Industrial achievements on Blue Waters using CPUs and GPUs

HPC User Forum, September 17, 2014 – Seattle

Seid Korić PhD
Technical Program Manager
Associate Adjunct Professor

koric@illinois.edu
“It is amazing what one can do these days on a dual-core laptop computer. Nevertheless, the appetite for more speed and memory, if anything is increasing. There always seems to be some calculations that ones wants to do that exceeds available resources. It makes one think that computers have and will always come in one size and one speed: “Too small and too slow”. This will be the case despite supercomputers becoming the size of football fields!”

Prof. Tom Hughes, 2003, President of International Association for Computing Mechanics-IACM
High fidelity virtual engine simulation and design

- > 1 trillion degrees of freedom (DOF)
- > 1 billion core hours per calculation
Private Sector Program at NCSA

Solve industry’s most demanding HPC challenges

Aerospace, agriculture, consumer goods, energy, life sciences, health care, and technology sectors

Largest HPC industry engagement program in the USA (World)
National Petascale Computing Facility

World-Class Datacenter

- Dept. of Energy-like security
- 88,000 sq. feet of space
- 25 MW of power; LEED Gold
- 100 (400) Gb/sec bandwidth

Hosting Benefits to Industry

- Low-cost Power & Cooling
- 24/7/365 Help Desk
- 1 block from University Research Park
NCSA Supercomputers

iForge – a Supercomputer for Industry
100% designed for and dedicated to industry
99% up time
On-demand, reservation, and hosted options
Primary or “burst” HPC capacity
Evergrene-yearly upgrades to stay on cutting edge

Blue Waters – a Football field size Supercomputer
1 petaflop/sec sustained with real applications
400,000 x86 cores, 12 million CUDA cores
1.5 petabytes of RAM
400+ petabytes of storage
<table>
<thead>
<tr>
<th><strong>Node Type</strong></th>
<th><strong>Dell PowerEdge C8000</strong></th>
<th><strong>Intel Reference</strong></th>
<th><strong>Dell PowerEdge C6145</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>Intel &quot;Ivy Bridge&quot; Xeon E5 2680 v2</td>
<td>Intel “Ivy Bridge” Xeon E7 4890 v2</td>
<td>AMD “Abu Dhabi” Opteron 6380</td>
</tr>
<tr>
<td><strong>Total Nodes</strong></td>
<td>144</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td><strong>Total x86 Cores</strong></td>
<td>2,880</td>
<td>120</td>
<td>576</td>
</tr>
<tr>
<td><strong>Cores/Node</strong></td>
<td>20 cores</td>
<td>60</td>
<td>32 cores</td>
</tr>
<tr>
<td><strong>Memory/Node</strong></td>
<td>64 or 256 GB, 1.86 GHz</td>
<td>1.5 TB, 1.6 GHz</td>
<td>256 GB, 1.6 GHz</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>700 TB on network filesystem (IBM GPFS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Interconnect</strong></td>
<td>QDR InfiniBand, 40 Gb/sec, 100 ns latency</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>Red Hat Enterprise Linux 6.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Blue Waters

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Cray XE6</th>
<th>Cray XK7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>2 x AMD “Interlagos” Opteron 6276</td>
<td>1 x AMD “Interlagos” Opteron 6276</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>NA</td>
<td>1 x Nvidia “Kepler” Tesla K20x</td>
</tr>
<tr>
<td><strong>Total Nodes</strong></td>
<td>22,640</td>
<td>4,224</td>
</tr>
<tr>
<td><strong>Total x86 Cores</strong></td>
<td>362,240</td>
<td>33,792</td>
</tr>
<tr>
<td><strong>Cores/Node</strong></td>
<td>16 FP x86 cores</td>
<td>8 FP x86 Cores, 2688 CUDA cores</td>
</tr>
<tr>
<td><strong>Memory/Node</strong></td>
<td>64 GB, 1.6 GHz</td>
<td>32 GB, 1.6 GHz</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>26.4 petabytes (disk), 380 petabytes (nearline)</td>
<td></td>
</tr>
<tr>
<td><strong>Interconnect</strong></td>
<td>Cray “Gemini” 3D Torus</td>
<td></td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>Cray Linux 6</td>
<td></td>
</tr>
</tbody>
</table>
Significant Performance Increase for Big FEA Problems (iForge 3 v. iForge 2)

iForge 3 = Intel Xeon E5 2680 v2 (“Ivy Bridge”) w. 256 GB of RAM per node
iForge 2 = Intel Xeon E5 2670 v1 (“Sandy Bridge”) w. 128 GB of RAM per node

Simulia Abaqus (Std.) on 8 iForge nodes
Application Breakthroughs on Blue Waters !!

