Performance on Trinity (a Cray XC40) with Acceptance-Applications and Benchmarks

Nathan Wichmann, Cindy Nuss, Pierre Carrier, Ryan Olson, Sarah Anderson and Mike Davis
Cray Inc.,
wichmann, cnuss, pcarrier, ryan, saraha, u3186@cray.com

Randal Baker, Erik W. Draeger, Stefan Domino, Anthony Agelastos, Mahesh Rajan
rsb@lanl.gov, draeger1@llnl.gov, spdomin@sandia.gov, amagela@sandia.gov, mrajan@sandia.gov

Cray User Group Meeting, May 8-11, 2016, London, UK

This work was supported in part by the U.S. Department of Energy. Sandia is a multi program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States National Nuclear Security Administration and the Department of Energy under contract DE-AC04-94AL85000.
NNSA’s First Advanced Technology System (ATS-1)

Previous Capability Computing Systems: Cielo, Sequoia

• Trinity (ATS-1) deployed by ACES (New Mexico Alliance for Computing at Extreme Scale) and sited at Los Alamos.
• ATS-2 will be led by LLNL, ATS-3 by ACES, ...

Cielo
- Cray XE6
- Nodes = 8944
- Memory > 291.5TB
- Peak Performance = 1.37 PF
- AMD MagnyCours (16 cores/node)

Sequoia
- IBM BG/Q
- Nodes = 98,304
- Memory = 1.6PB
- Peak Performance = 20PF
- IBM PowerPC A2 (16 cores/node)

Trinity
- Cray XC40
- Nodes > 19000
- Memory > 2PB
- Peak Performance > 40PF
- Intel Haswell (32 cores/node) & Knights Landing (72 cores/node)
Trinity Architecture: Phase-1 Haswell Nodes (installed 2015); Phase-2 KNL nodes (in 2016)

- Peak Node Performance: 32 cores * 16 FLOPs/cycle * 2.3 GHz = 1,177.6 GFLOPS/node
- Intel® Hyper-Threads and Intel® Turbo Boost
Trinity Phase-1 Acceptance
Completed December 2015
Focus here on application performance measures

Acceptance Tests and Criteria

1) **Capability Improvement (CI) metric**
   - CI Metric = problem-size-increase x run-time-speedup
   - 4X over a baseline performance measured on 2/3rd of the nodes on Cielo
   - runs at near full scale of Trinity
   - May use appropriately scaled inputs
   - Applications representative of planned Tri-lab productions apps

2) NERSC’s Sustained System Performance (SSP) target of 400; specified input: "large"

3) Microbenchamrks: Stream, OMB, SMB, mpimemu, psnap, pynamic

4) Run at full scale SSP benchmarks: miniFE, miniGhost, AMG, UMT and SNAP

<table>
<thead>
<tr>
<th>System</th>
<th>Cielo (XE6)</th>
<th>Trinity (XC40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Nodes</td>
<td>8,894</td>
<td>9,436</td>
</tr>
<tr>
<td>Total Cores</td>
<td>142,304</td>
<td>301,952</td>
</tr>
<tr>
<td>Processor</td>
<td>AMD MagnyCours</td>
<td>Intel Haswell</td>
</tr>
<tr>
<td>Processor ISA</td>
<td>SSE4a</td>
<td>AVX2</td>
</tr>
<tr>
<td>Clock Speed (GHz)</td>
<td>2.40</td>
<td>2.30</td>
</tr>
<tr>
<td>Cores/node</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Memory-per-core (GB)</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Memory</td>
<td>DDR3 1,333 MHz</td>
<td>DDR4 2,133 MHz</td>
</tr>
<tr>
<td>Peak node GFLOPS</td>
<td>153.6</td>
<td>1,177.6</td>
</tr>
<tr>
<td>Channels/socket</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Processor Cache:</td>
<td>8 x 64</td>
<td>16 x 32</td>
</tr>
<tr>
<td>L1(KB)</td>
<td>8 x 512</td>
<td>16 x 256</td>
</tr>
<tr>
<td>L2(KB)</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>L3(MB)</td>
<td>18 x 12x24</td>
<td>Aries Dragonfly</td>
</tr>
<tr>
<td>Interconnect Topology</td>
<td>Gemini 3D Torus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18x12x24</td>
<td></td>
</tr>
</tbody>
</table>
CI Metric and Applications

