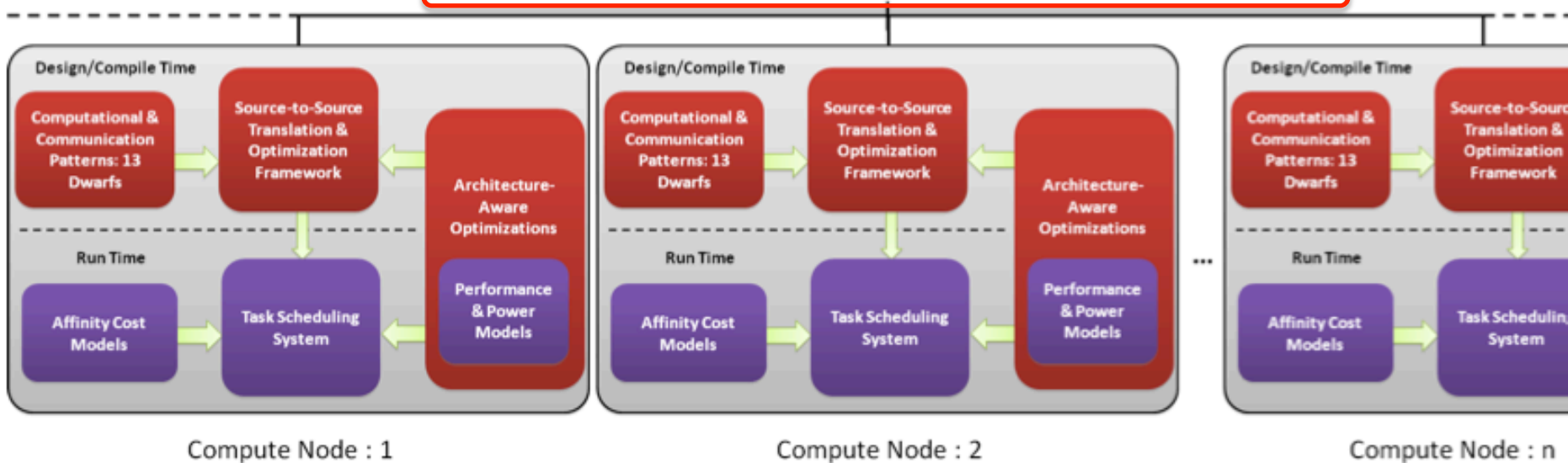
- Incorporation of remaining face transforms
 - Rotations, mirroring, and scaling
- Validation of ASYNC variants of face transforms and exchanges
 - MPI exchanges should be the most tricky, due to callback chains and helper functions required for transparent management of async transfers
- Develop plane ordering inversal primitive for SENSEI
- Identify and Isolate primitives needed by computational math
 - MatVec operations, Multigrids, other computational patterns
- Expand MPI plugin to transfer “extents” and sets of extents
 - to support SPH and other Particle-in-cell style computations
- Investigate ParMETIS interoperability for potential plugin

Ecosystem for Heterogeneous Parallel Computing

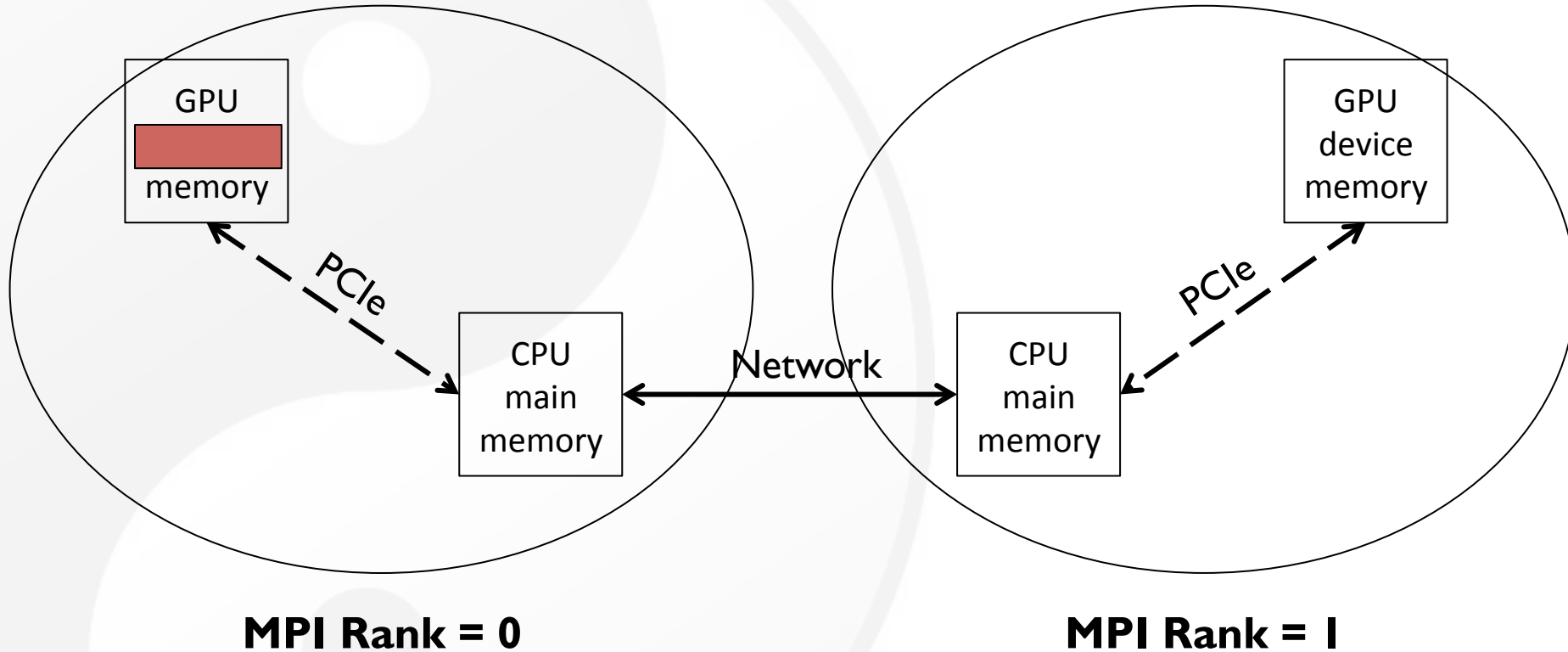
- Goal
 - Co-designed data movement library that hides all the hardware and system software details from the algorithm developer while supporting a multitude of environments



MPI-ACC: Data Communication Library across Heterogeneous Compute Systems



Data Movement in CPU-GPU Clusters



MPI Rank = 0

