Efficient Implementation of CFD Algorithms on GPU Accelerated Supercomputers

Ali Khajeh-Saeed

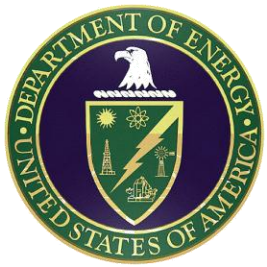
Software Engineer

CD-adapco

J. Blair Perot

Mechanical Engineering

UMASS, Amherst



Outline

- Supercomputers
- Optimization
- Stream Benchmark
- Stag++ (3D Incompressible Flow Code)
- Matrix Multiply Function (Parallel Nsight)
- Isotropic Turbulence Decay
- Speedup Results on Lincoln, Forge and Keeneland Supercomputers
- Summary

Lincoln, Keeneland and Forge Supercomputers

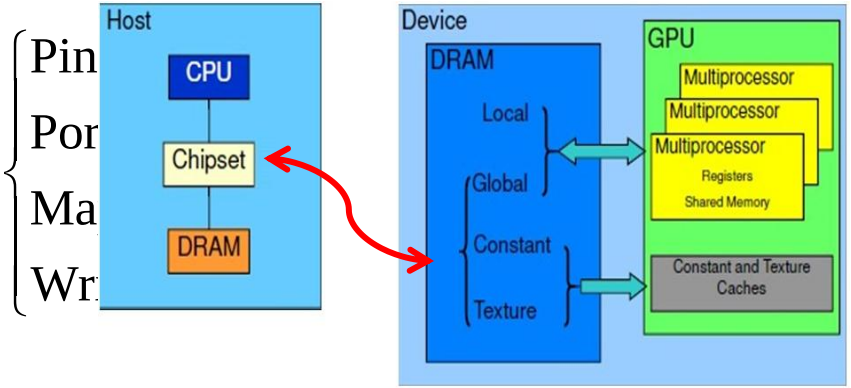
- 120 servers
- 360 Tesla M2070 NVIDIA Fermi GPUs (6 GB DDR5)
- two hex-core Intel Xeon Processors (12 cores per node)
- PCI-e Gen2 ×16

- 88 Tesla M2070 NVIDIA (6 GB DDR5)
- two eight-core AMD (16 cores per node)
- PCI-e Gen2 ×8 (36 servers)

Memory Architecture

Memory	Location	Cached	Access	Scope	Lifetime
Register	On-chip	N/A	R/W	One thread	Thread
Local	Off-chip	No	R/W	One thread	Thread
Shared	On-chip	N/A	R/W	All threads in a block	Block
Global	Off-chip	No	R/W	All threads + host	Application
Constant	Off-chip	Yes	R	All threads + host	Application
Texture	Off-chip	Yes	R	All threads + host	Application

Page-Locked Host Memory

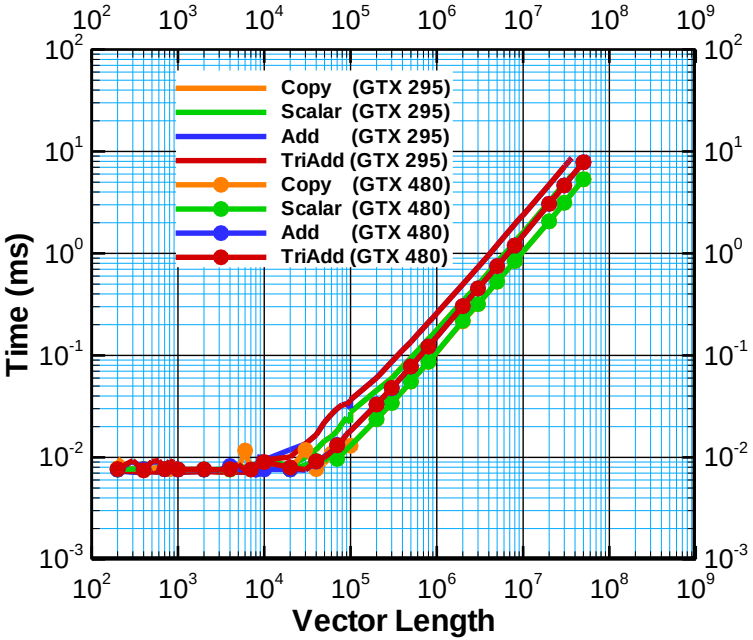


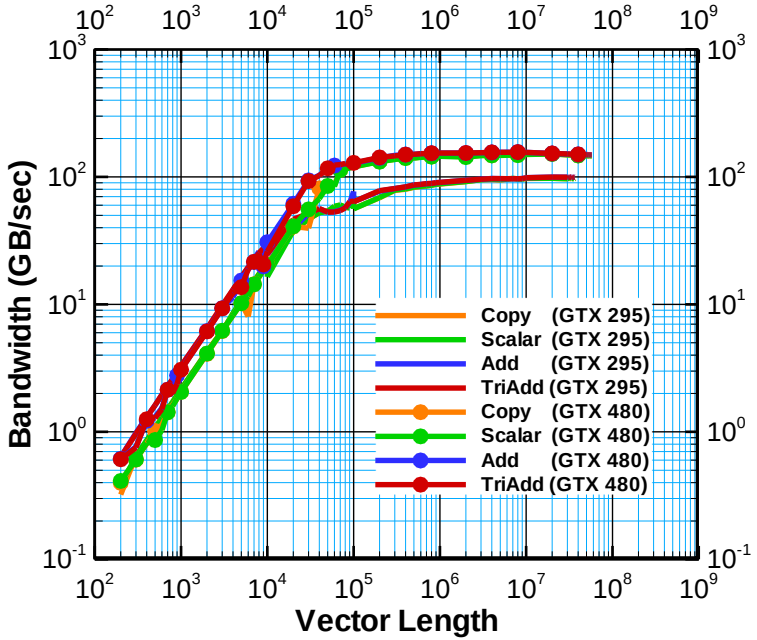
Performance Optimization Strategies (GPU)

- Minimize the data transfer between CPU (host) and GPU (device)
- Maximize coalesced access to the global memory where possible
- Use suitable memory type for the saving data (shared, texture, constant, mapped or write-combining memory)
- Overlap copying data with kernel launches where possible
- Run kernels concurrently where possible
- Minimize CUDA synchronization call
- Use single large copy instead of many small copies
- Maximize the occupancy where possible.

Stream Benchmark

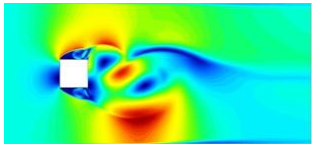
- ***a*** = ***b*** (one read and one write - Copy)
- ***a*** = *s****b*** (one read and one write - Scalar)
- ***a*** = ***b*** + ***c*** (two reads and one write - Add)
- ***a*** = *s****b*** + ***c*** (two reads and one write - TriAdd)



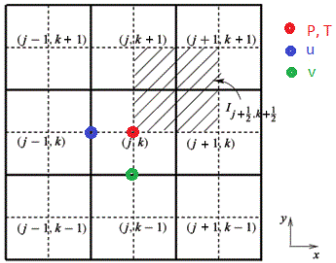


Stag++

- 3D incompressible

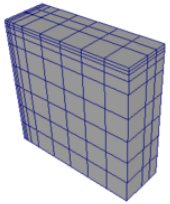


- Second-order staggered mesh



- Three-step Runge-Kutta

- Non-uniform spacing in all directions



- Parallel

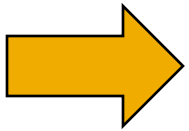
- CPU

- GPU

Optimizations

Kernel

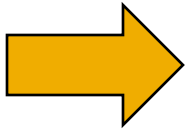
- Coalesced Access
- Shared Memory
- Maximize the Occupancy
- Minimize the Thread Divergence
- Use Faster Math Function