SNL App: SIERRA/Nalu:
- Low Mach CFD code for incompressible flows; unstructured mesh; LES/Turbulence Models
- Test Problem:
  - Turbulent open jet (Reynolds number of ~6,000)
  - Weak scaling meshes (R1:268k elements, R2:2.15M elements, .... R6: 9 billion elements)
- Figure of Merit: Solve time/Linear iteration (66%) & Assemble time/non-linear step(34%)

LANL App: PARTISN:
- Particle transport code provides neutron transport solutions on orthogonal meshes in one, two, and three dimensions
- Test Problem: MIC_SN (MIC with group-dependent Sn quadrature).
- Figure of Merit: Solver Iteration Time (should stay constant for weak scaling)

LLNL App: Qbox:
- first-principles molecular dynamics code used to compute the properties of materials at the atomistic scale
- Test Problem: benchmark problem is the initial self-consistent wave function convergence of a large crystalline gold system (FCC, a0 = 7.71 a.u).
- Figure of Merit: maximum total wall time to run a single self-consistent iteration with three non-self-consistent inner iterations)
SIERRA/Nalu Weak Scaling

Trinity 8X; SNL Nalu weak scaling; Assemble & Solve times

Cielo; Assemble
Trinity; Assemble
Trinity; Solve
Cielo; Solve

Average time: Sec

# of MPI Tasks
4 16 64 256 1024 4096 16384 65536 262144

R4 Mesh  R5 Mesh  R6 Mesh
Running the same problem (9 Billion element mesh) on 2.3 times the number of PEs resulted in a Capability Improvement Metric of $1.15/0.286 = 4.02$
PARTISN Weak Scaling
(2,880 & 11,520 zones/core)
(‘asis’: default MPI mapping; ‘grid’ mapping with grid_order)

PARTISN Solver Iteration Time

Nodes: XC40 32 ranks/node, one thread per rank- CIELO: 4 Ranks/node 4 threads/rank
# PARTISN CI Performance

<table>
<thead>
<tr>
<th>BASELINE on Cielo</th>
<th>Trinity Phase -1 CI Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>Nodes</td>
</tr>
<tr>
<td>MPI Tasks (4 Threads/task)</td>
<td>MPI Tasks</td>
</tr>
<tr>
<td>Problem Size Complexity Measure</td>
<td>Problem Size Complexity Measure</td>
</tr>
<tr>
<td>RunTime FOM</td>
<td>RunTime FOM</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8,192</td>
<td>32,768</td>
<td>2,880 zones/core</td>
<td>209.4 secs</td>
<td>9,418</td>
<td>301,376</td>
<td>11,520 zones/core</td>
<td>397.71 secs</td>
</tr>
</tbody>
</table>

Running a \((11,520/2880) \times 2.3 = 9.19\), larger problem on \(2.3\) times the number of PEs, took \(1.899\) times longer solver iteration time leading to a Capability Improvement Metric of \(\frac{9.19}{1.899} = 4.84\)
Qbox Weak Scaling
(1600 Atoms on 98,304 PEs; Trinity is 5.3X faster)
### Qbox CI Performance

<table>
<thead>
<tr>
<th>BASELINE on Cielo</th>
<th>Trinity Phase -1 CI Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>MPI Tasks</td>
</tr>
<tr>
<td>6,144</td>
<td>1 thread/task</td>
</tr>
<tr>
<td>98,304</td>
<td>8800/1600</td>
</tr>
</tbody>
</table>
Capability Improvement Summary

**Qbox:**
Nodes, = 9418  
MPI Tasks= 75,344  
OMP Threads/task=8  
Hyperthreading used  
Complexity Increase = 166.3  
Run Time Ratio = 0.208

**PARTISN:**
Nodes, = 9418  
MPI Tasks = 301,376  
OMP Threads/task = None  
Complexity Increase = 9.2  
Run Time Ratio = 0.526

**Trinity CI**  
Target =4.0

Performance Improvement Factor

<table>
<thead>
<tr>
<th></th>
<th>Trinity CI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sierra Nalu</strong></td>
<td>4.009</td>
</tr>
<tr>
<td><strong>Qbox</strong></td>
<td>34.700</td>
</tr>
<tr>
<td><strong>PARTISN</strong></td>
<td>14.517</td>
</tr>
<tr>
<td><strong>Average CI</strong></td>
<td></td>
</tr>
</tbody>
</table>

5/4/2016  
Sandia Unclassified Unlimited Release
PARTISN: Performance tuning

• `opt_sweep3d()`, which actually performs the KBA sweep that comprises the wave-front algorithm, took 85% of time; optimized for vectorization by Randy Baker and team at LANL.

