Rapid Aerodynamic Performance Prediction on a Cluster of Graphics Processing Units

Everett Phillips
Yao Zhang
Roger Davis
John Owens

47th AIAA Aerospace Sciences Meeting
Jan. 5-8, 2009
Orlando, Florida

Overview

- Background/Motivation
- Advantages of Graphical Processing Units (GPU)
- Today's GPU Hardware Capability
- Programming on the GPU
- Current GPU solver implimentations
- Results of Current Benchmarks
- Summary

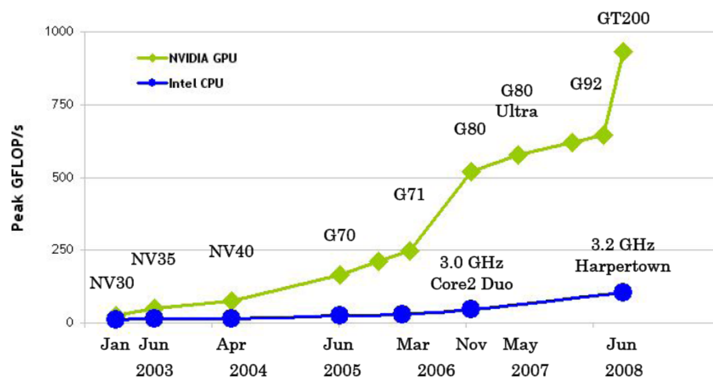
Background/Motivation

- Graphical processing units (GPUs) have proven success for gaming applications
- Recently shown to also be useful for scientific simulations
- Current investigation focuses on demonstrating:
  - Optimal performance gains using GPUs
  - GPU performance gains for existing “general purpose” codes typical of those used in government and industry

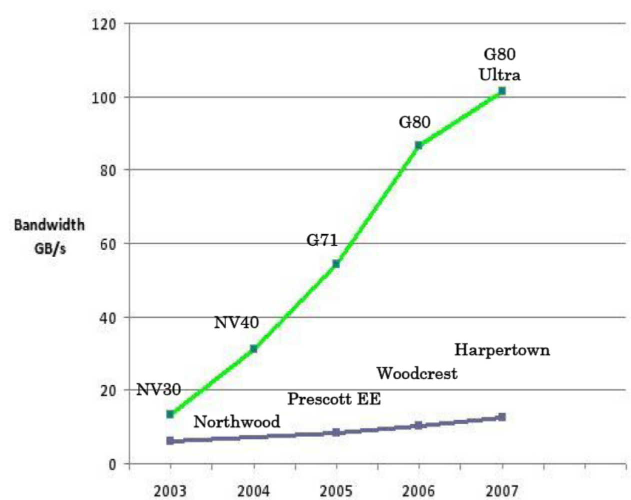
Advantages of Graphics Processors

- Order of magnitude increase in
  - floating point
  - memory bandwidth
- Very low cost
- Easy to program with new programming models (CUDA)
- Good at processing large data sets where same operation is applied over large arrays
- Scales well when added to cluster nodes
- Perfect fit for CFD applications

GPU vs. CPU Performance Trends



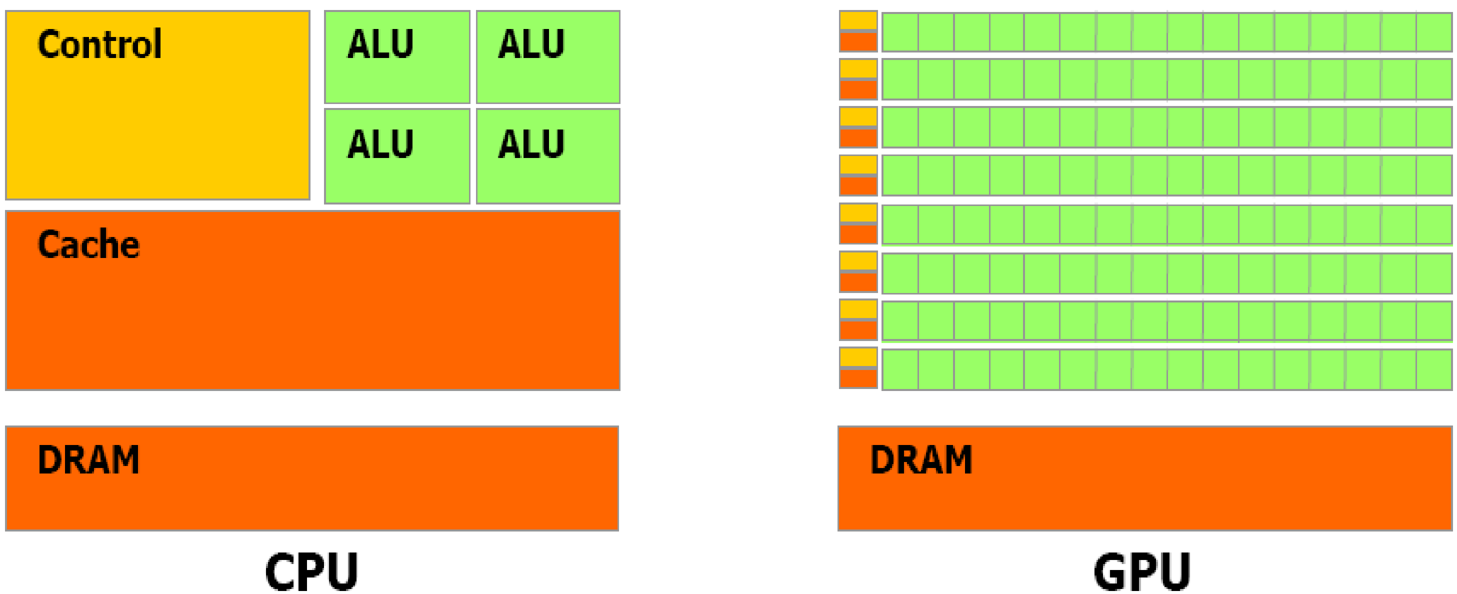
GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	