Compute Visual Profiler

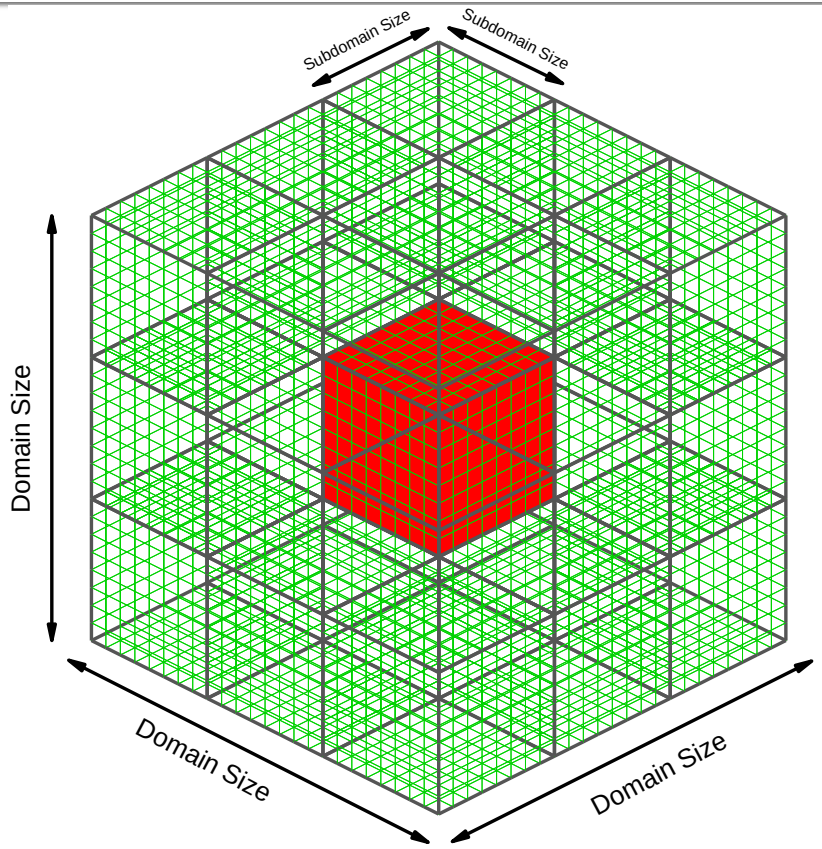
Algorithm

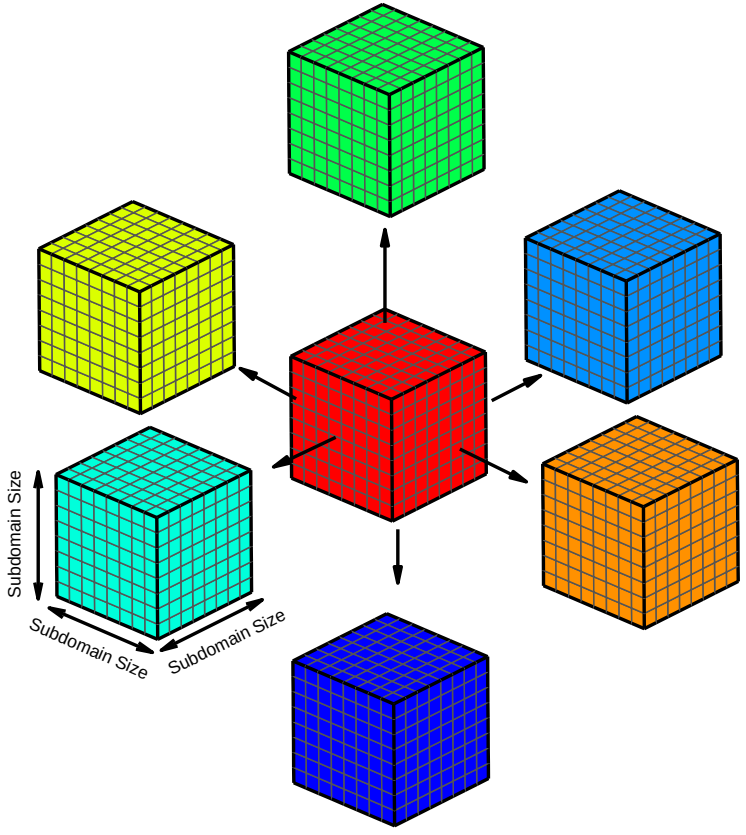
- Minimize Data Transfer
- Run Kernel Concurrently
- Minimize CUDA Synchronization
- Overlap Copying data with Kernel Launches
- Use Suitable Memory (Constant, Mapped, …)
- Single Large Copy Instead of Many Small Copies



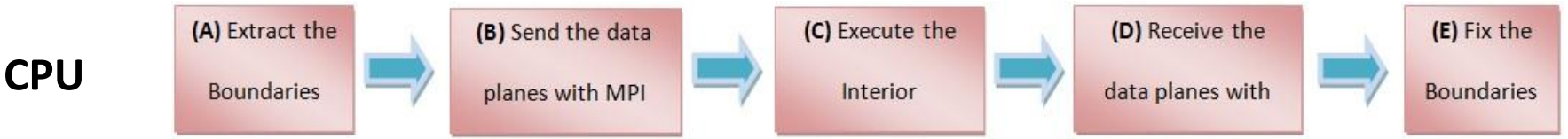
NVIDIA Parallel Nsight

Partitioning

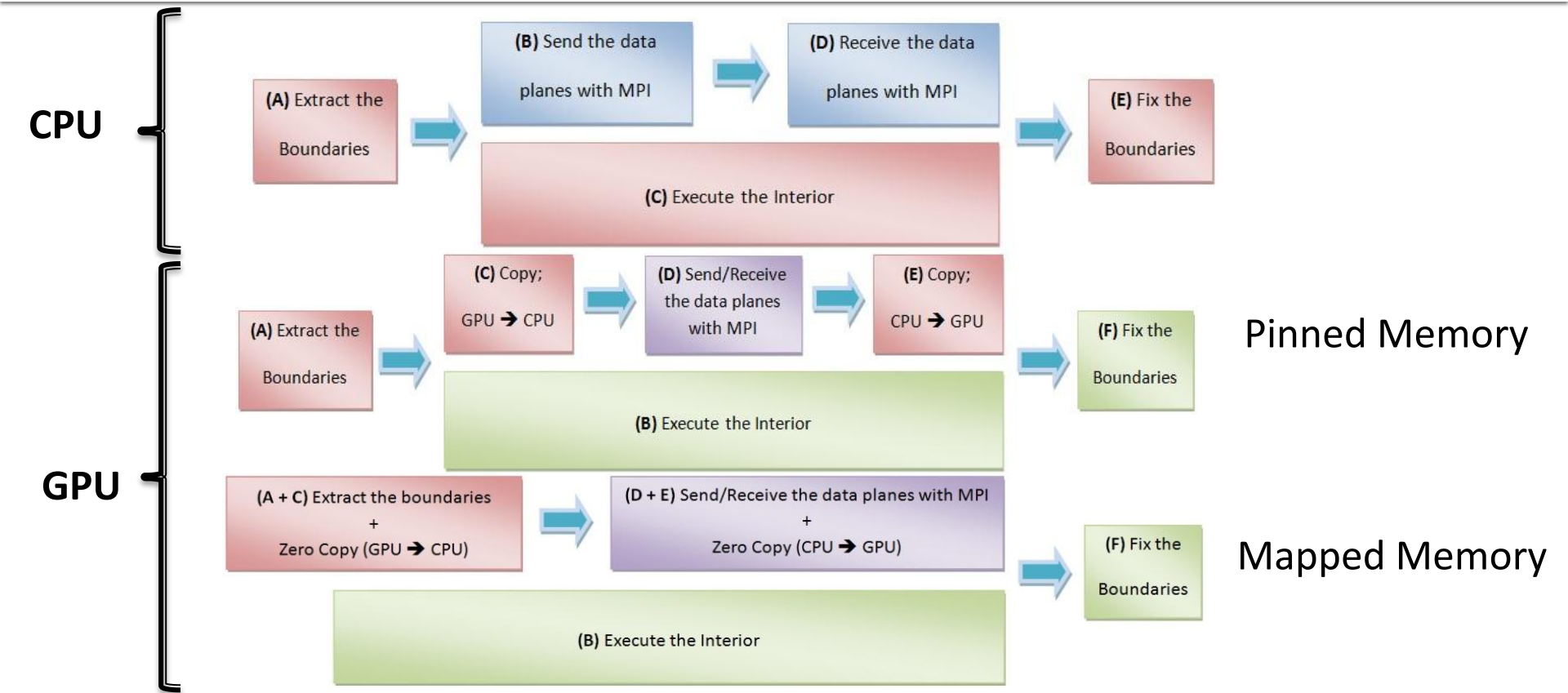




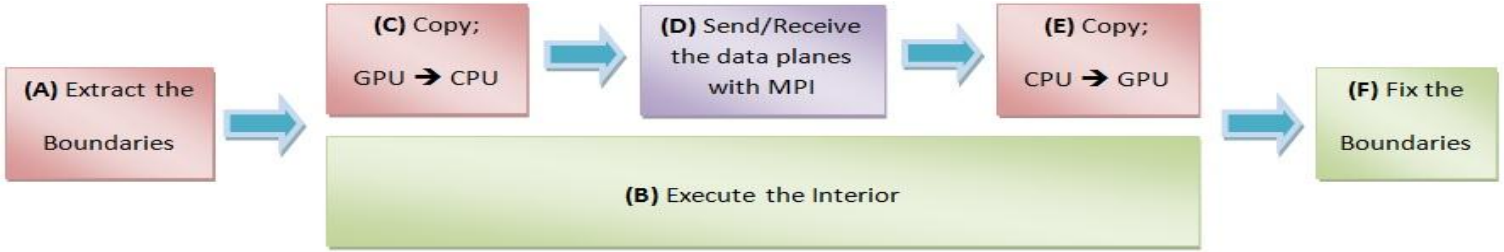
Matrix Multiply Function in Detail (Naïve)