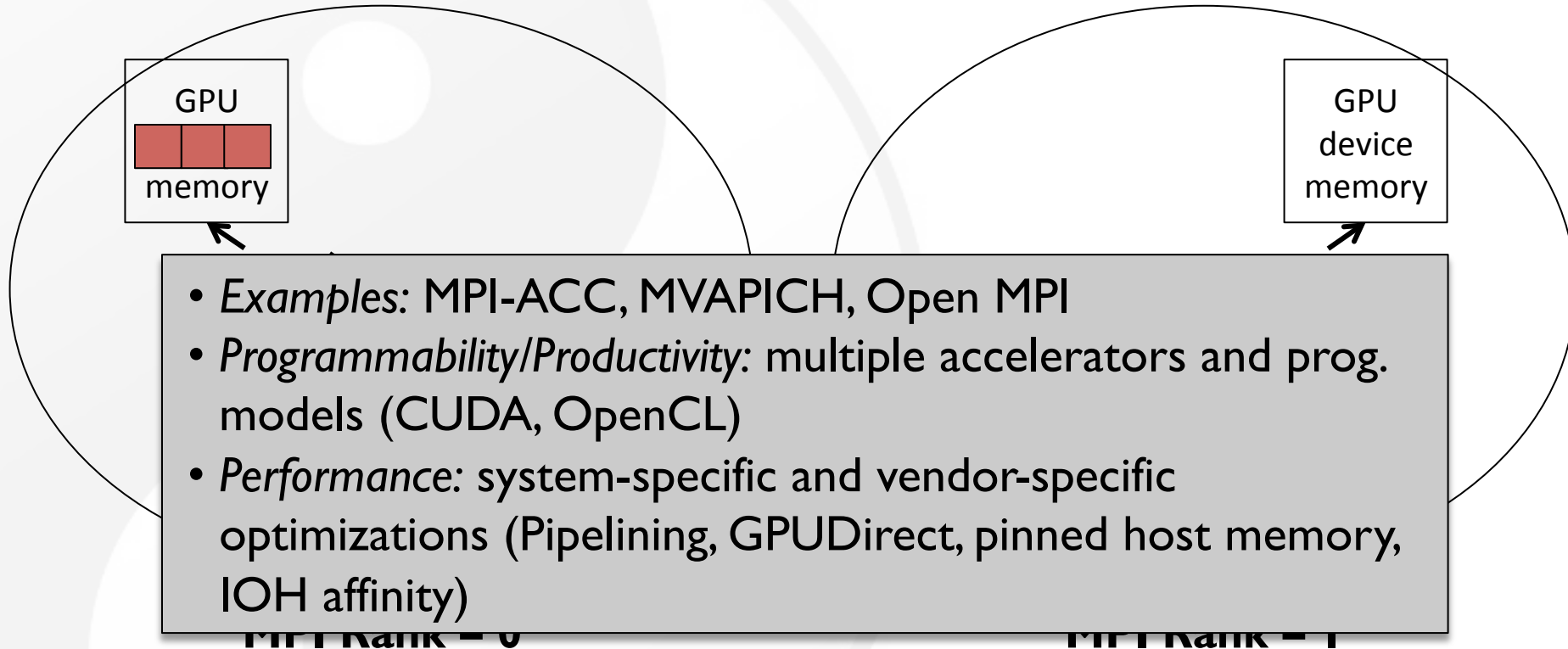
MPI Rank = 1

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

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


MPI-ACC:

Generalized Runtime for Accelerator Systems

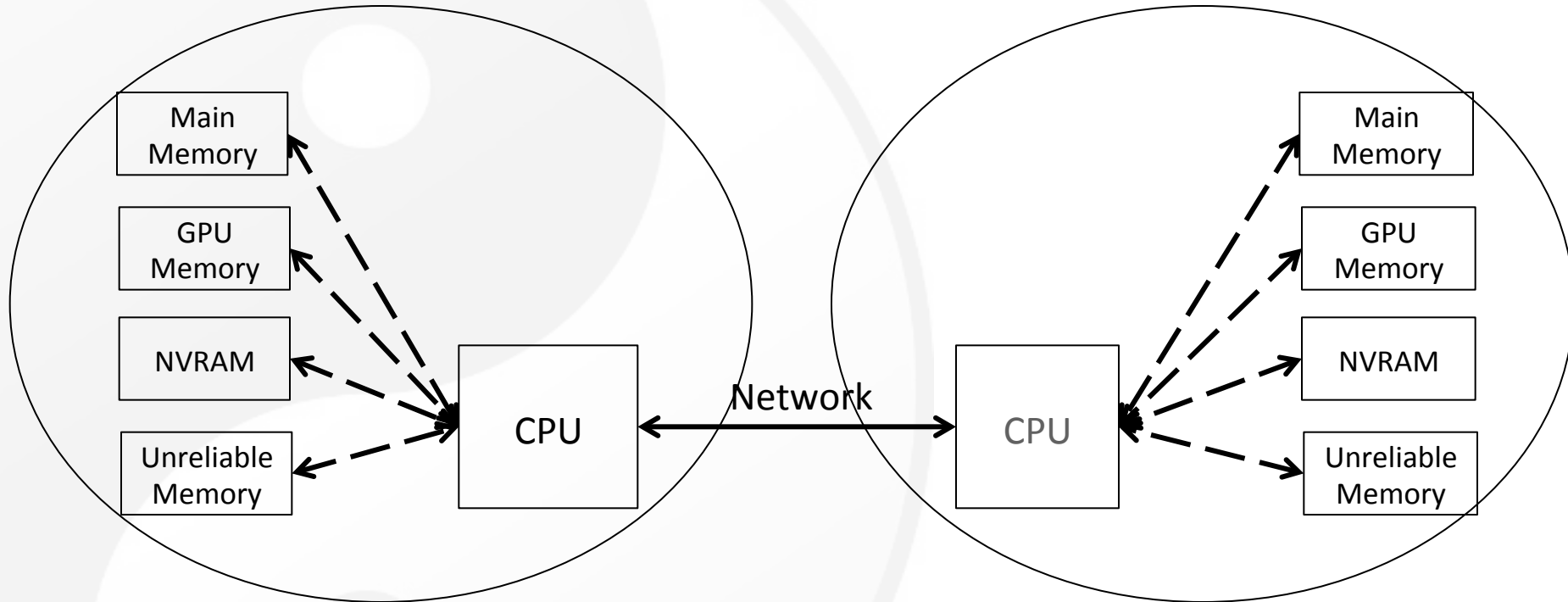


```
if(rank == 0)
{
  MPI_Send(any_buf, .. ..)
}
```

```
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

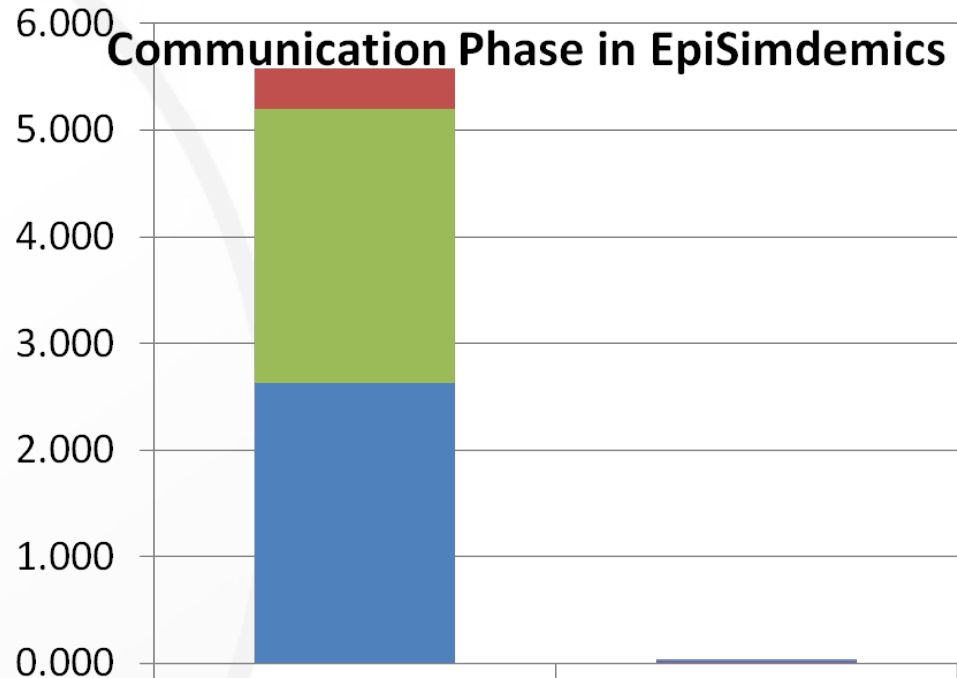
```
if (rank == 1)
{
    MPI_Recv(r_buf, .. ..);
}
```

MPI+CUDA vs. MPI-ACC

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

Slated to be integrated into the next official major release of MPICH

Time (seconds)



	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

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

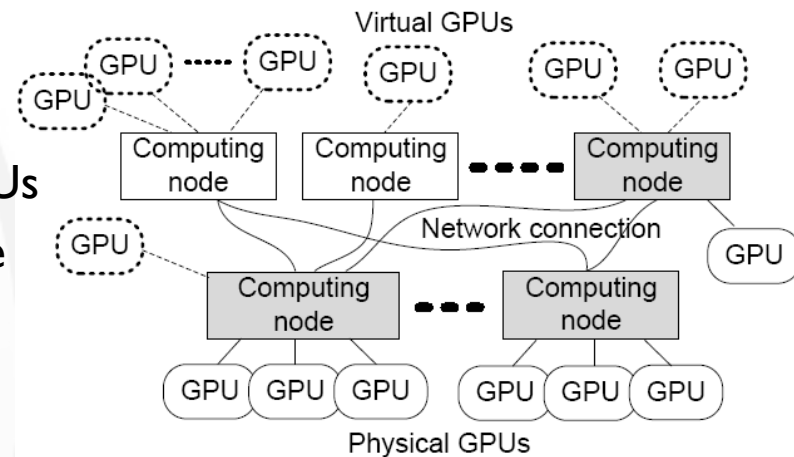
- GPU Virtualization

- 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

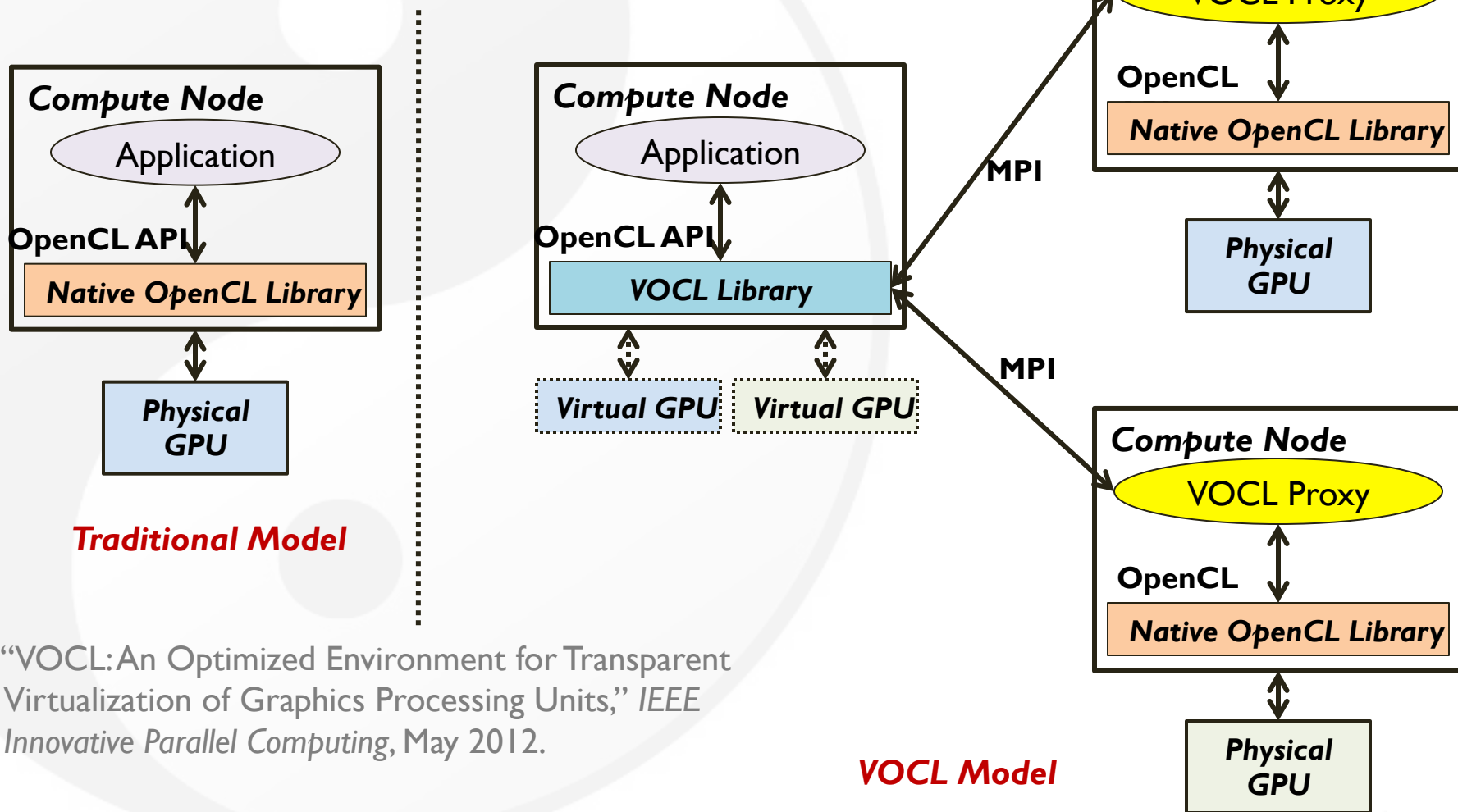
- Virtual GPUs can migrate from one physical GPU to another
 - If a system admin wants to add or remove a node, he/she can do that while applications are running (hot-swap capability)



- “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