Figures courtesy NVIDIA

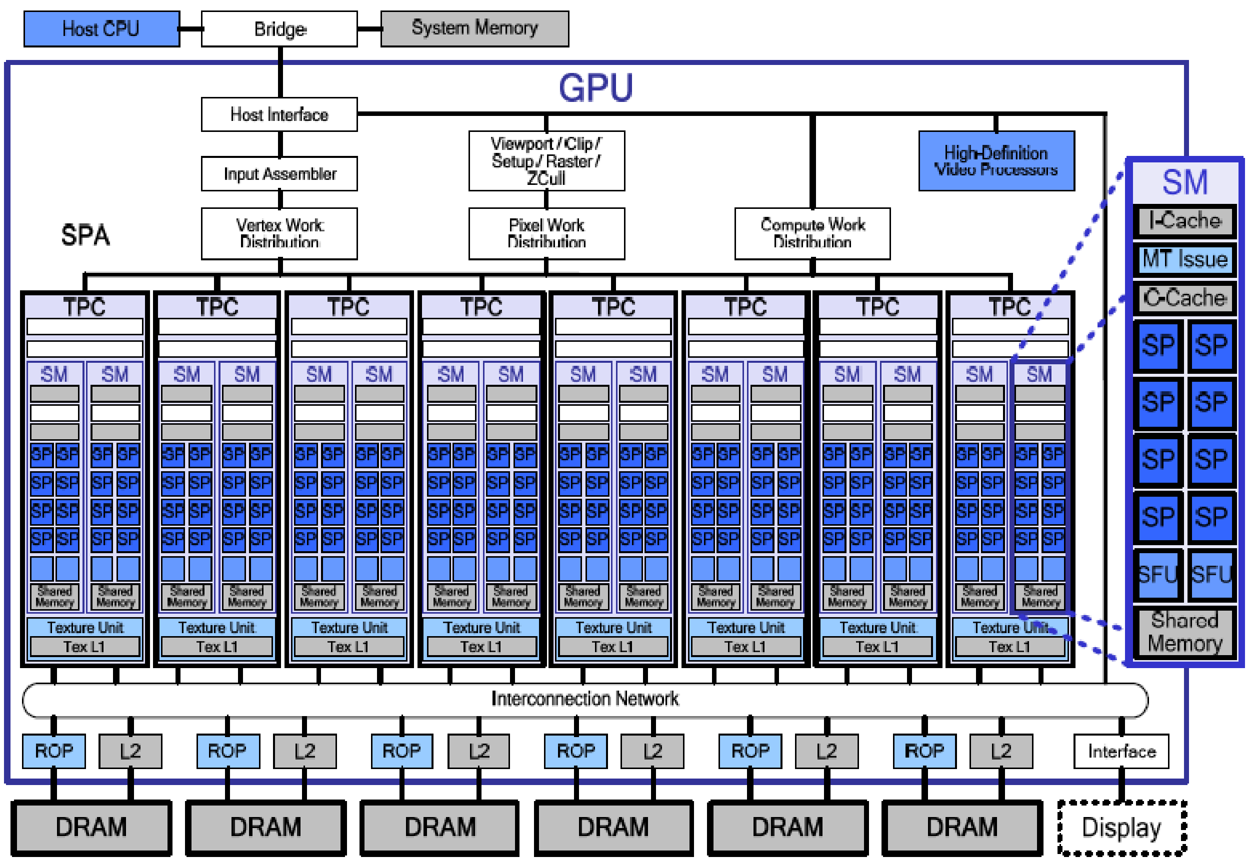
GPU Architecture

- More transistors devoted to data processing (shown in green)
- Optimized for throughput
- Data Parallel (SPMD)