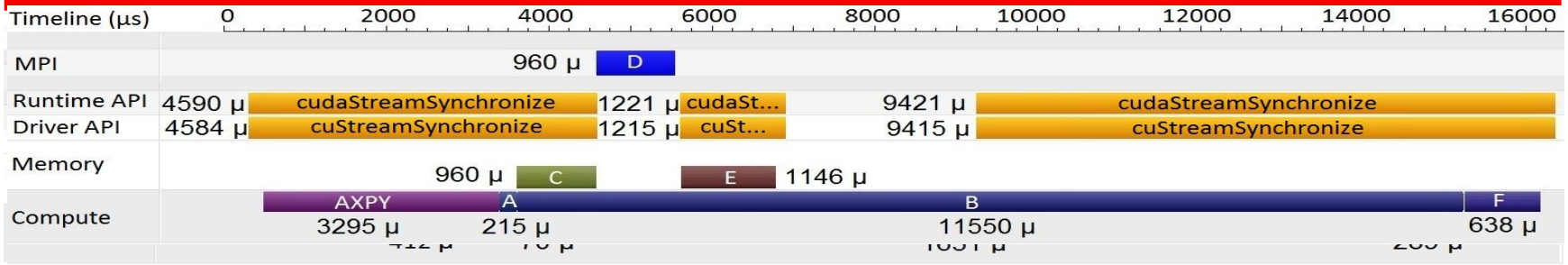


Matrix Multiply Function in Detail (Optimized)



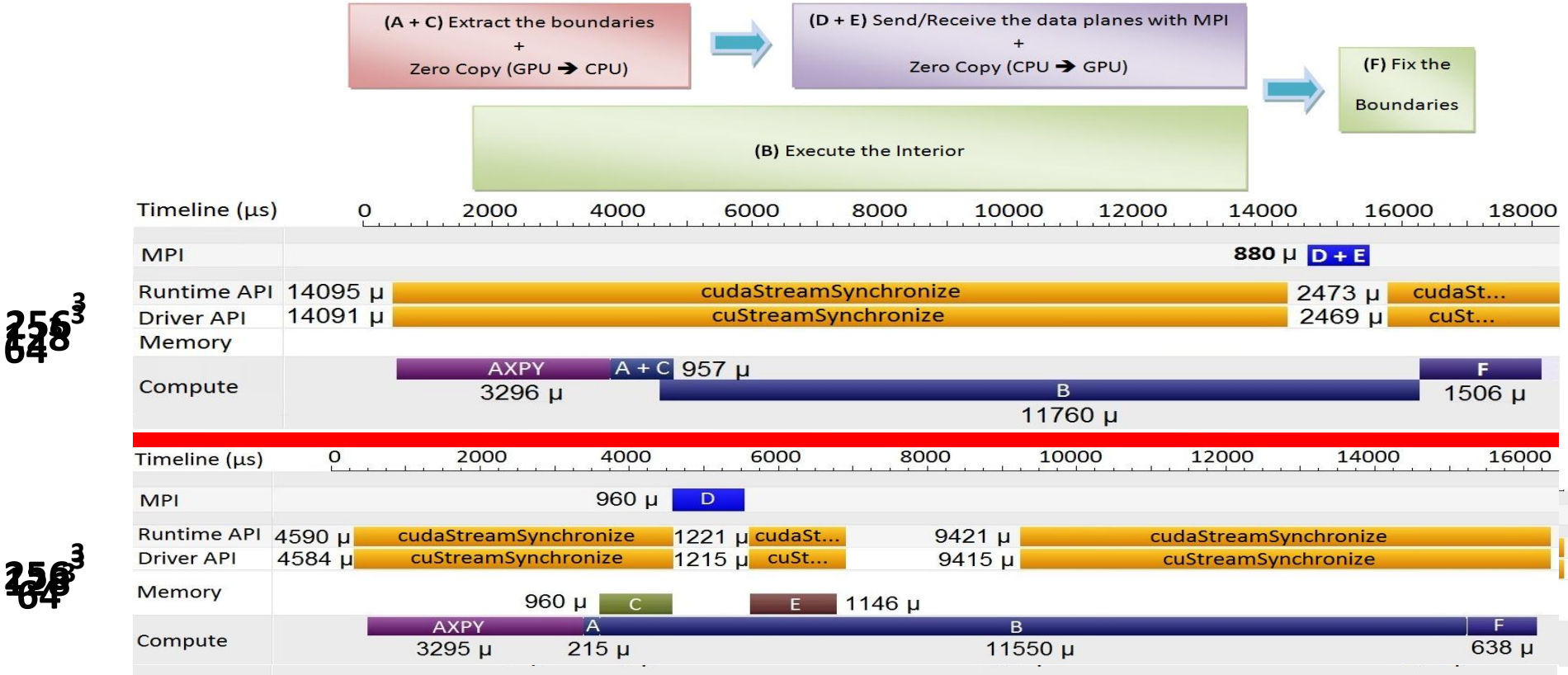
Matrix Multiply Function (Pinned Memory)



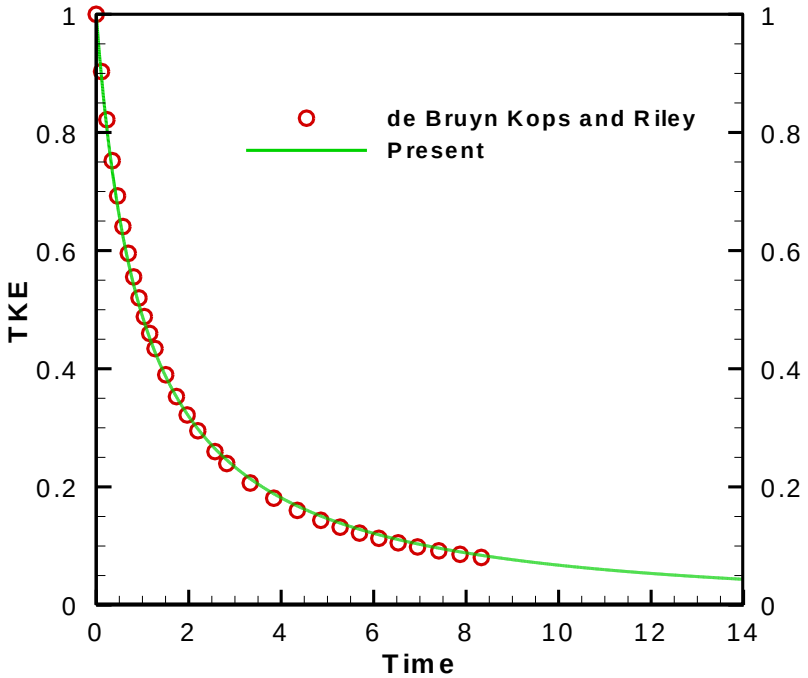


256^3
128
64

Matrix Multiply Function (Mapped Memory)



Isotropic Homogenous Decay

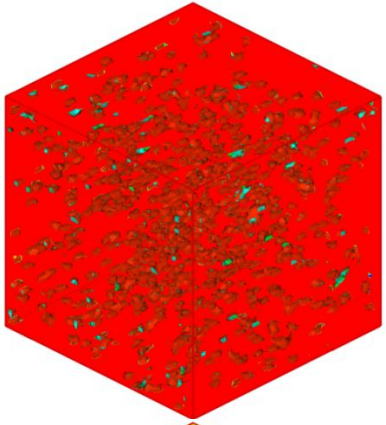


Turbulence Kinetic Energy

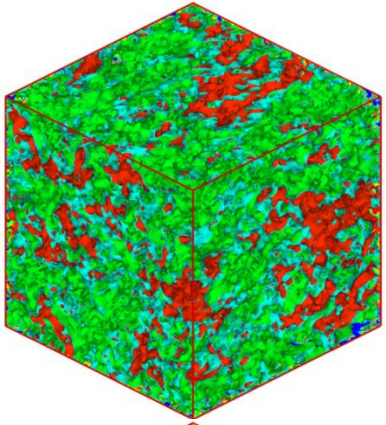
S. M. de Bruyn Kops, J. J. Riley, “Direct numerical simulation of laboratory experiments in isotropic turbulence,” Physics of Fluids 10 (9), 2125-2127 (1998)

