LARGE SCALE COMPUTATIONAL FLUID DYNAMICS SIMULATIONS ON HYBRID SUPERCOMPUTERS

Eric Kelmelis
EM Photonics

May 16, 2012
Modernization of CFD Solvers

- CFD solvers are used throughout the DoD, NASA, and elsewhere
  - New solvers are being developed
- Real world CFD simulations require massive computational power
- HPC hardware technology has recently evolved much faster than software implementations
  - Current solvers do not scale to new hardware

**Case Study**

- NAVAIR uses CFD solvers to model ships and their interaction with moving planes
- Full ships require >500,000 CPU hours on a supercomputer
- More performance means faster turnaround times for simulations and more complex models
PROJECT OVERVIEW

» We are experts in high-performance computing and accelerated processing

» Experience in scientific computing in general and CFD specifically

» **Project Goal:** Improve performance of CFD solvers on massive HPC systems
<table>
<thead>
<tr>
<th>CFD Solvers Addressed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AVUS</strong></td>
</tr>
<tr>
<td><strong>Created by:</strong> AFRL</td>
</tr>
<tr>
<td>» 50k lines of Fortran</td>
</tr>
<tr>
<td>» Can be coupled to</td>
</tr>
<tr>
<td>external codes</td>
</tr>
<tr>
<td><strong>Used by:</strong> Primarily DoD</td>
</tr>
<tr>
<td><strong>FUN3D</strong></td>
</tr>
<tr>
<td><strong>Created by:</strong> NASA Langley</td>
</tr>
<tr>
<td>» 765k lines of Fortran</td>
</tr>
<tr>
<td>» Multiphysics</td>
</tr>
<tr>
<td><strong>Used by:</strong> NASA,</td>
</tr>
<tr>
<td>commercial, some DoD</td>
</tr>
</tbody>
</table>

» Unstructured grid solvers
» “MPI Everywhere” structure
» Desire to scale to massive problems (10,000+ cores)
» Include many legacy manual optimizations
NAVAIR uses:

- FUN3D
- AVUS
- USM3D and Cobalt also popular
- Several other solvers and tools used too
The three major chip makers are all following the heterogeneous “accelerator model”

- Develop solutions that use entire system (multicore + accelerators) and scale as new hardware is added
- Communication is more complex and critical to performance and scaling

<table>
<thead>
<tr>
<th>NVIDIA</th>
<th>Intel</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>![NVIDIA Logo]</td>
<td>![Intel Logo]</td>
<td>![AMD Logo]</td>
</tr>
<tr>
<td>![Tesla Logo]</td>
<td>![MIC Logo]</td>
<td>![OpenCL Logo]</td>
</tr>
<tr>
<td>Heavily backing their mature CUDA technologies</td>
<td>Developing “Knights Corner” in addition to multicore processors</td>
<td>Supports the OpenCL standard for their CPUs &amp; graphics cards</td>
</tr>
</tbody>
</table>
High-performance computing nodes have evolved
Nodes contained single-core CPUs and were connected by MPI.
» Jobs equally divided between identical nodes
» Larger problems can be solved
» Processing time is reduced

**MPI is designed for inter-node communication**
CURRENT SOFTWARE ON MODERN HPC NODES

» MPI software can be run on multicore chips without modification so it often is
» MPI software cannot take advantage of vector processors (GPUs)

What is needed
» Extend software to take advantage of GPUs
» Optimize intra-chip communication of multicore chips
  » MPI not the most efficient way for scaling
» Continue to use MPI for inter-node communication
» Performed profiling using NAVAIR inputs
» Observed similar results to NASA
Profiling AVUS

» Air Force CFD solver
  » AFRL/VAAC
» Profiling shows over 60% of compute time is in one routine
  » Gauss-Seidel solver
» Similar to NASA FUN3D’s profile
**Initial Prototyping**

### Accelerating Bottlenecks

- Prototyped most computationally intense routine on GPU
- Work distribution became dramatically unbalanced – no overall improvement in speed