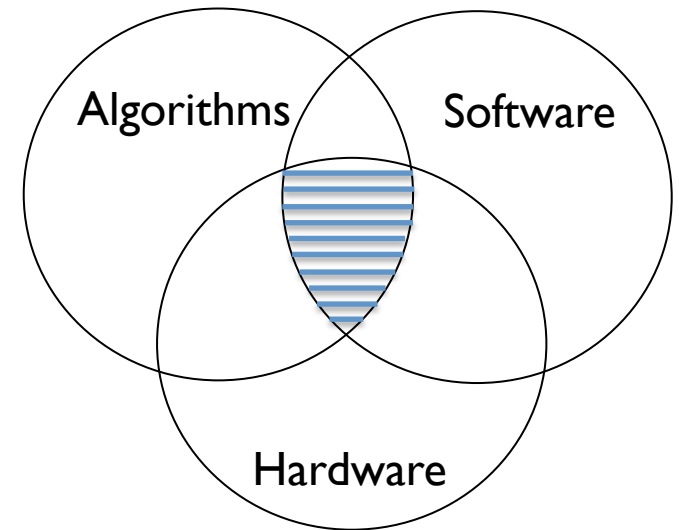
- Transparent utilization of remote GPUs
- Efficient GPU resource management



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

Roadmap

- Vision
- Team
- Approach: Synergistic Co-Design
- **Artifacts**
 - **Optimized CFD Codes**
 - **CoreTSAR / AffinityTSAR**
 - **MetaMorph**
 - MPI-ACC
 - VOCL:Virtual OpenCL
- **Achievements & Publications**
- **Next Steps**



Sampling of Achievements

- Computer Science
 - At Run Time: Automated run-time system that maps the right task(s) to the right processor at the right time for best performance → vendors
 - Identified commonality for library for CFD codes: generalized GPU-to-GPU communication (via MPI), ghost-cell exchange between GPUs, ...

Publications in 2014 (1 of 3)

1. B. Pickering, C. Jackson, T. Scogland, W. Feng, C. Roy, "Directive-Based GPU Programming for Computational Fluid Dynamics," *52nd AIAA Aerospace Sciences Meeting (SciTech)*, National Harbor, MD, Jan. 2014.
2. 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 (SciTech)*, National Harbor, MD, Jan. 2014.