Isotropic Homogenous Decay

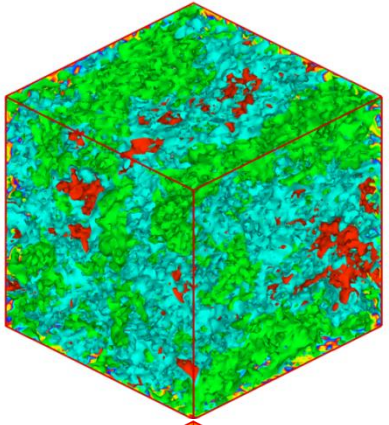
5s



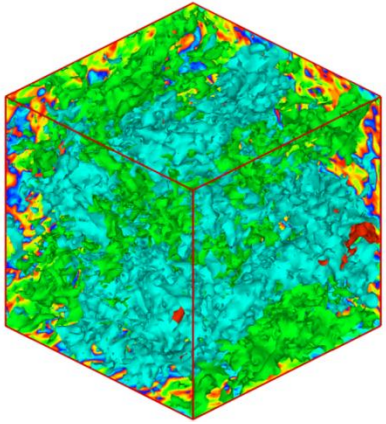
7s



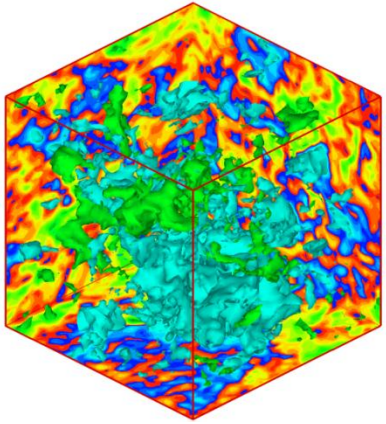
12s



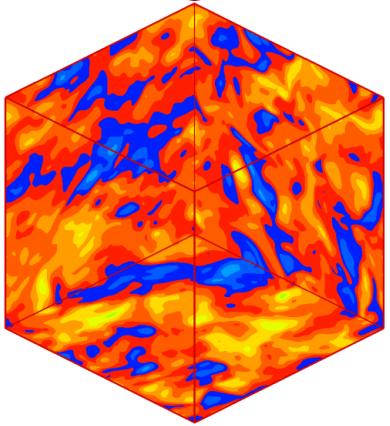
20s



40s

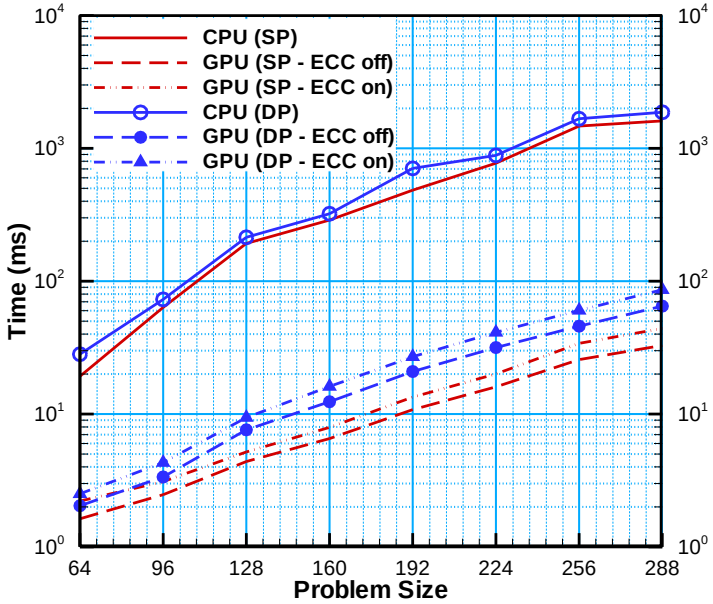


110s

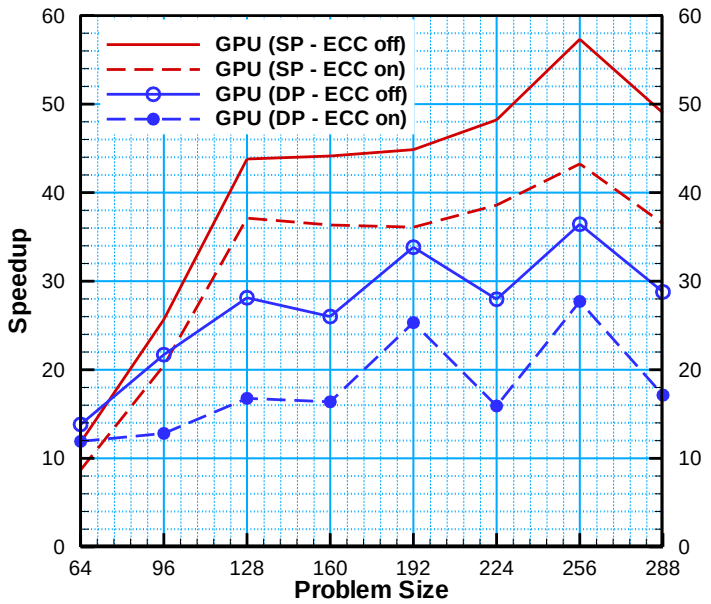


Orion (Single GPU)

- Single (SP) and Double (DP) precision
- SP is **2** times faster than DP

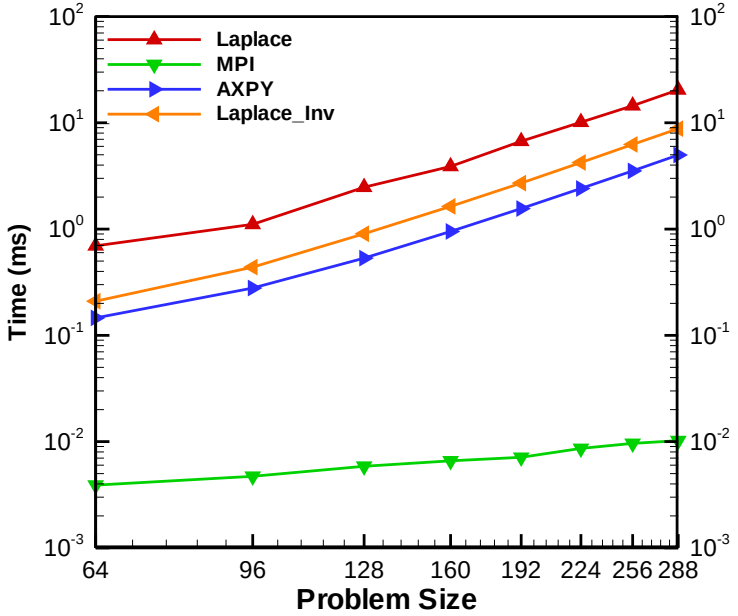


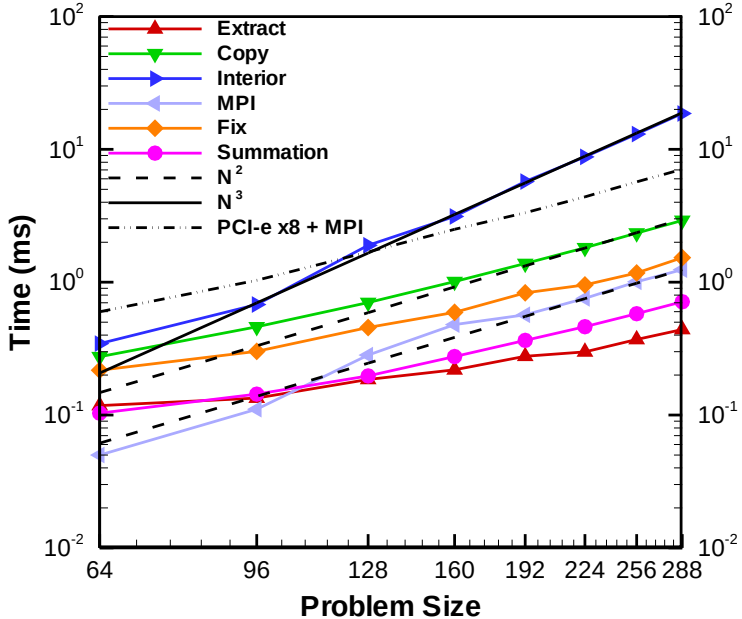
- ECC effects
- UP **58x** and **44x** speedup



Single GPU and PCI-e Effect

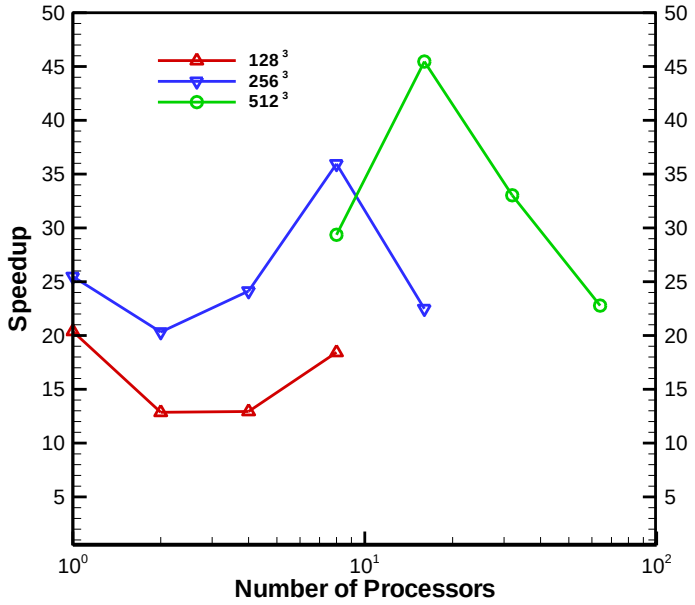
- CG (Conjugate Gradient) Solver
- PCI-e effects on Lincoln and Forge Supercomputers