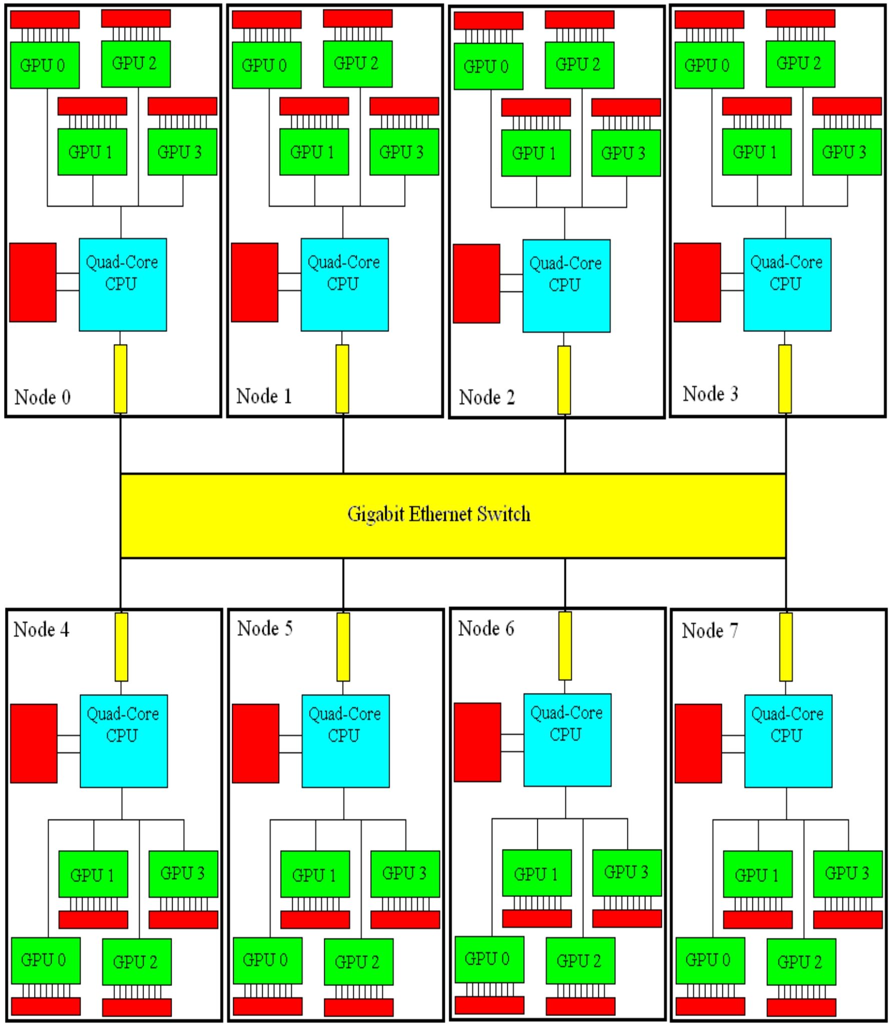
Figures courtesy NVIDIA

GPU Architecture : G80



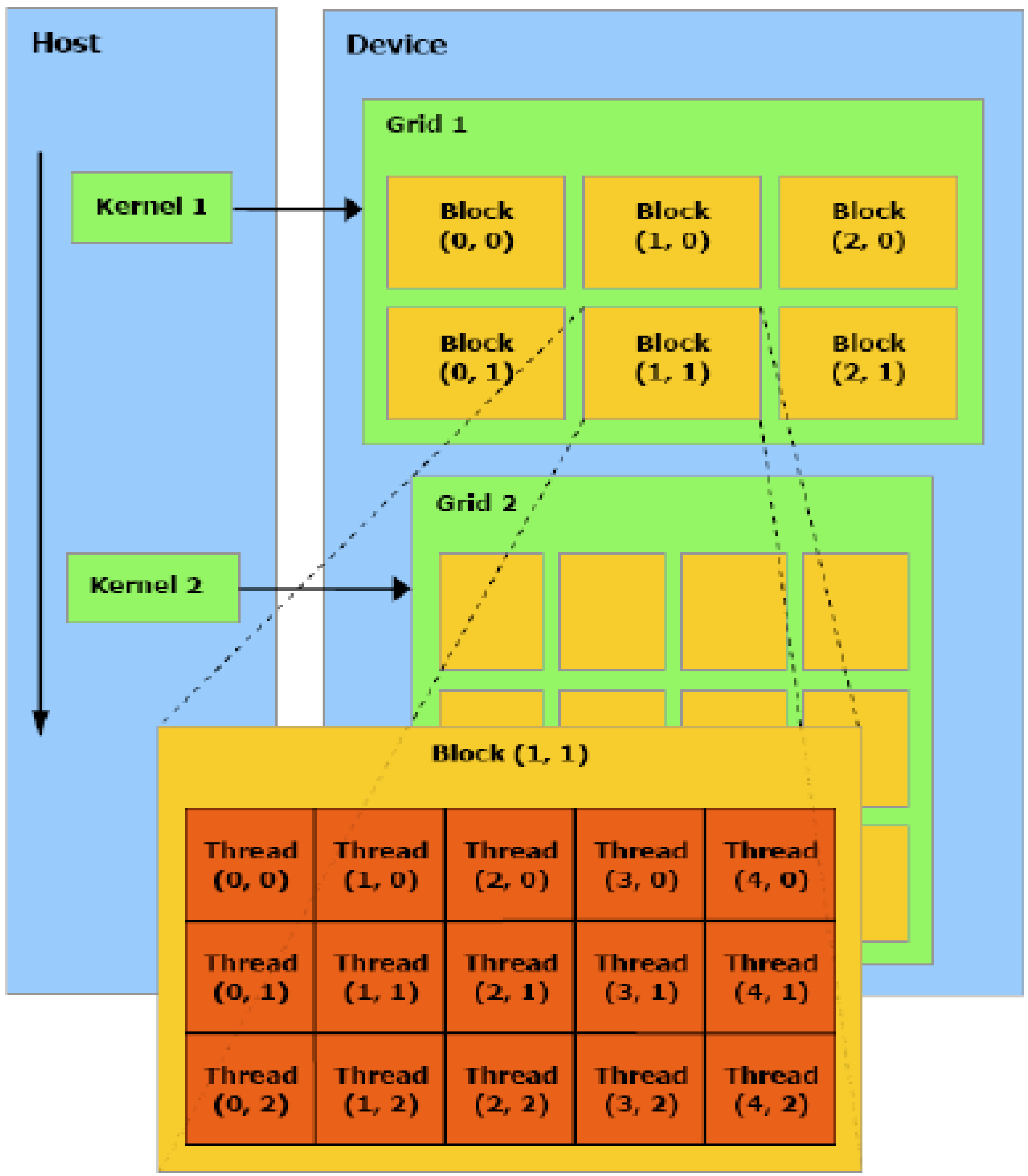
Figures courtesy NVIDIA

Cluster Hardware



- Running Rocks Linux for Clusters OS
- 8 Nodes, each including:
  - 2.5 GHz quad-core CPU with 6 Mb cache
  - 8 Gigs DDR3 memory
  - 4 GPUs w/128 floating point units each
- 32 GPU/32 CPU cores
- Over 12 Teraflops
- Cost: $25,000
- Equivalent CPU cluster:
  - 500 nodes and ~$1,000,000