3. Y. Xia, H. Luo, L. Luo, J. Edwards, J. Lou, F. Mueller, "OpenACC-based GPU Acceleration of a 3-D Unstructured Discontinuous Galerkin Method," *52nd AIAA Aerospace Sciences Meeting (SciTech)*, National Harbor, MD, Jan. 2014.
4. L. Luo, J. Edwards, H. Luo, F. Mueller, "Performance Assessment of Multi-block LES Simulations using Directive-based GPU Computation in a Cluster Environment," *52nd AIAA Aerospace Sciences Meeting (SciTech)*, National Harbor, MD, Jan. 2014.
5. C. Li, Y. Yang, H. Dai, S. Yan, F. Mueller, H. Zhou, "Understanding the Tradeoffs between Software-Managed vs. Hardware-Managed Caches in GPUs," *IEEE International Symposium on Performance Analysis of Systems and Software*, Mar. 2014.
6. T. Scogland, W. Feng, B. Rountree, B. de Supinski, "CoreTSAR: Adaptive Worksharing for Heterogeneous Systems," *Int'l Supercomputing Conf.*, Leipzig, Germany, Jun. 2014

Publications in 2014 (2 of 3)

7. L. Luo, J. Edwards, H. Luo, F. Mueller, “GPU Port of a Parallel Incompressible Navier-Stokes Solver based on OpenACC and MVAPICH2,” *AIAA Aviation 2014*, Atlanta, GA, Jun. 2014.
8. Y. Xia, L. Luo, H. Luo, J. Lou, J. Edwards, F. Mueller, “On the Multi-GPU Computing of a Reconstructed Discontinuous Galerkin Method for Compressible Flows on 3D Hybrid Grids,” *7th AIAA Theoretical Fluid Mechanics Conference*, Atlanta, GA, Jun. 2014.
9. A. Amritkar, D. Tafti, P. Sathre, K. Hou, S. Chivakula, W. Feng. “Accelerating Bio-Inspired MAV Computations using GPUs.” *AIAA Aviation and Aeronautics Forum and Exposition 2014*, Atlanta, GA, Jun. 2014.
10. K. Swirydowicz, E. de Sturler, X. Xu, C. Roy, “Fast Solvers and Preconditioners,” *SIAM Annual Meeting*, Chicago, IL, Jul. 2014.
11. A. Amritkar, D. Tafti. “CFD Computations using Preconditioned Krylov Solver on GPUs.” *ASME 2014 Fluids Engineering Division Summer Meeting*, Chicago, IL, Aug. 2014.
12. K. Swirydowicz, A. Amritkar, E. de Sturler, D. Tafti. “Recycling Krylov Subspaces for CFD Application,” *ASME 2014 Fluids Engineering Division Summer Meeting*, Chicago, IL, Aug. 2014.

Publications in 2014 (3 of 3)

13. E. Zharovsky, A. Sandu, H. Zhang, “A Class of IMEX Two-step Runge-Kutta Methods,” *SIAM Journal on Numerical Analysis*, 2014.
14. H. Zhang, A. Sandu, “FATODE: A Library for Forward, Adjoint, and Tangent Linear Integration of ODEs,” *SIAM Journal on Scientific Computing*, 2014.
15. P. Tranquilli, A. Sandu, “Rosenbrock-Krylov Methods for Large Systems of Differential Equations,” *SIAM Journal on Scientific Computing*, 36(3):1313-1338, 2014.
16. A. Cardone, Z. Jackiewicz, A. Sandu, H. Zhang, “Extrapolated IMEX Runge-Kutta Methods,” *Mathematical Modelling and Analysis*, 19(1):18-43, 2014.
17. A. Cardone, Z. Jackiewicz, A. Sandu, H. Zhang, “Extrapolation-Based Implicit-Explicit General Linear Methods,” *Numerical Algorithms*, 65(3):377-399, 2014.
18. H. Zhang, A. Sandu, “Application of implicit-explicit general linear methods to reaction diffusion problems,” *12th International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2014)*, Rodos Palace Hotel, Rhodes, Greece, September 2014.

Thesis Manuscripts in 2014

- Nishanth Balasubramanian, “ScalaMemAnalysis: A Compositional Approach to Cache Analysis of Compressed Memory Traces,” M.S. Thesis, Dept. of Computer Science, North Carolina State University, Jun. 2014. *Now at NVIDIA.*
- Brent Pickering, “Evaluating the OpenACC API for Parallelization of CFD Applications,” M.S. Thesis, Dept. of Aerospace and Ocean Engineering, Virginia Tech, Jul. 2014. *Now pursuing Ph.D. at Virginia Tech.*
- Thomas Scogland, “Runtime Adaptation for Autonomic Heterogeneous Computing,” Ph.D. Thesis, Dept. of Computer Science, Virginia Tech, Aug. 2014. *Heading to DOE Lawrence Livermore National Laboratory.*