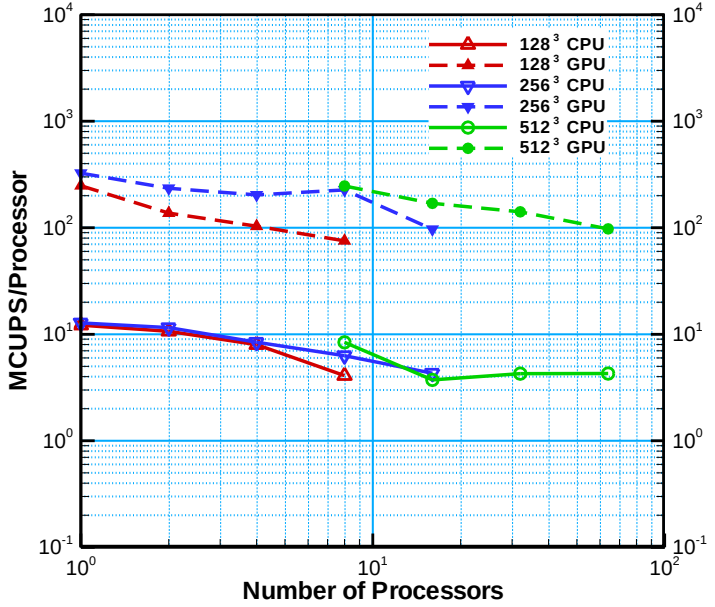




Lincoln (Strong Scaling)

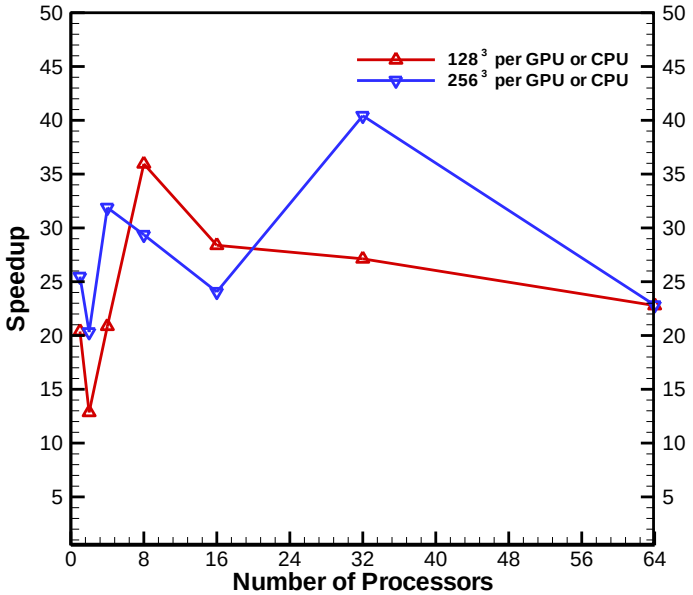
- UP **45x** speedup (16 GPUs Vs. 16 CPU cores)
- Reasonable performance loss from 2 GPUs to 64 GPUs (less than **50%**)
- Lincoln supercomputer has lower bandwidth between CPU and GPU (**2 GB/s** instead of **8 GB/s**)

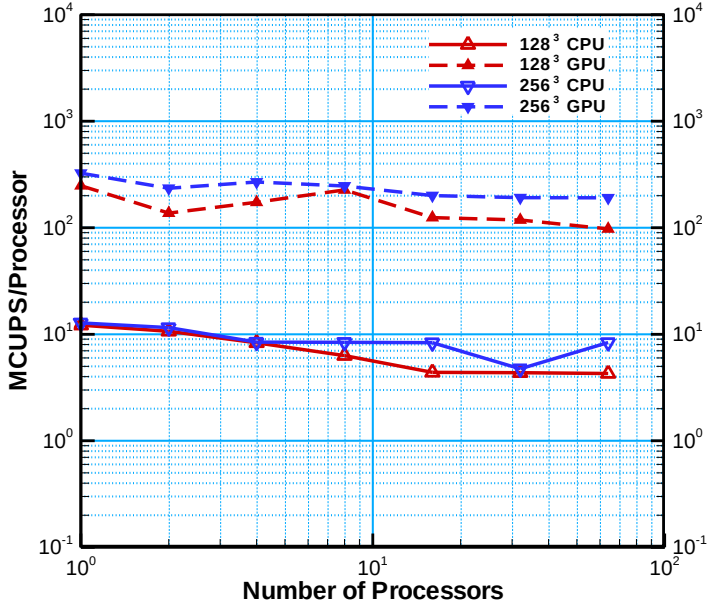




Lincoln (Weak Scaling)

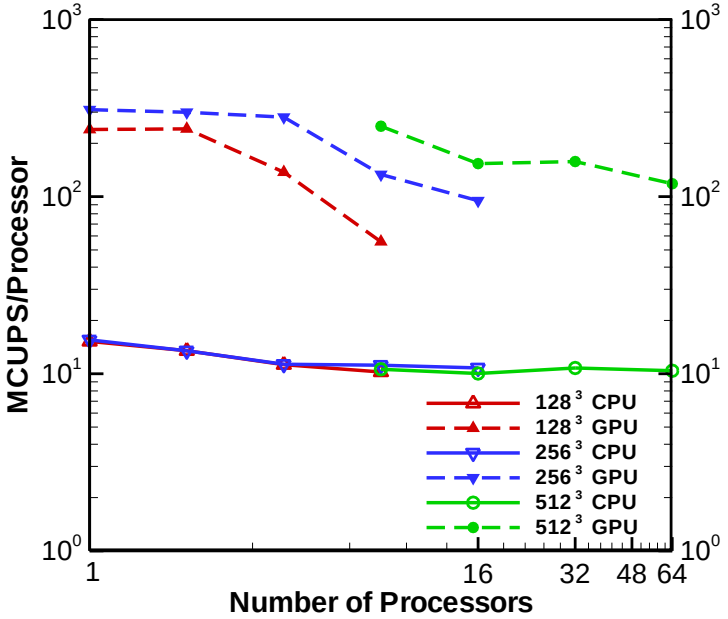
- UP **40x** speedup (32 GPUs Vs. 32 CPU cores)
- Simulating **1024^3** case with 64 GPUs (256^3 per GPU)

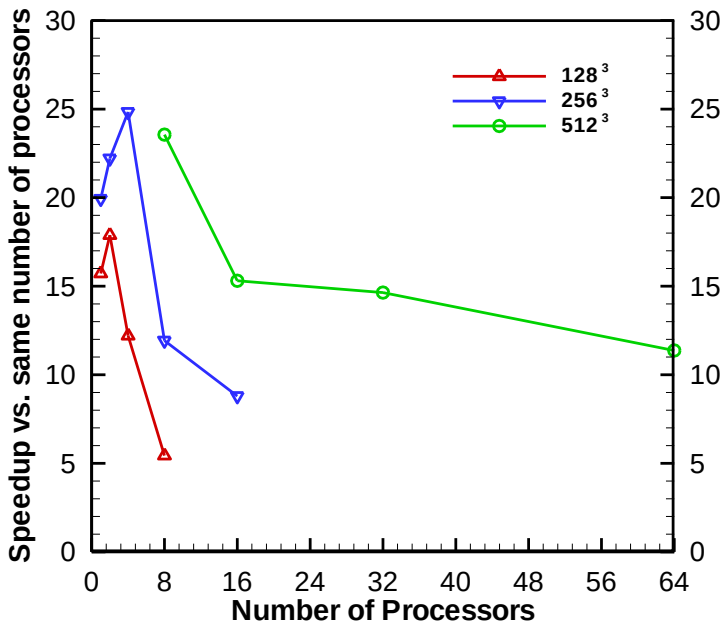




Forge (Strong Scaling)