NVIDIA CUDA Program Structure



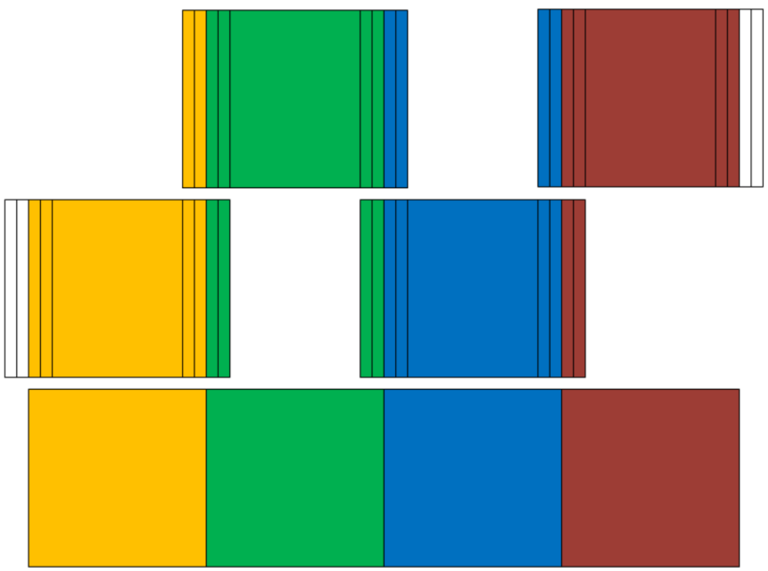
Figures courtesy NVIDIA

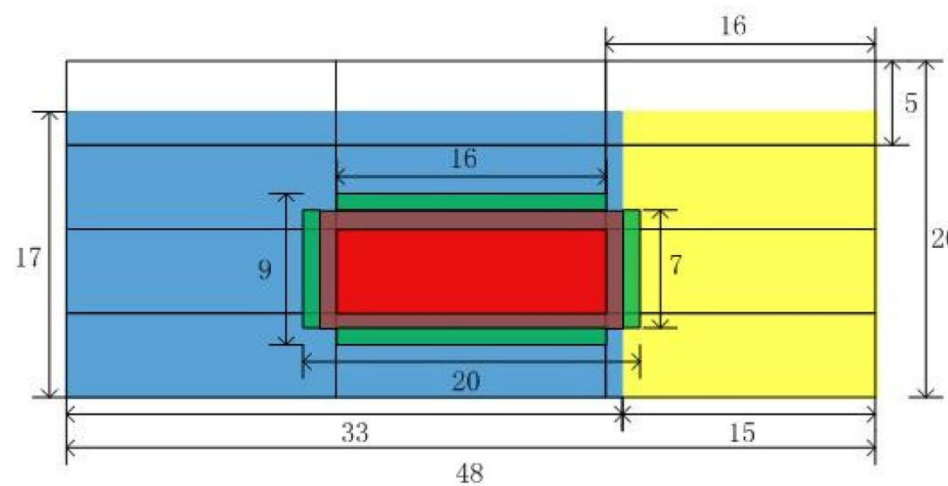
Current Investigation

- Application of GPUs to Computational Fluid Dynamics (CFD)
- Determine optimal performance gains using Euler code constructed specifically for GPU
- Determine "typical" performance gains for existing "general purpose" CFD codes
  - Use MBFLO multi-block, structured-grid Navier-Stokes code
    - Arbitrary block connectivity and orientation
    - Several turbulence modeling strategies including 2-equation RANS, DES, and hybrid RANS/LES

Decomposition

- Current GPU implementation uses
  - 1D decomposition (stripes)
  - 2 layers of ghost nodes/cells
- General decomposition using distributive operator underway





MBFLO Subroutines

CPU Code

```
subroutine lamvis

do j = 1,jmax(n)
  do i = 1,imax(n)
    tott = (gama – 1.0)*(u6 – 0.5*(u7**2 u8**2)
    xmu(1,i,j,n) = xmufree*(tott**1.5d0)/(tott + suthcnst)
  enddo
enddo
```

GPU Code

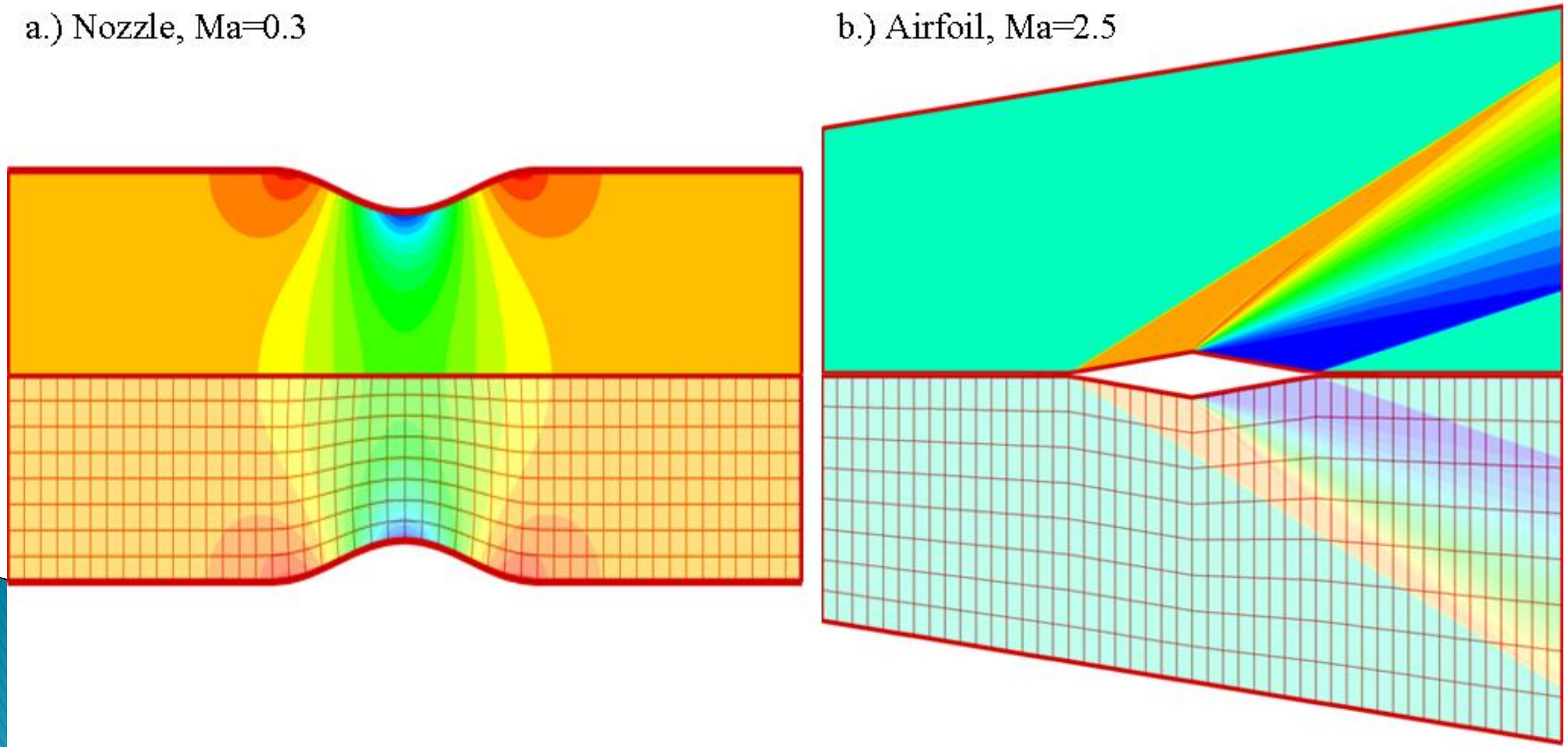
```
__global__ void lamvis_kernel( ... )
{
unsigned int i = threadIdx.x + (blockDim.x)*blockIdx.x;
unsigned int j = threadIdx.y + (blockDim.y)*blockIdx.y;
unsigned int index  = i + j*(imax);
float tott,u6,u7,u8,output;
  if(i<imax && j<jmax)
  {
    tott       = (gama-1.0f)*(u6 – 0.5f*(u7*u7+u8*u8) )/(rttovfree*gama);
    xmu[index] = xmufree*powf(tott,1.5f)/(tott + suthcnst);
  }
}
extern "C" void gpu_lamvis_( ... )
{
dim3 dimBlock(16, 4, 1);
dim3 dimGrid ((imax+dimBlock.x-1)/(dimBlock.x), (jmax+dimBlock.y-
1)/(dimBlock.y));
lamvis_kernel<<<dimGrid, dimBlock>>>( ... );
}
```