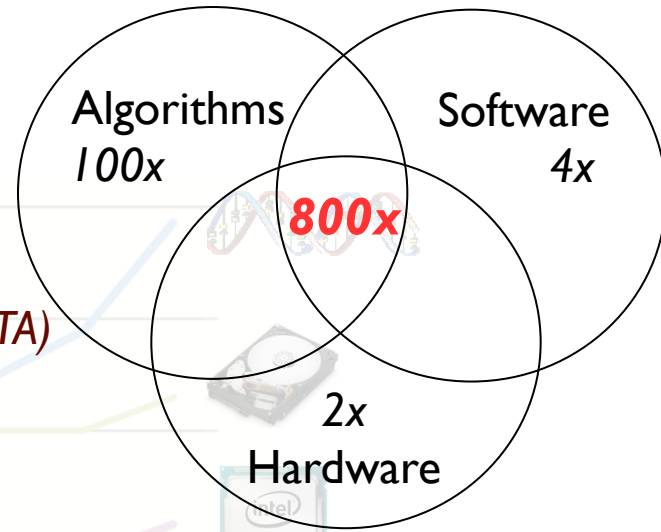
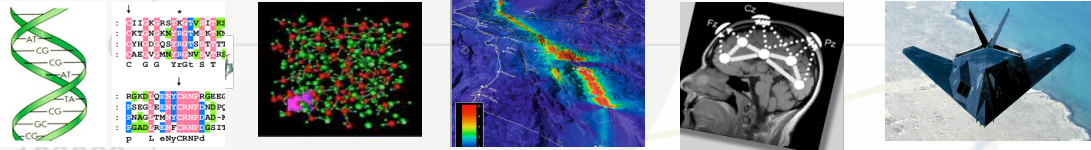
Synergistic Co-Design for BIG DATA

Prof. Wu Feng, Virginia Tech

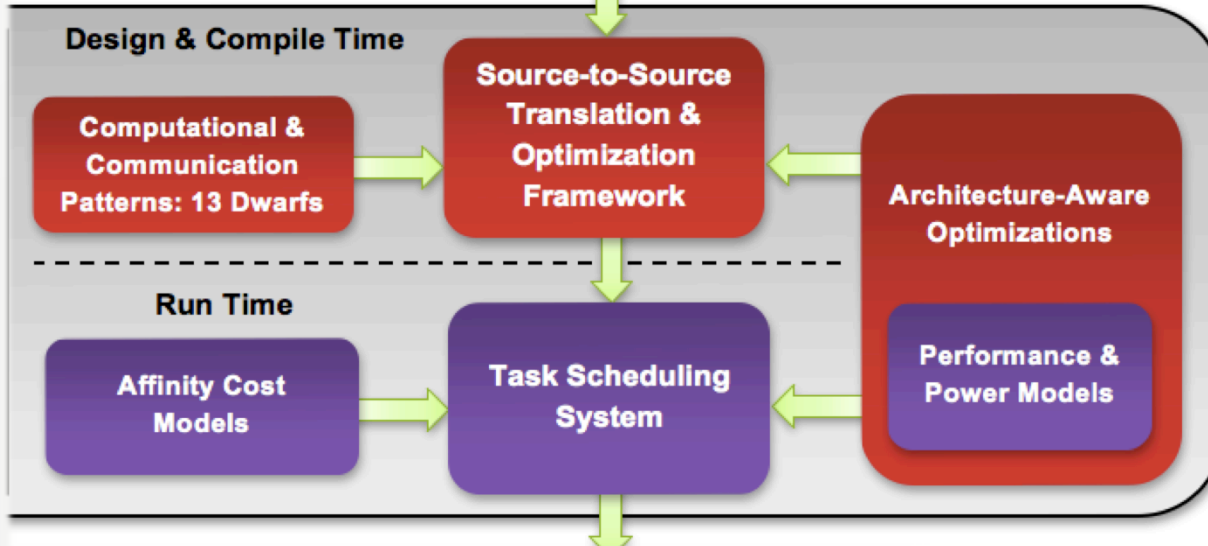
Changing Landscape

... from FLOPS ("old HPC") to bytes ("new HPC" → BIG DATA)

Apps

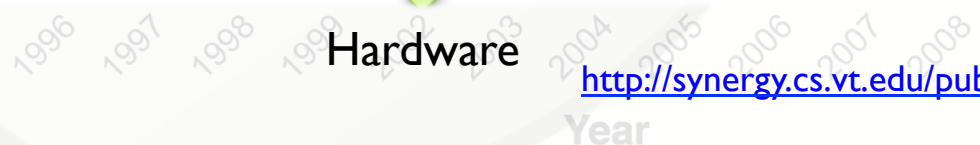


Software Ecosystem



Co-Design Exemplars

- ParaMEDIC: Parallel Metadata Environment for Distributed I/O & Computing (yrs → mins)*
→ Find missing genes
<http://archive.isgtw.org/?pid=1000811>
- Molecular Modeling*
→ Rational drug design
<http://www.youtube.com/watch?v=zPBFenYg2Zk>
- Temporal Data Mining of Brain*
<http://synergy.cs.vt.edu/pubs/papers/feng-temporal-data-mining-gpu-gems-2011.pdf>

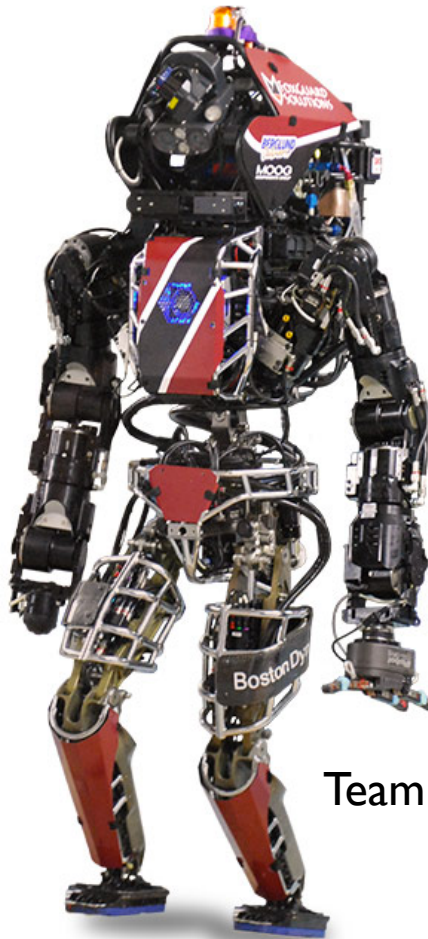


Synergistic Co-Design for DARPA Challenges



Height: 5ft 10in (1.78m)
Weight: 143lbs (65kg)
Wingspan: 82in (2.08m)

Team Valor



Team ViGIR

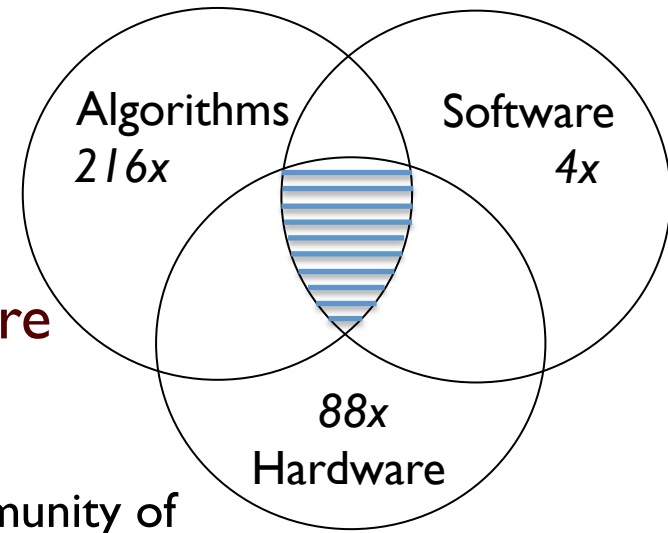


<http://www.theroboticschallenge.org/>

II Finalists (w/ two from Virginia Tech)

- Team ViGIR
- Team Valor

AFOSR Basic Research Initiative: Transformational Computing via Co-Design of High-Performance Algorithms and Hardware



This effort is envisioned as a “MURI-like” effort with collaborative grants to teams that will create a new community of researchers skilled in the design, development, and deployment of systems tuned to maximize the synergy of advanced algorithms and high-performance hardware.

New Center at Virginia Tech:
Synergistic Environments for Experimental Computing (SEEC)



What's Next? (Last Year)

- Platforms
 - AMD & Intel CPU, AMD APU, AMD & NVIDIA GPUs, Intel MIC
- Towards Ease of Use and Automation (for Performance, Programmability, and Portability)
 - Webresource 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

Plans for Upcoming Year

SENSEI-LDC

- Publish article with de Sturler's group on preconditioners/solvers for GPU
- Collaborate w/ Mueller's group on MemTrace (explicit and implicit codes)

SENSEI-Lite

- Complete code development to include viscous terms & implicit Jacobian
- Collaborate with Sandu's group on time accurate solutions
- Collaborate with de Sturler's group on preconditioners/solvers

SENSEI

- Improve general preconditioners/solvers and implementation on GPU w/ de Sturler's group
- Develop strategy for handling function pointers and allocatables within OpenACC w/ Feng's group
- Implement OpenACC directives in SENSEI code base w/ Feng's group

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: 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

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.

Mobile and Desktop Supercomputing

- Description
 - AMD GPUs (4), NVIDIA GPUs (4), AMD APUs (2), Intel MIC (2)
- Pictures of above
 - Note early access to potpourri of experimental architectures
- Make sure to give 5-second blurb to “memory-unlimited” GPU, i.e., AMD Radeon 7990 and 8970.
 - How? Virtual memory addressing. Limited only by system memory.

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





APPENDIX

HPL: High-Performance Linpack

- A benchmark that
 - Solves a dense n by n system of linear equations $Ax = b$. (Used since 1993 by TOP500 to rank the fastest supercomputers in the world.)
 - Seeks to approximate how fast a computer will perform when solving real problems.
 - Number of Operations: $\frac{2}{3}n^3 + 2n^2$
- Issue
 - LINPACK
 - Top500 (
 - Floating floating po
 - operations
 - No longer apps.
 - Reports see only I
 - Encourag features
 - Overall u
 - Used as

HPCG: High-Performance Conjugate Gradient

What is an AMD APU?