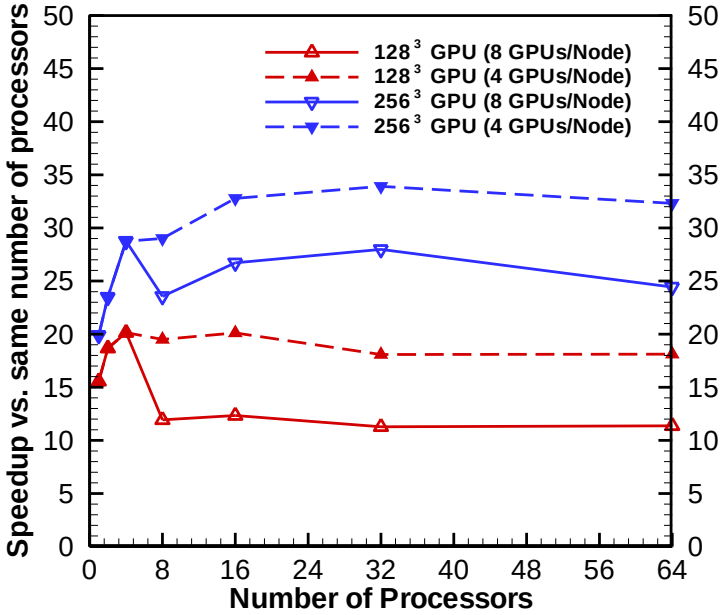
- UP **25x** speedup (4 GPUs Vs. 4 CPU cores)
- Perfect Performance up to 4 GPUs per node

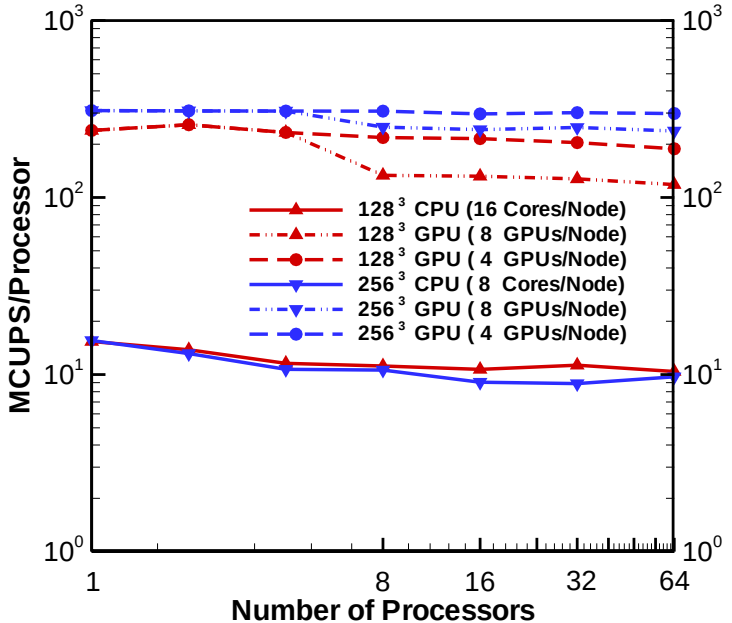




Forge (Weak Scaling)

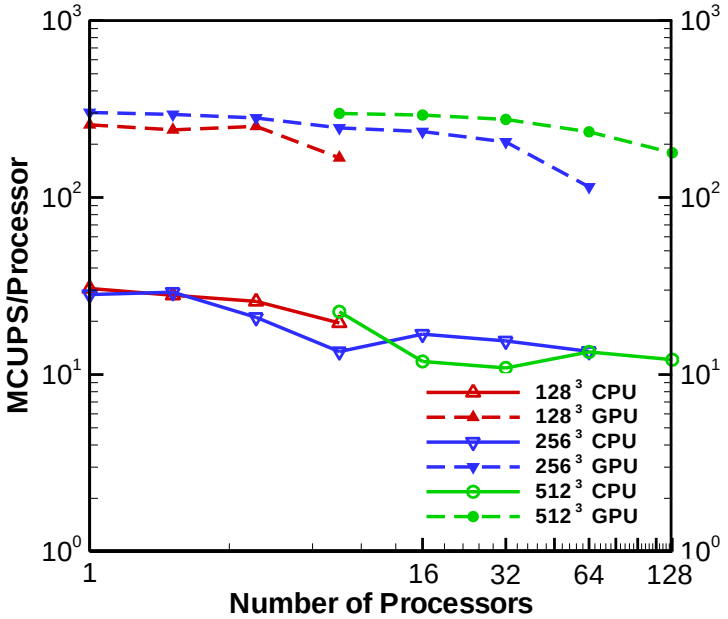
- UP **35x** speedup (32 GPUs Vs. 32 CPU cores)
- Simulating 1024^3 case with 64 GPUs (256^3 per GPU)
- Perfect Performance up to 64 GPUs (4 GPUs per node)

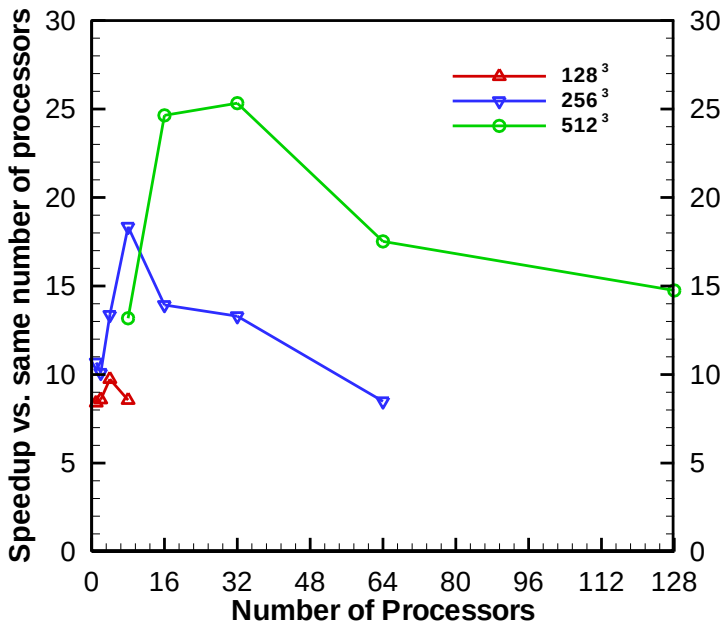




Keeneland (Strong Scaling)

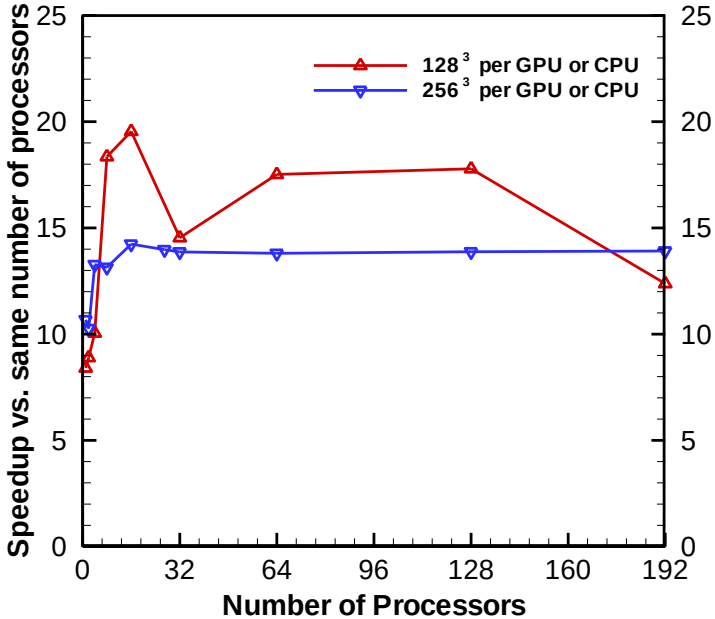
- Speedup and Performance per processor for Strong scaling of the 128^3 and 256^3 CFD problem on Keeneland using GPUs and CPUs
- UP **25x** speedup (32 GPUs Vs. 32 CPU cores)

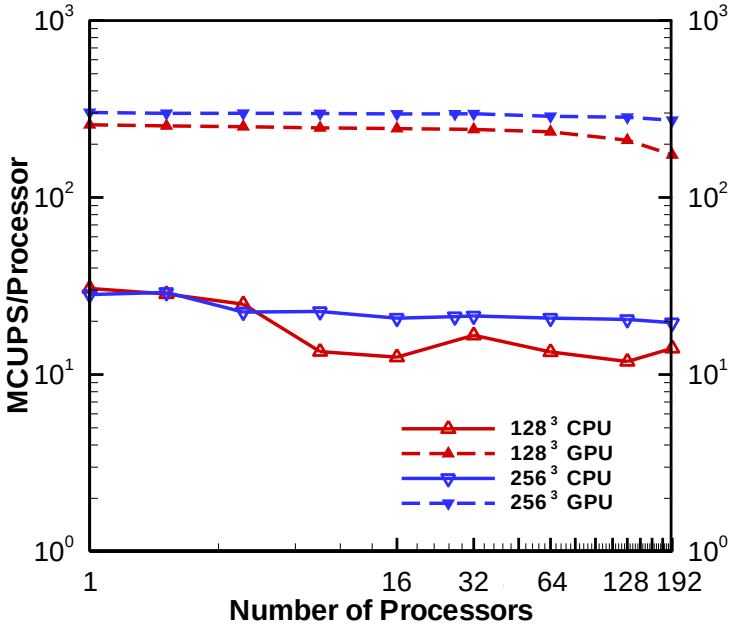




Keeneland (Weak Scaling)