### A New Approach

- Initial prototypes showed the need to escape intra-chip MPI parallelism and static work distribution
- Prototyped main computational region
  - 16x speedup that could not be realized
- Completely update AVUS for OpenMP + MPI
» First step is extend the underlying framework to handle work queues

» CPU optimization and GPU programming can happen concurrently
SOFTWARE EVOLUTION STEP 1: WORK QUEUE

**Original AVUS**

1. Job is broken up into pieces and distributed to processors in the system.

**Modified Version**

1. Create a queue of work that can be dynamically distributed to processors.
# Advantages of the Work Queue

<table>
<thead>
<tr>
<th>Current Model</th>
<th>Work Queue Approach</th>
</tr>
</thead>
</table>
| » Slowest worker sets whole system pace  
  » Slower CPU  
  » Larger work unit  
 » Problem partitioning is static  
  » Rebalancing can be added later but not as efficient | » Different processing rates supported  
  » Increase number of work packets – workers receive different amounts  
 » Easily supports dynamic reallocation  
 » Allows for mixed processor speeds |
Each core has its own memory that cannot be shared. Communication with MPI.

Transition to OpenMP and shared memory to improve intra-node efficiency.
ADVANTAGES OF USING ONE PROCESS

» Less communication overhead
» More efficient data transfers through shared memory
» Reduced memory usage
» Allows partitioning work to GPU while still using the CPU
» OpenMP chosen – minimal disruption to existing code
SOFTWARE EVOLUTION STEP 3: ADDING GPUs

» GPU processes portions of the work for critical routines
  » Amount of work taken based on its relative performance

» GPU code impact is small
  » Could change accelerators easily

» OpenMP allows for flexibility
ADVANTAGES OF ADDING GPUs

» GPUs are moving from novelty to mainstream in current generation HPCs
» GPUs have much more computational power and are more computationally efficient than CPUs
» Next-generation hardware from all major vendors will incorporate GPU-like vector processing
CPU Results
» Memory usage decreased by 20%
  » More drastic as number of cores increases (4 core)
» Flexible architecture allows for runtime performance adaptation
» Speed gains will be realized over current MPI-only model

GPU Results
» A 16x and 13x performance increase were realized in two important code sections
» This leads to a 35% reduction in overall runtime
» Further tuning is ongoing

Overall Runtime

Runtime of prototype segments
Benefits of Our Approach

Lower Memory Usage
Shared data is truly shared

Higher Performance
On CPU and GPU

Flexible Architecture
Supports accelerator paradigms

Dynamic Balancing
Respond to conditions on-the-fly

More Scalable
Optimizing communication allows scaling to more nodes
Next Steps

» Remove intra-chip MPI
  » Direct memory transfers
» Implement dynamic load balancing
  » Move zones between threads and nodes
» Move additional code segments to the GPU
  » Only one subroutine thus far
» Transition into Kestrel
CONCLUSION

» HPC model has changed
  » Many CPU cores need to be optimally managed within a node and GPU technology must be leveraged

» Vital to CFD
  » Current CFD solvers are unable to scale up to the number of cores available today

» Applicable Elsewhere
  » Computationally intense software in other fields can benefit
  » We are always looking for new problems to tackle

Questions?
BIG PICTURE

Node

Process

OpenMP

Node

Process

OpenMP

MPI
**Computing Technology Trends**

**Parallel Processing**
To achieve maximum performance, an algorithm’s parallelism must be exploited.

**Multicore**
Parallelism is now available within a single chip, shifting the programming paradigm.

**Heterogeneous Computing**
Systems are being designed to use different types of processing devices at the same time.

**Hybrid Supercomputing**
Heterogeneous paradigm now being scaled to large HPC systems.
NASA’s FUN3D

» Important CFD code for unstructured grids
» Capable of analyzing a broad range of CFD physics
» Used in commercial, defense, and research
» EM Photonics is working closely with NASA engineers