Paradox Solved: One Design, Fewer Watts, Massive Capability

Northbridge



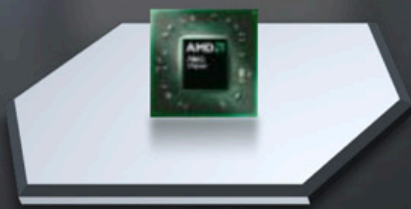
Dual-Core
CPU



Discrete-level
DirectX® 11
GPU



AMD
Fusion
APU



- 66 sq. mm
- 13 watts



- 117 sq. mm
- 25 watts



- 59 sq. mm
- 8 watts



- 75 sq. mm
- 18 watts



Collaborations with Math

Collaboration on the implicit SENSEI-LDC code

- Focus is on solvers and preconditioners
- Maximum efficiency is found when considering interactions between: matrix storage format, memory use, hardware, preconditioner, solver
- K. Swirydowicz, E. de Sturler, X. Xu, and C. J. Roy, “Fast Solvers and Preconditioners,” SIAM Annual Meeting, Chicago, IL, July 7-11, 2014

Collaboration on SENSEI

- SENSEI uses modern Fortran, but includes ISO-C bindings so we can interface with existing solvers in C
- SENSEI uses a built in CPU solver library (Fortran), but has recently been extended towards GPU functionality using the CUDA ITSOL interface (C); this is the same interface used by de Sturler’s group
- The folks in Math should now have access to the SENSEI GIT repository

Collaborations with CS

Collaboration w/ Feng's group: SENSEI-LDC and SENSEI

- Worked with Tom Scogland to get explicit SENSEI-LDC code running on multiple GPUs (AIAA Paper, journal submission in progress)
- Developed plan for GPU-parallelizing SENSEI using OpenACC
- B. P. Pickering, C. W. Jackson, T. R. W. Scogland, W.-C. Feng, and C. J. Roy, "Directive-Based GPU Programming for Computational Fluid Dynamics," AIAA Paper 2014-1131, 52nd Aerospace Sciences Meeting, National Harbor, MD, January 13-17, 2014


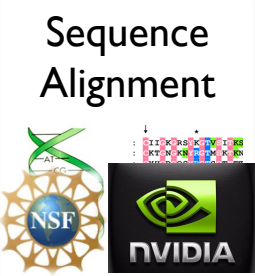
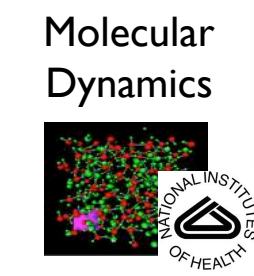
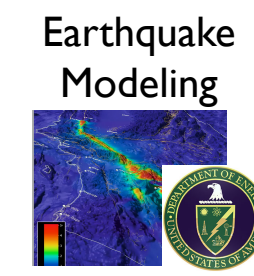
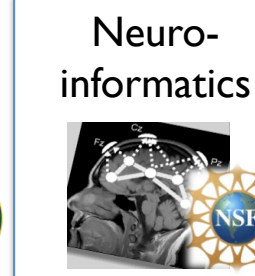
Collaboration w/ Sandu's group: SENSEI-Lite

- Developed a MATLAB version of SENSEI: the "real" SENSEI-Lite
- Current code capabilities: structured grid, general geometry, finite volume method, single block, inviscid, and explicit solver
- Upcoming capabilities: viscous (Navier-Stokes) & implicit w/ full Jacobian
- Sandu's group currently has access to code through github
- Sandu's group will use the implicit code for studying their IMEX and ROK/EXPK schemes for time accurate simulations

Intra-Node

Ecosystem for Heterogeneous Parallel Computing

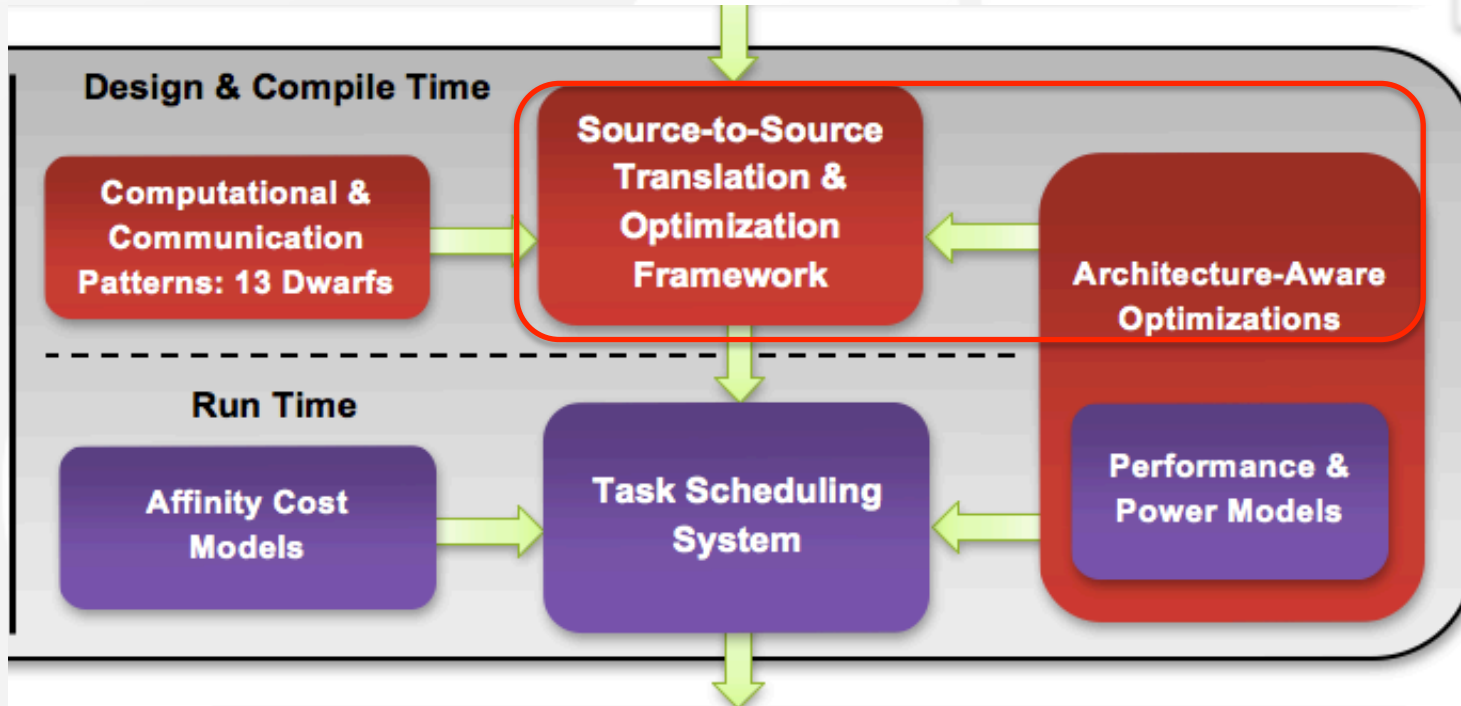
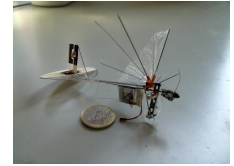
MANUAL CO-DESIGN

Epidemiology 	Sequence Alignment 	Molecular Dynamics 	Earthquake Modeling 	Neuro-informatics 
---	--	--	--	---

Cybersecurity



MAVs



Manual
Co-Design
→
Automated
Co-Design

Heterogeneous Parallel Computing Platform


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


Overview of MAV Requirements

Algorithm Choice	Processing Req'ts	Communic. Req'ts	Memory Req'ts
Grid Type			
<i>Structured</i>	High	High	High
<i>Unstructured</i>	High	High	High
Spatial Discretization			
<i>Finite Difference</i>	Medium	High	High
<i>Finite Volume</i>	High	High	High
<i>Finite Element</i>	High	High	High
Temporal Discretization			
<i>Explicit</i>	Low	Low	Low
<i>Implicit (line)</i>	High	High	High
<i>Implicit (plane)</i>	High	High	High
<i>Implicit (volume)</i>	High	High	High


High



Medium



Low



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)

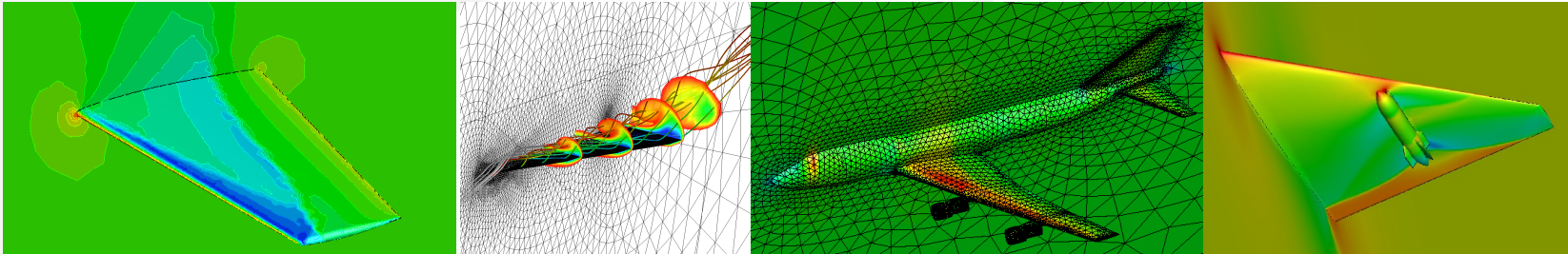
- Structured, multiblock finite volume code
- Second or higher order spatial accuracy
- ALE and IBM

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.