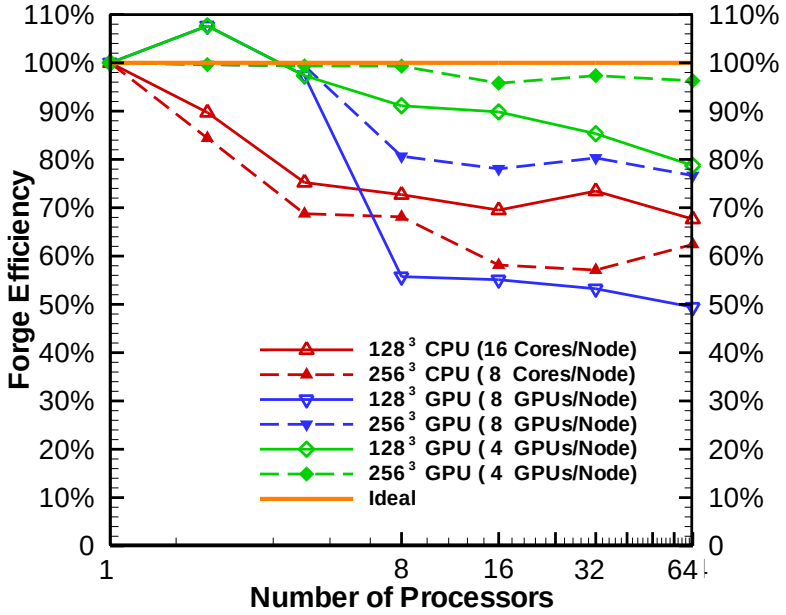
- UP 20x speedup (16 GPUs Vs. 16 CPU cores)
- Simulating 1024x1536x2048 case with 192 GPUs (256^3 per GPU)
- Perfect Performance up to 192 GPUs

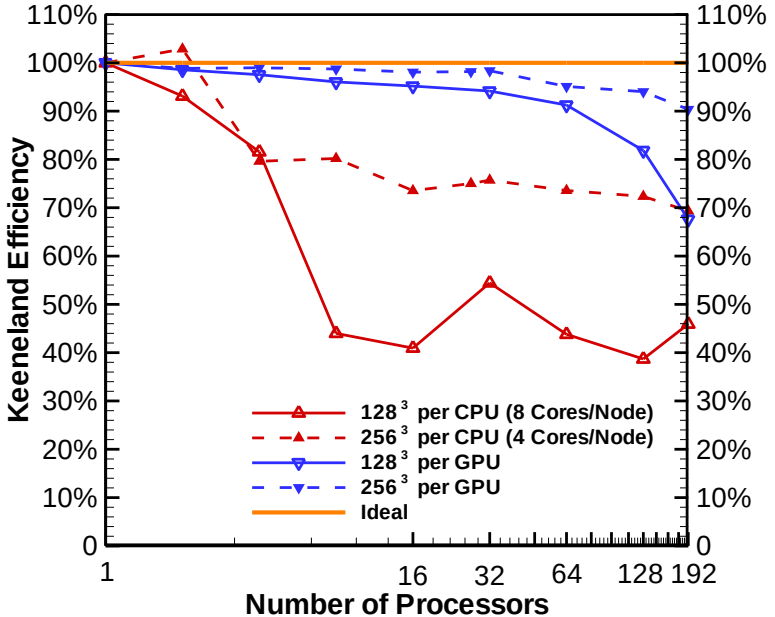




Efficiency

- 96% Efficiency up to 64 GPUs on Forge and Keeneland supercomputers
- 90% Efficiency up to 192 GPUs on Keeneland
- 70% Efficiency up to 64 and 192 CPUs on Forge and Keeneland





Summary

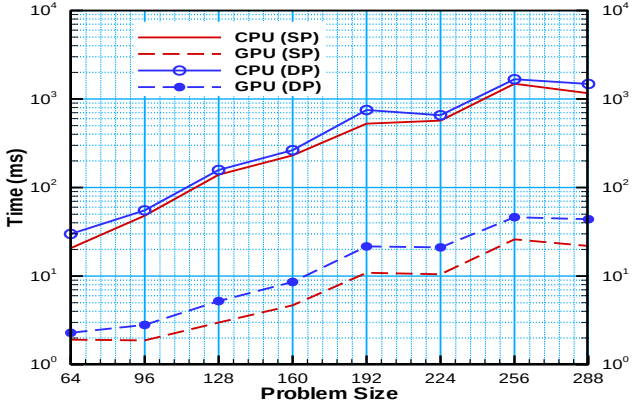
- Unlike the CPU, GPU is well suited for large problems
- PCI-e speed controls the performance for large scientific problems.
- Internal calculations are now so efficient that the operations related to MPI communication are the primary scaling bottleneck.
- It was determined that GPU synchronization calls can have a profound effect on the GPU performance.
- It was determined that GPUs can significantly enhance the speed of CFD calculations (by roughly a factor of 20-30x)

Question ?

References

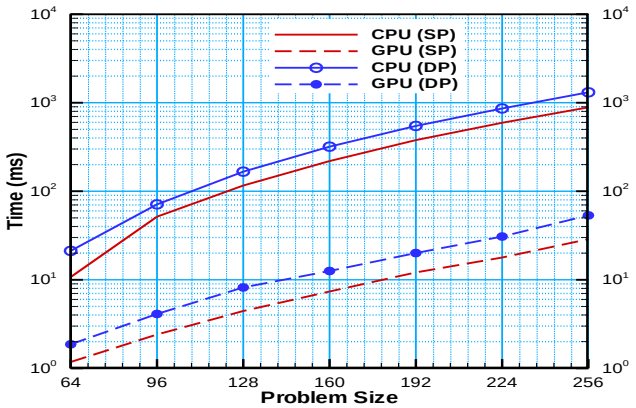
1. Ali Khajeh-Saeed and J. Blair Perot, GPU-Supercomputer Acceleration of Pattern Matching, GPU Computing Gems Emerald Edition, Chapter 13, January 2011, 185-198.
2. Ali Khajeh-Saeed, Stephen Poole and J. Blair Perot, Acceleration of the Smith-Waterman algorithm using single and multiple graphics processors, Journal of Computational Physics 229 (2010) 4247-4258.
3. Ali Khajeh-Saeed and J. Blair Perot, Computational Fluid Dynamics Simulations using many Graphics Processors, Submitted to the Computing in Science and Engineering - Special Issue on scientific computing with GPUs, August 2011.
4. Timothy McGuiness, Ali Khajeh-Saeed, Stephen Poole and J. Blair Perot, High Performance Computing on GPU Clusters, Submitted to the SIAM Journal on Scientific Computing, October 2010.
5. Ali Khajeh-Saeed and J. Blair Perot, Turbulence Simulation using many Graphics Processors, Submitted to the 64th Annual Meeting of the APS Division of Fluid Dynamics, November 20-22 , 2011, Baltimore, MD.
6. T. McGuiness and J.B. Perot, Parallel Graph Analysis and Adaptive Meshing using Graphics Processing Units, 2010 Meeting of the Canadian CFD Society, London, Ontario, 2010.
7. S. Menon, J. B. Perot, “Implementation of an efficient conjugate gradient algorithm for Poisson solutions on graphics processors,” Proceedings of the 2007 Meeting of the Canadian CFD Society, Canada, (2007).