CUDA integration with MBFLO

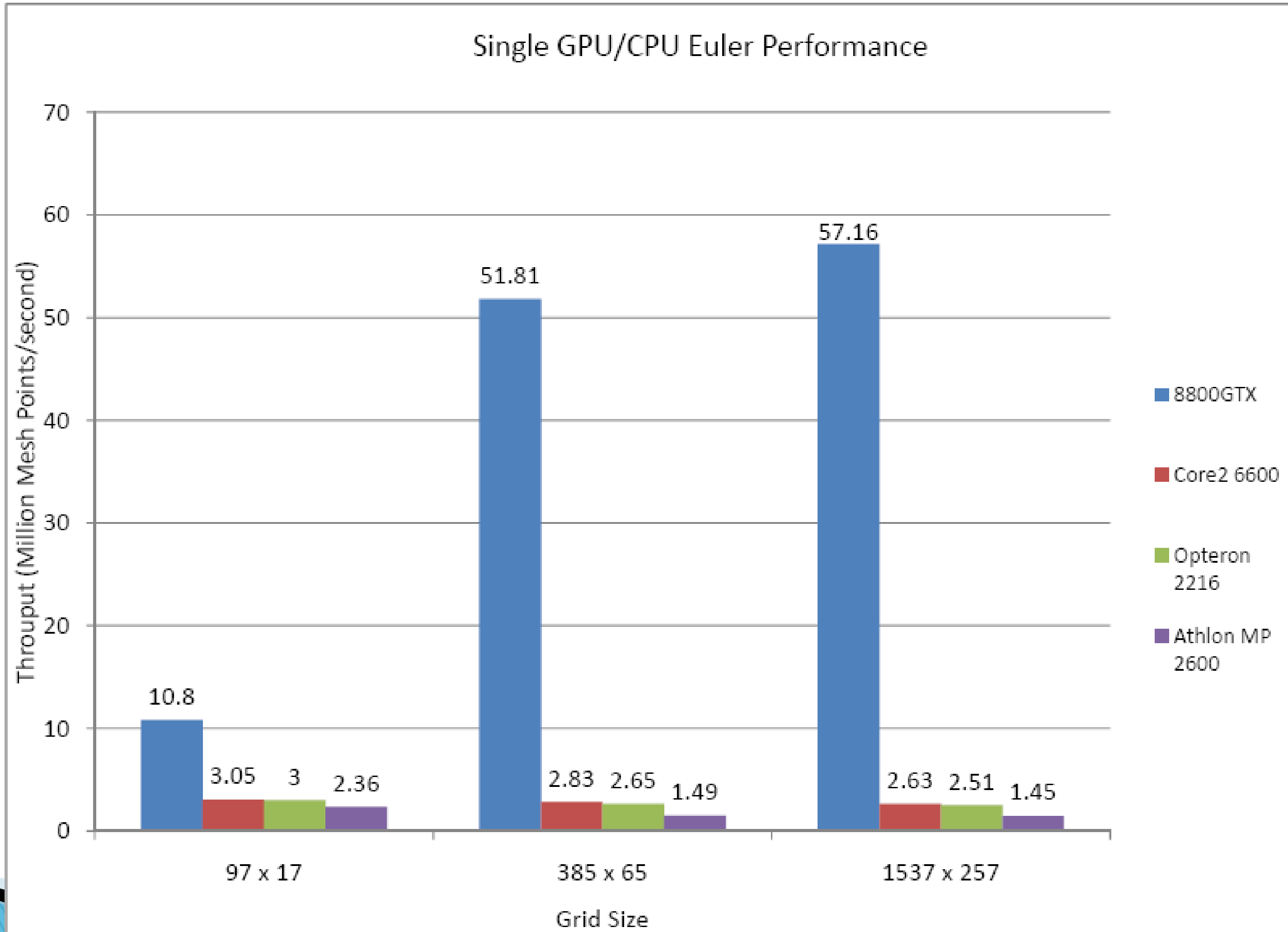
```
if(gpu==1) then
      call gpu_function(...)
else
      call function
endif

if(gpu==1) then
      call gpu_pack_buffer(...)
      call copy_to_host(buffer_d, buffer)
      call blkbnd
      call copy_to_gpu(buffer_d, buffer)
      call gpu_unpack_buffer(...)
else
      call blkbnd
endif
```