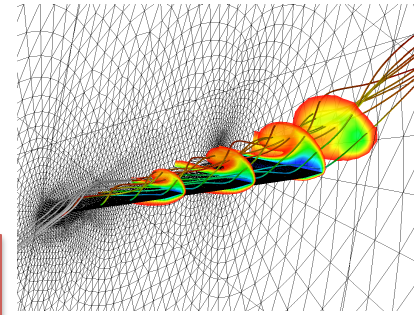
Need GPU acceleration for RDGFLO because ...

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, ...

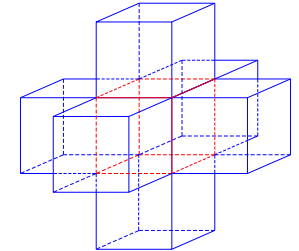
OpenACC does not require much change of data structures and algorithms in a legacy code.

RDGFLO: GPU Parallelization



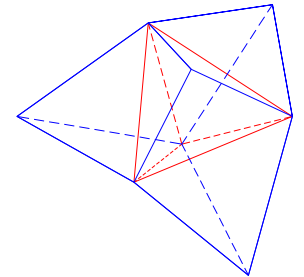
Race condition occurs in GPU parallelization in loops over faces when data writes to their left and right cell arrays.

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).

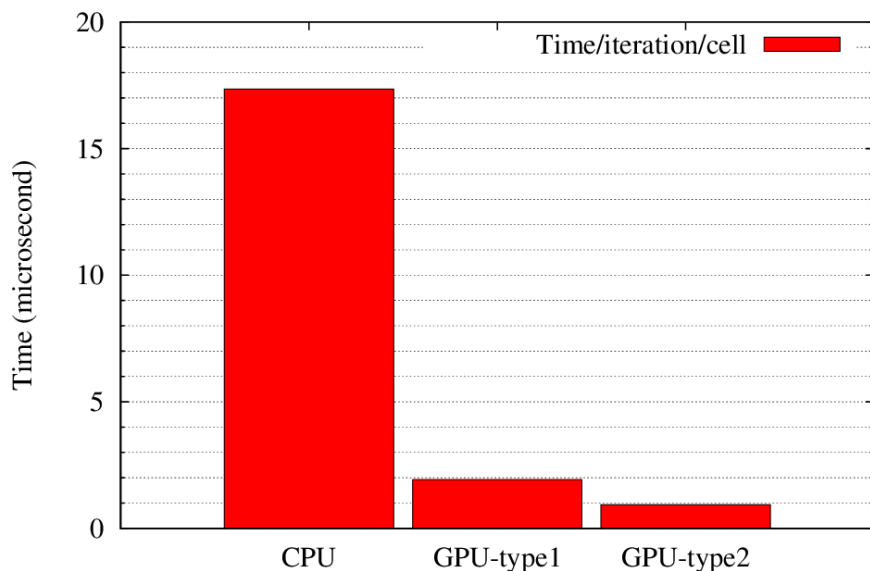
Example: an ACC sequential loop over the groups is nested outside the ACC parallel loop over the faces (the original loop is untouched).



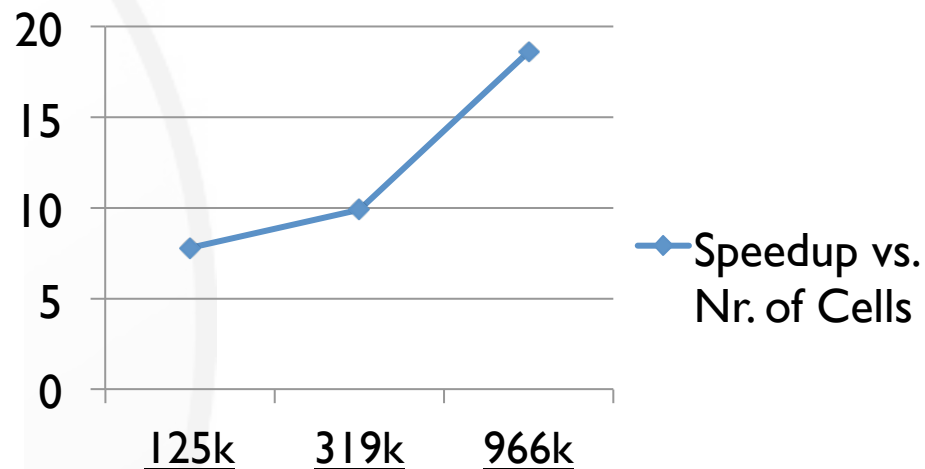
All geometric and solution arrays are copied from host to device only once. For steady-state problems, solution arrays are only copied back to host memory at the end of time iterations and dumped in files.