15,000+ cores (LS-DYNA)

20,000+ cores (Fluent)

60,000+cores (WSMP)

100,000+ cores (Alya)

OpenACC with GPU-Aware-MPI

CPU+GPU (Abaqus)
LS-DYNA on Blue Waters (2013)

Hybrid LS-DYNA Parallel Scalability on NCSA's Blue Waters
Rolls-Royce case; 26.5M nodes, 80M DOFs, Time in Hours, Lower = Better

Wall Clock (hours) vs. CPU Cores

- iForge (Dell/Intel) outperforms Blue Waters (Cray/AMD) at “large” scale...
- ...while Cray outperforms at “extreme” scale
As scaling increases, communication becomes more important than computation.
Pushing LS-DYNA Further

NCSA Private Sector Program, Procter & Gamble, LSTC, and Cray

Real geometry, loads, boundary conditions, highly non-linear transient dynamic problem with difficult (eroding) contact conditions

MPP DYNA solver ported and optimized for Cray Linux and “Gemini” interconnect
LS-DYNA Scalability

From 400 days with serial execution to 4.3 hours on 15,000 cores!

Hybrid LS-DYNA Parallel Scalability on NCSA's Blue Waters
P&G Case, 72M nodes, Wallclock Time (Seconds); Lower= Better

Highest known scaling of any ISV FEA code!!
Future Challenges to DYNA Scaling

Collaboration is essential → model = NCSA + PSP Partner + ISV + Hardware Vendor

Identifying and building even larger problems (Companies have to think BIG!)

Memory management is critical (decomposition, distribution, MPMD, etc.)

Refining the hybrid solver (MPI + OpenMP) to minimize memory and communication

CPU affinity and Topological Awareness

Improving load balancing of (eroding) contact and rigid body algorithms
ANSYS Fluent at Extreme Scale

NCSA Private Sector Program, ANSYS, Dell, Cray, and Intel

Generic gas turbine combustor of 830 million mesh elements

Fluid flow, energy, chemical species transport, no DPM, no combustion

One of the biggest real-world cases ever
Source: Generic “Gas Turbine Combustor” provided by ANSYS

Code/Version: ANSYS Fluent v.15.0

Physics: Transient, turbulent flow, energy, chemical species transport, six non-reacting flows

Mesh size: 830 million cells

Used Intel Ivy-Bridge EX Node on iForge with 1.5 terabytes of RAM for building the mesh
Dr. Ahmed A. Taha, National Center for Supercomputing Applications (NCSA) reported scalability > 80% up to 20,480 cores for a 830 M case (April 2014)

**BREAKTHROUGH:**
- > 80% efficiency
- Super-Linear Scale-up

Best scaling of a real-world Fluent problem ever!!
Designed by the Barcelona Supercomputer Center as a multiphysics parallel FEA code

Unstructured spatial discretization, explicit and implicit integration in time

Staggered schemes (with iterations) for coupled physics on a single mesh

Mesh partitioning and hybrid parallel implementation

Uses built-in iterative CG solver with preconditioning

Highly modular, with each module representing a different physics; easy to combine them at job launch

Ported to Blue Waters in March 2014

Nominated for the Top Supercomputing Achievement at hpcwire-readers choice 2014!
2 Real-World Cases

**Human Heart**
- Non-linear solid mechanics
- Coupled with electrical propagation
- 3.4 billion elements, scaled to 100,000 cores

**Kiln Furnace**
- Transient incompressible turbulent flow
- Coupled with energy and combustion
- 4.22 billion elements, scaled to 100,000 cores
Alya – Kiln Furnace