Euler Results

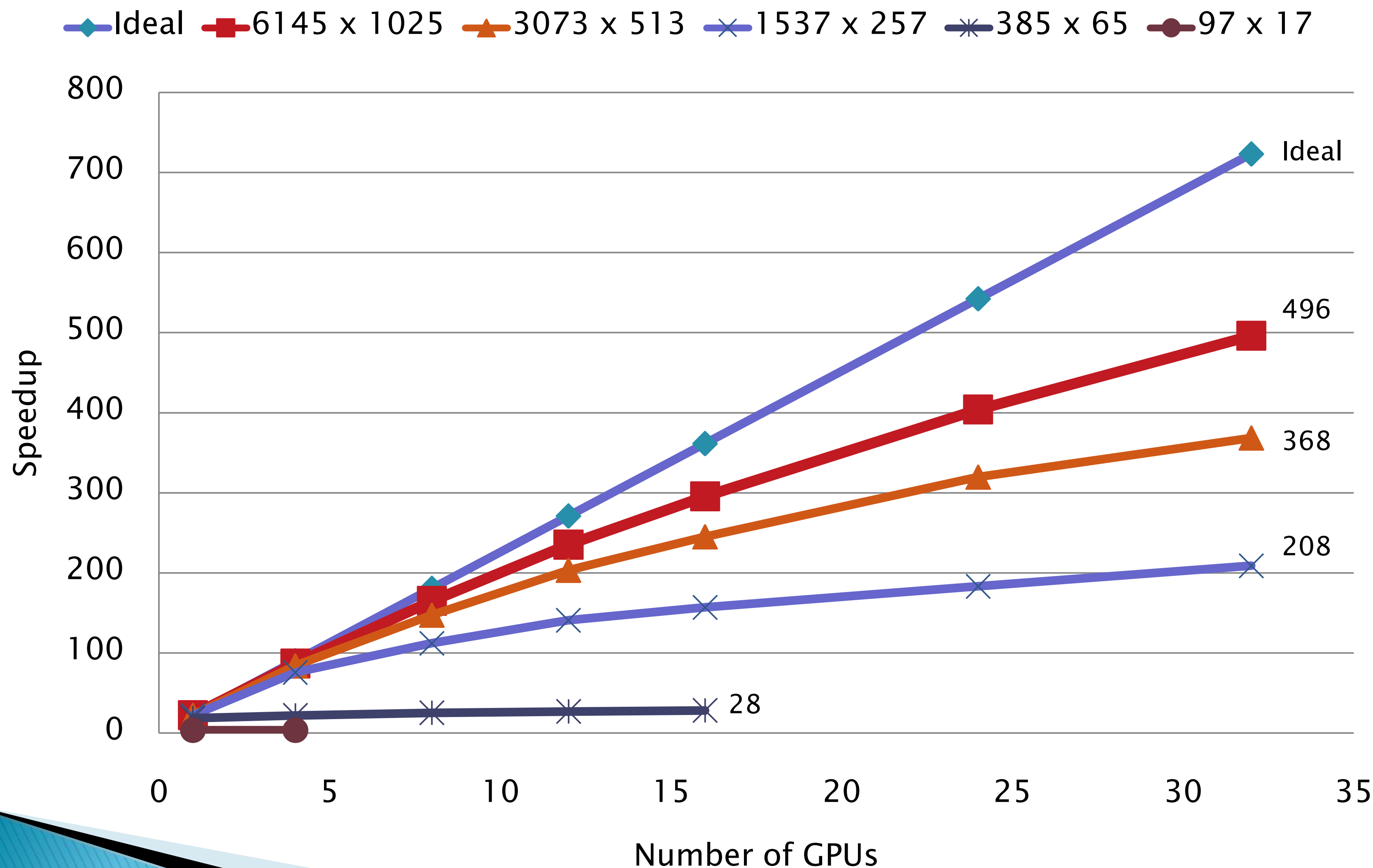
- Subsonic nozzle and supersonic diamond airfoil
  - Grids up to 6.4M points