RDGFLO: Weak Scaling Tests

Wall clock profiling: CPU vs. GPU



Speedup vs. Nr. of Cells



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)

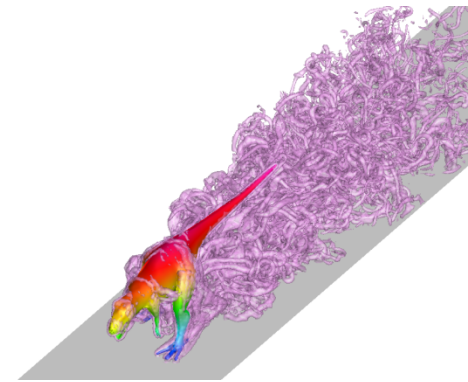
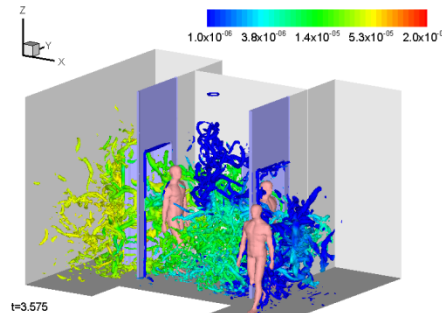
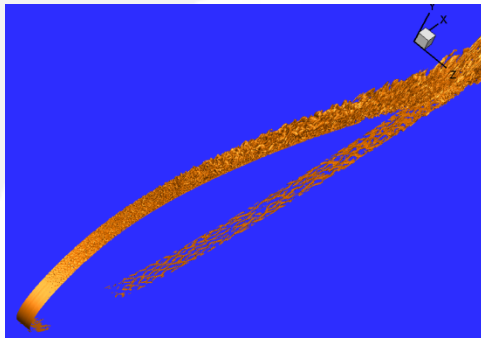
- Structured, multiblock finite volume code
- Second or higher order spatial accuracy
- ALE and IBM

INCOMP3D: Overview

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

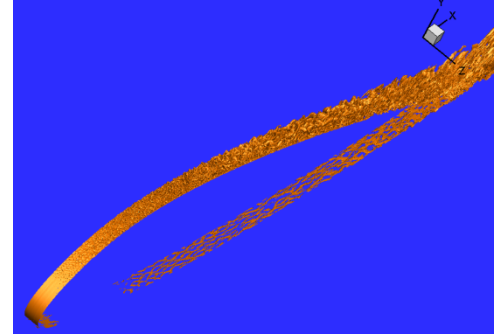
Key Features

- 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

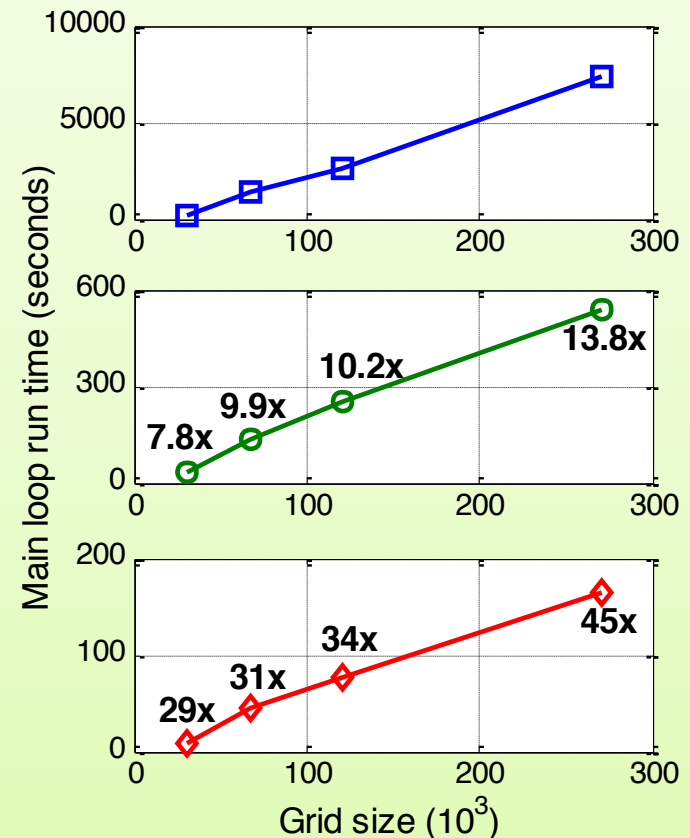


- Two scaled-down versions realized
- OpenACC (ACC) port
 - Main loop is carried out on GPU only. All essential arrays remain on GPU main memory. Temporary data arrays are created directly on GPU.
 - One code can be compiled into pure CPU or GPU-accelerated versions, using different compiler options. 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 ($\sim 10x$), 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.



Upcoming Tasks (Edwards)

- Accuracy study and optimization of 2-stage BILU(0)
 - Storage of factorized $D \downarrow_{i,j,k}$ is different from the original algorithm.
 - Triangle substitution needs modification.
 - Solver accuracy may be slightly different from the original codes.
- Scheduling multiple blocks on one GPU – rehashing
 - Balance must be found between “size of blocks” vs. “number of blocks”.
- Improvement on MPI
 - Recently available hardware GPU RDMA may improve MPI efficiency.
 - General data exchange libraries can be adopted.
- Atomic operations instead of coloring schemes
 - Atomic operations are now available in PGI’s latest compilers supporting OpenACC 2.0.
- Two-level synchronization in wavefront scheme
- Further optimizations of other kernels (flux, BC, LHS)

Collaboration (Edwards)

- Collaboration with F. Muller's group (NCSU CS)
 - Invaluable cluster and software support by Muller's group.
 - Technical challenges on OpenACC and CUDA are actively discussed.
- Collaboration with RDGFLO3D (NCSU MAE)
 - Repositories of codes are set up for easy access.
 - Common algorithms on implicit methods are shared, reducing code development effort.
 - Y. Xia, L. Luo, H. Luo, J. Lou, J. Edwards, F. Mueller, "On the Multi-GPU Computing of a Reconstructed Discontinuous Galerkin Method for Compressible Flows on 3D Hybrid Grids," AIAA Aviation 2014, Georgia
- Incorporation of GPU-aware functionalities of MVAPICH2 (VT CS)
 - With the support on MVAPICH2 from Hao Wang, INCOMP3D is able to conduct efficient data transfers across GPUs on different cluster nodes.
 - Portability of the data transfer codes is greatly improved.
 - H. Wang, S. Potluri, D. Bureddy, C. Rosales, D. K. Panda, "GPU-Aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation," IEEE Transactions on Parallel and Distributed Systems, vol. 99, PrePrints, 2014.
- Advanced wavefront scheme for the implicit solvers (VT CS)
 - An advanced synchronization scheme pioneered by Feng's group (VT CS) is being studied.
 - S. Xiao, W. Feng, "Inter-Block GPU Communication via Fast Barrier Synchronization," 24th IEEE International Parallel and Distributed Processing Symposium, Atlanta, Georgia, April 2010.

Publications in 2013

- 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.

Publications (Under Review and In Preparation)

- Under Review

- P. Tranquilli, A. Sandu, “Exponential-Krylov Methods for Ordinary Differential Equations,” *Journal of Computational Physics*.
- H. Zhang, A. Sandu, S. Blaise: High Order Implicit-Explicit General Linear Methods with Optimized Stability Regions,” *SIAM Journal on Scientific Computing*, MS 097670.
- A. Aji et al., “MPI-ACC: GPU-Integrated MPI for Scientific Applications,” *IEEE Transactions on Parallel & Distributed Systems*.
- T. Scogland, W. Feng, B. Rountree, B. de Supinski, “CoreTSAR: Core Task-Size Adapting Runtime,” *IEEE Transactions on Parallel & Distributed Systems*.
- J. Lou, Y. Xia, L. Luo, H. Luo, J. Lou, J. Edwards, F. Mueller, “OpenACC-based GPU Acceleration of a p -multigrid Discontinuous Galerkin Method for Compressible Flows on 3D Unstructured Grids,” *AIAA Science and Technology Forum*.

- In Preparation

- B. Pickering, C. Jackson, T. Scogland, W. Feng, C. Roy, “Directive-Based GPU Programming for Computational Fluid Dynamics,” in preparation for *Computers and Fluids*, Aug. 2014.
- J. Derlaga, T. Phillips, C. J. Roy, “SENSEI Computational Fluid Dynamics Code: A Case Study in Modern Fortran Software Development,” in preparation for *Journal of Aerospace Computing, Information, and Communication*, Aug. 2014.
- K. Swirydowicz, E. de Sturler, X. Xu, and C. Roy, “Effective Solvers and Preconditioners on GPUs for CFD Applications,” in preparation for *Parallel Computing*.
- A. Amritkar, E. de Sturler, K. Swirydowicz, D. Tafti, K. Ahuja, “Recycling Krylov Subspaces for CFD Applications,” in preparation for *Computer Methods in Applied Mechanics and Engineering*.
- A. K. Grim McNally, M. Li, E. de Sturler, S. Gugercin, “Preconditioning Parameterized Linear Systems,” in preparation for *SIAM J. Scientific Computing*.