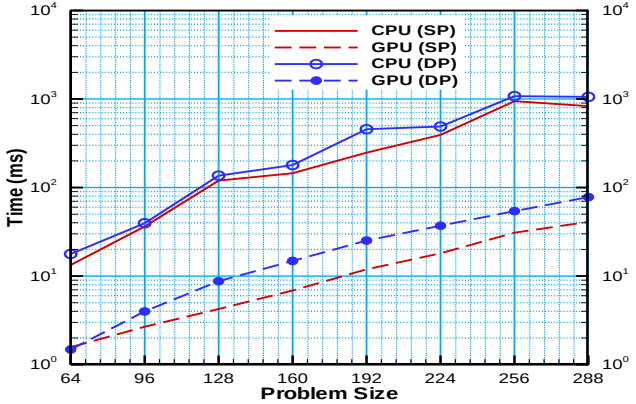
Single CPU and GPU Results



← Orion

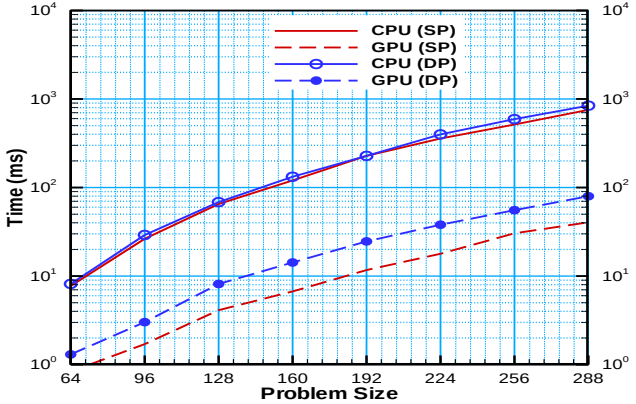
Lincoln →



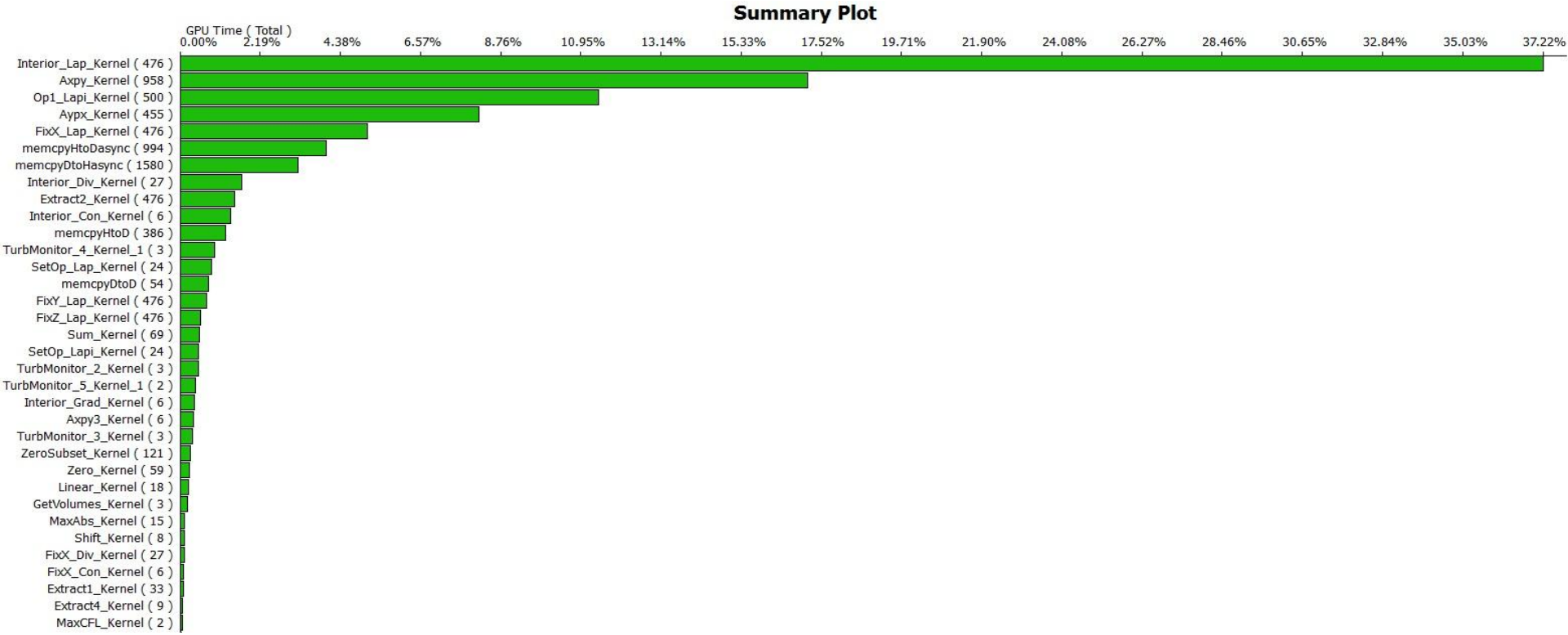


← Forge

Keeneland →



Stag++


Summary Plot
GPU Time (Total)
0.00%
2.19%
4.38%
6.57%
8.76%
10.95%
13.14%
15.33%
17.52%
19.71%
21.90%
24.08%
26.27%
28.46%
30.65%
32.84%
35.03%
37.22%
Interior_Lap_Kernel (476)
Axpy_Kernel (958)
Op1_Lapi_Kernel (500)
Aypx_Kernel (455)
FixX_Lap_Kernel (476)
memcpyHtoDasync (994)
memcpyDtoHasync (1580)
Interior_Div_Kernel (27)
Extract2_Kernel (476)
Interior_Con_Kernel (6)
memcpyHtoD (386)
TurbMonitor_4_Kernel_1 (3)
SetOp_Lap_Kernel (24)
memcpyDtoD (54)
FixY_Lap_Kernel (476)
FixZ_Lap_Kernel (476)
Sum_Kernel (69)
SetOp_Lapi_Kernel (24)
TurbMonitor_2_Kernel (3)
TurbMonitor_5_Kernel_1 (2)
Interior_Grad_Kernel (6)
Axpy3_Kernel (6)
TurbMonitor_3_Kernel (3)
ZeroSubset_Kernel (121)
Zero_Kernel (59)
Linear_Kernel (18)
GetVolumes_Kernel (3)
MaxAbs_Kernel (15)
Shift_Kernel (8)
FixX_Div_Kernel (27)
FixX_Con_Kernel (6)
Extract1_Kernel (33)
Extract4_Kernel (9)
MaxCFL_Kernel (2)

Stag++ (128^3)

Method	# Calls	GPU (ms)	% GPU Time
Interior_Lap_Kernel	**476**	**1062.38**	**37.22**
Axpy_Kernel	**958**	**488.144**	**17.1**
Op1_Lapi_Kernel	**500**	**325.253**	**11.39**
Aypx_Kernel	**455**	**232.61**	**8.14**
FixX_Lap_Kernel	**476**	**145.558**	**5.09**
memcpyHtoDasync	994	113.355	3.97
memcpyDtoHasync	1580	91.6147	3.2
Interior_Div_Kernel	27	47.3499	1.65
Extract2_Kernel	**476**	**42.1359**	**1.47**
Interior_Con_Kernel	6	38.4573	1.34
memcpyHtoD	386	34.9347	1.22
TurbMonitor_4_Kernel_1	3	26.1639	0.91
SetOp_Lap_Kernel	24	23.8348	0.83
memcpyDtoD	54	21.6957	0.76
FixY_Lap_Kernel	**476**	**20.1434**	**0.7**
FixZ_Lap_Kernel	**476**	**15.4049**	**0.53**
Sum_Kernel	69	14.3039	0.5
SetOp_Lapi_Kernel	24	13.5273	0.47
TurbMonitor_2_Kernel	3	13.3754	0.46
TurbMonitor_5_Kernel_1	2	11.5356	0.4
Interior_Grad_Kernel	6	10.2606	0.35
Axpy3_Kernel	6	10.0193	0.35
TurbMonitor_3_Kernel	3	9.32128	0.32
ZeroSubset_Kernel	121	7.23562	0.25

Method	# Calls	GPU (ms)	% GPU Time
Zero_Kernel	59	6.37715	0.22
Linear_Kernel	18	5.97683	0.2
GetVolumes_Kernel	3	5.43398	0.19
MaxAbs_Kernel	15	3.06858	0.1
Shift_Kernel	8	2.62534	0.09
FixX_Div_Kernel	27	2.42592	0.08
FixX_Con_Kernel	6	1.86371	0.06
Extract1_Kernel	33	1.57504	0.05
Extract4_Kernel	9	1.388	0.04
MaxCFL_Kernel	2	1.30173	0.04
TurbMonitor_4_Kernel_2	3	0.698208	0.02
TurbMonitor_5_Kernel_2	2	0.570176	0.01
FixX_Grad_Kernel	6	0.516256	0.01
FixY_Div_Kernel	27	0.394432	0.01
FixZ_Div_Kernel	27	0.34624	0.01
FixY_Con_Kernel	6	0.223872	0
FixZ_Con_Kernel	6	0.210336	0
AxpySubset_Kernel	18	0.17616	0
ShiftSubset_Kernel	24	0.15712	0
FixEdges_Con_Kernel	6	0.082944	0
SetBCval_Kernel	4	0.063008	0
FixY_Grad_Kernel	6	0.05456	0
FixZ_Grad_Kernel	6	0.051456	0
memset32_aligned1D	1	0.003168	0

Speedup

Speedup for Different Memory Type		Send Buffer			
		Synchronized	Mapped	Pinned	Write-Combined
Receive Buffer	Synchronized	21.2	26.3	25.6	10.7
	Mapped	23.6	**30.4**	27.4	10.8
	Pinned	24.1	29.2	28.3	11.2
	Write-Combined	24.2	29.3	28.1	10.3