BSC “Alya” on NCSA Blue Waters; 4.2 Billion Elements
Transient incompressible turbulent flow coupled with energy and combustion

Alya achieves nearly 90% scaling efficiency at 100,000 cores!!
17.4 years with serial execution to 1.8 hours on 100,000 cores!
The Future of HPC
A View from SC 2010

This chart is only illustrative

- Something else
- Traditional CPU with IB topology
GPGPU Computing

Applications

Libraries
- “Drop-in” Acceleration

OpenACC Directives
- Incrementally Accelerate Applications

Programming Languages (CUDA)
- Maximum Flexibility
OpenACC: Lowering Barriers to GPU Programming

The OpenACC API QUICK REFERENCE GUIDE

The OpenACC Application Programming Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++, and Fortran to be offloaded from a host CPU to an attached accelerator, providing portability across operating systems, host CPUs and accelerators.

Most OpenACC directives apply to the immediately following structured block or loop, a structured block is a single statement or a compound statement (C or C++) or a sequence of statements (Fortran) with a single entry point at the top and a single exit at the bottom.

Program myscience
... serial code ...
$acc kernels
  do k = 1,n1
  do i = 1,n2
    ... parallel code ...
    enddo
  enddo
$acc end kernels
... End Program myscience

OpenACC Compiler Hint
Minimize Data Movement!
The name of the game in GPGPU
Performance limitations – scaling MPI codes on GPUs

Memory Bandwidth

– Data path at scale for MPI resembles:
  – GPU <-> PCI <-> CPU <-> MPI_network

Vendor and community response:

– GPU(CUDA) Aware MPI
  • GPUDirect for Nvidia and Infiniband (mvapich)
  • MPICH_RDMA_ENABLED_CUDA (Cray mpi)
    Cray's implementation of MPI (MPICH2) allows GPU memory buffers to be passed directly to MPI function calls, eliminating the need to manually copy GPU data to the host before passing data through MPI

– Unified Virtual Memory (UVM) by Nvidia
  Unified Virtual Memory effectively pools the GPU and CPU memory into a single addressable space from the programmer's perspective.
  CUDA 6 supports this, OpenACC?
MPI+OpenACC Example: Solving Laplace (Heat) 2D Equation with FED

Iteratively converges to correct value (temperature) by computing new values at each point from the average of neighboring points.

\[
\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0
\]

\[
T_{i,j}^{k+1} = \frac{T_{i-1,j}^k + T_{i+1,j}^k + T_{i,j-1}^k + T_{i,j+1}^k}{4}
\]
Hybrid Laplace 2D
(MPI+OpenMP vs. GPU-Aware-MPI+OpenACC)

//Makes the address of device data at “u” available on the host
#pragma acc host_data use_device(u)
{
    // Exchange Data on domain boundaries
    if (id> 0)
        MPI_Sendrecv (&u[1*N], N, MPI_DOUBLE, id-1, 0,
                      &u[0*N], N, MPI_DOUBLE, id-1, 0, MPI_COMM_WORLD, &status);
    if (id < p-1)
        MPI_Sendrecv (&u[(my_rows-2)*N], N, MPI_DOUBLE, id+1, 0,
                      &u[(my_rows-1)*N], N, MPI_DOUBLE, id+1, 0, MPI_COMM_WORLD, &status);
}
diff = 0.0;
…
Comparing code with and without : host_data use_device

Without host_data use_device

80: update directive reached 18148 times
80: data copyout reached 671476 times
device time(us): total=1,667,834,025 max=2,674 min=25 avg=2,483

99: update directive reached 18148 times
99: data copyin reached 671476 times
device time(us): total=1,833,402,409

54 min. total  Huge Data Movement Bottleneck through PCIe !

With host_data use_device

102: kernel launched 18148 times grid: [6143] block: [64x4]
device time(us): total=226,312,480 max=12,616

3.7 min of GPU computing only !
ISV Multinode GPU Acceleration on XK7

Abaqus/Standard, Cluster Compatibility Mode
S4B Benchmark (5.23M Dofs), Higher=Better

Parallel Speedup wrt Serial CPU

Nodes

Cray XE6 (CPU only)
Cray XK7(CPU+GPU)
Thank you!