Single GPU Euler Solver Performance

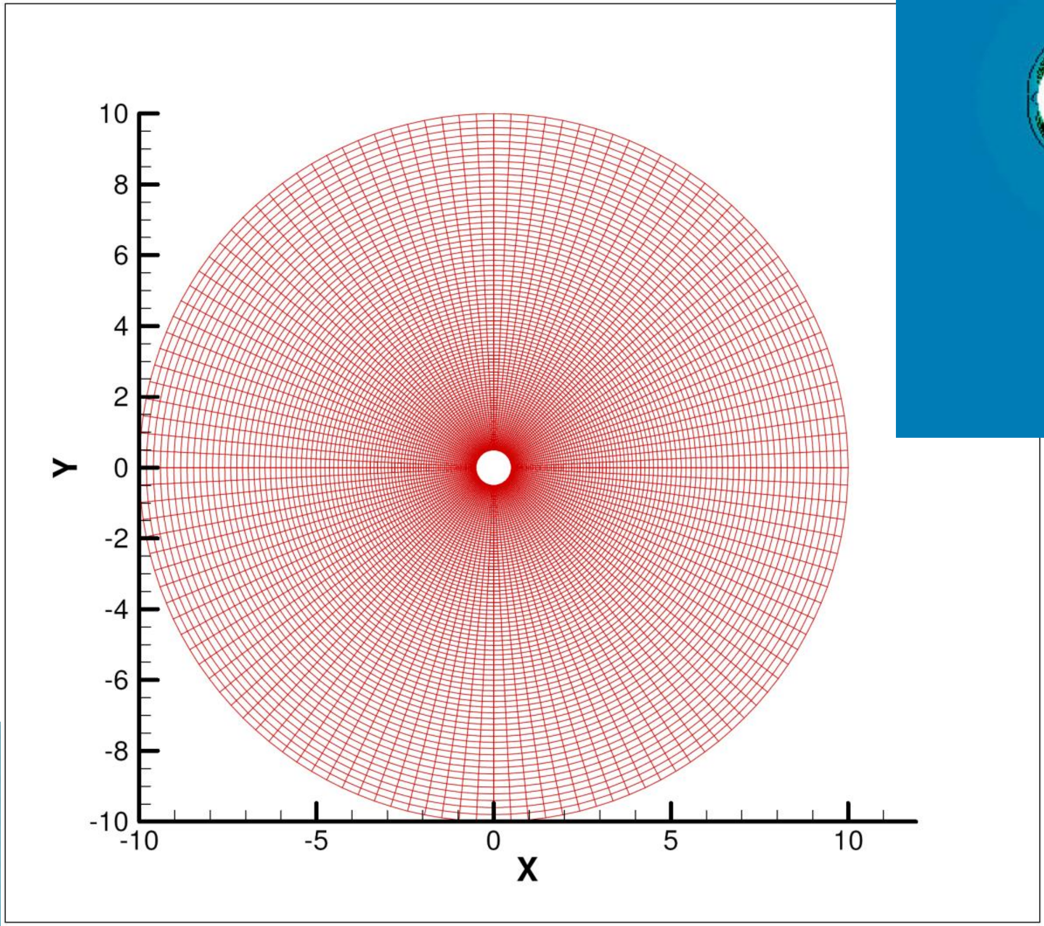


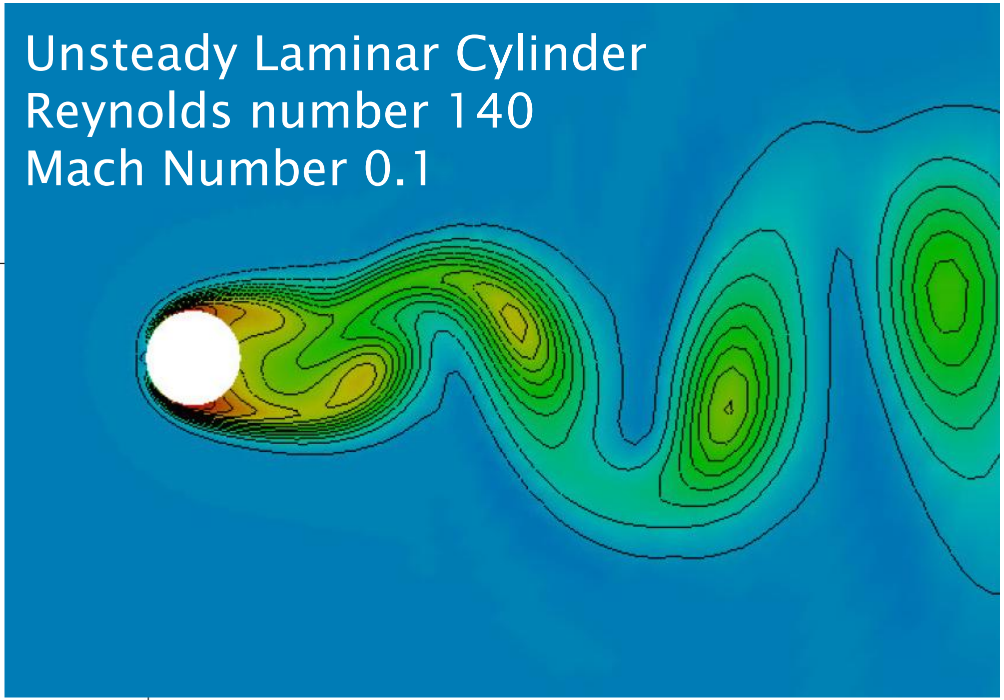
Parallel Euler Performance



MBFLO Results

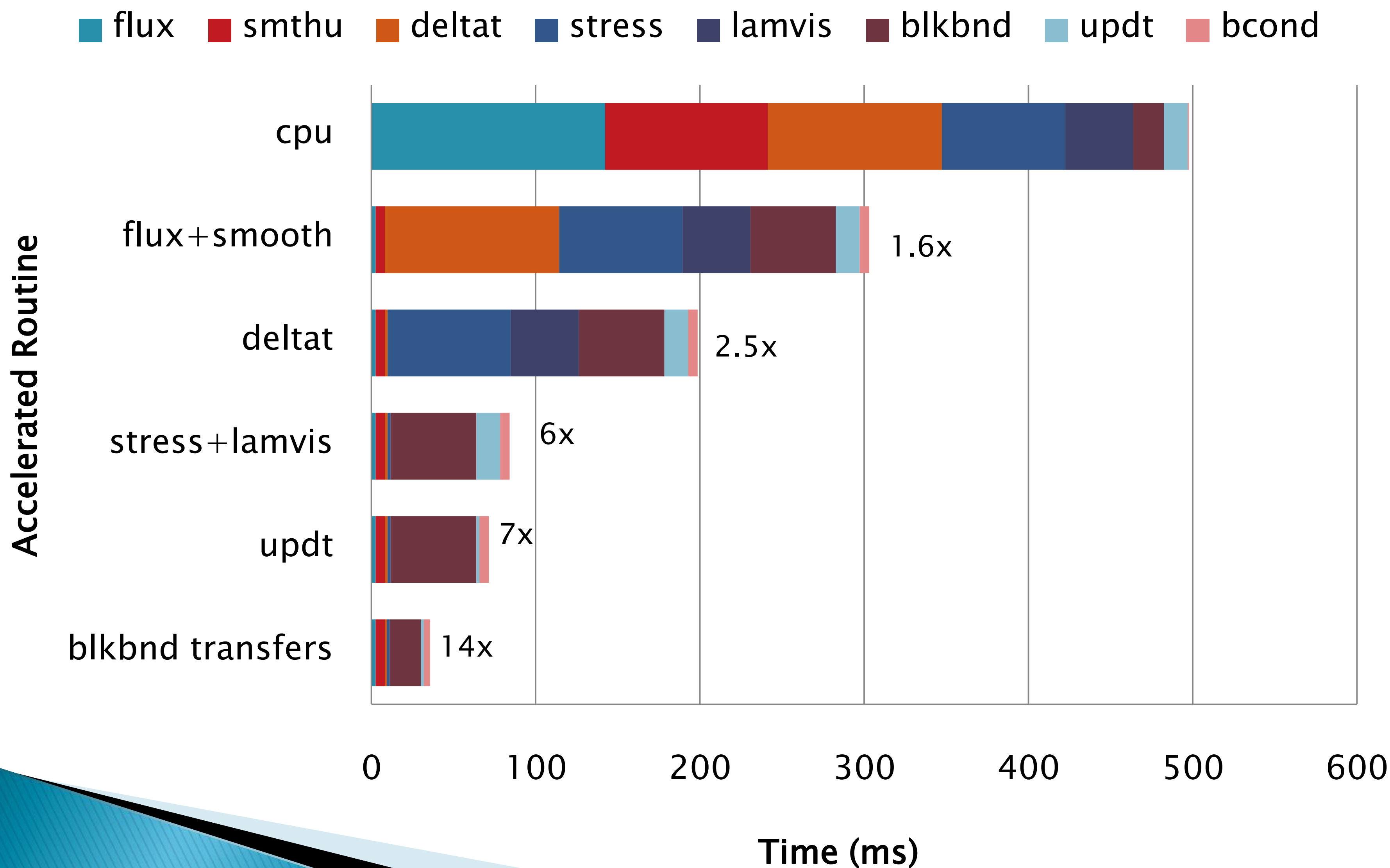
Up to 16 Blocks in
Computational Grid



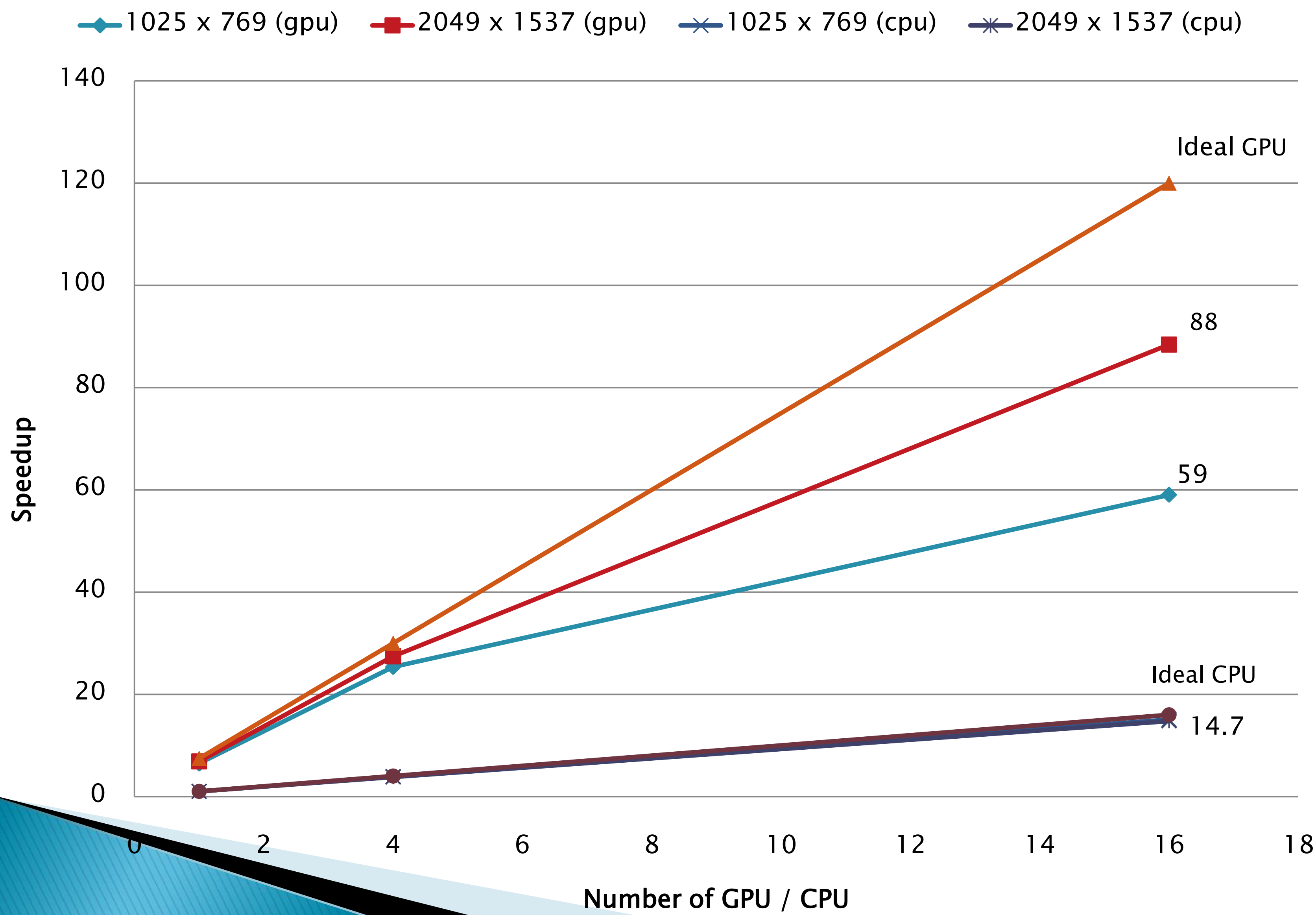


Entropy Contours

Single GPU MBFLO Performance



Parallel MBFLO Performance

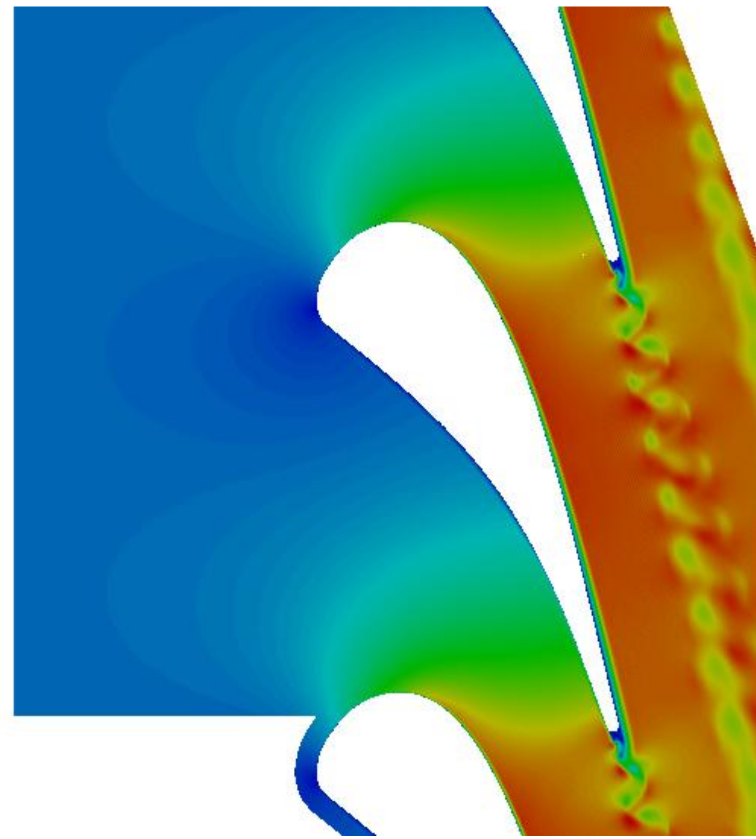


Summary

- The GPU shows great promise in increasing performance/price ratio by multiple orders in magnitude
- Research underway to demonstrate
  - Ease of use
  - Generality for different algorithms

Future Effort

- Unsteady, turbulent flow
- Detached-eddy or hybrid RANS/LES turbulence modeling
- Goal for unsteady and time-averaging:
  - 2D: under 30 seconds
  - 3D: under 1 hour



Future Research Directions

- Creation of GPU library with multilevel primitives
  - Low-level (kernels: face-flux, stress, etc.)
  - Medium-level (routines: flux, smoothing, etc)
  - High-level (algorithms: slor, adi, etc.)
- Adaptive Mesh Refinement with GPUs

Acknowledgements

- Thanks to Department of Energy's Early Career Principal Investigator Award (DE-FG02-04ER25609)
- The authors would also like to thank the managers of Wright-Patterson Air Force Research Laboratory for partial support of this effort under contract 03-S530-0002-11-C1 with Dr. John Clark as technical monitor
- NVIDIA for hardware donations