• MPI `Isend/MPI_Recv` communications were frequent on the 2D processor mesh. Cray utility `grid_order` which “repacks” MPI ranks so that Cartesian mesh communication neighbors are more often on node was used to minimize communication overhead.
  – For example, a 16384 PE 128x128 mesh problem, use of `grid_order` improved MPI time by 42% and overall time by 18%

• Example MPICH_RANK_ORDER file:

```plaintext
# grid_order -R -Z -c 2,8 -g 554,544 -m 301376 -n 32
# Region 0: 0,0 (0..301375)
0,1,2,3,4,5,6,7,544,545,546,547,548,549,550,551,8,9,10,11,12,13,14,15,552,553,554,555,556,557,558,559
16,17,18,19,20,21,22,23,560,561,562,563,564,565,566,567,24,25,26,27,28,29,30,31,568,569,570,571,572,573,574,575 ...
```
grid_order impact studied at all scales up to 25% speedup!!

Trinity PARTISN Speedup using grid_order rank ordering
Trinity’s best performance was with 1 thread/MPI rank; Cielo’s 4 threads/rank.
Qbox: performance tuning

- Compute time dominated by parallel dense linear algebra and parallel 3D complex-to-complex Fast Fourier Transforms (FFT)
- Efficient single-node kernels necessary to achieve good peak performance
- The communication patterns are complex, with nonlocal communication occurring both within the parallel linear algebra library (ScaLAPACK) and in sub-communicator collectives within Qbox, which are primarily MPI_Allreduce and MPI_Alltoallv operations.
- Threading implemented as a mix of OpenMP and threaded single-node linear algebra kernels
- 5,600 atom Qbox simulation showed 2X performance gain for 8 OpenMP threads/task when compared to 2
Qbox optimal performance: Impact of $nrowmax$ and threads/task

Number of gold atoms

- 8800 atoms
- 5600 atoms
- 1024 atoms
- 256 atoms

Number of nodes

- <512 nodes
- <2000 nodes
- <10000 nodes

- Omp=2
  - Nrowmax=512

- Omp=2 to 8
  - Nrowmax=1024 to 2048

- Omp=8
  - Nrowmax=2048
Qbox performance: `grid_order` gave 6% to 30% performance gain

Qbox *iteration time* using MPI Grid ordering for 9418 node run
“grid_order -R -P -c 2,2 -g 34,1108” (without hyperthreading)
“grid_order -R -P -c 4,2 -g 68,1108” (with hyperthreading)

<table>
<thead>
<tr>
<th></th>
<th>2400 gold atoms</th>
<th>8800 gold atoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without grid order</td>
<td>456</td>
<td>9383</td>
</tr>
<tr>
<td>With grid order</td>
<td>315</td>
<td>8834</td>
</tr>
<tr>
<td>With grid order and hyperthreading</td>
<td>---</td>
<td>7974</td>
</tr>
</tbody>
</table>
NERSC’s Sustained System Performance (SSP) Metric

- A set of benchmark programs that represent a workload
- Computed as a geometric mean of the performance of eight Tri-Lab and NERSC benchmarks
  - miniFE, miniGhost, AMG, UMT, SNAP, miniDFT, GTC and MILC
- Problem size (“large”) specified
- Baseline data collected on NERSC’s Hopper (XE6); Baseline node-count suggested by production use
- Trinity run node-count not specified; For a few benchmarks, use of fewer nodes than the baseline, skewed the metric
- SSP being revised by NERSC, ACES for use with ATS-3
Trinity Phase-1 SSP target was 400: Achieved 500

Baseline SSP performance on NERSC’s Hopper (Cray XE6)

<table>
<thead>
<tr>
<th>Hopper Nodes</th>
<th>6384</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Application Name</th>
<th>MPI Tasks</th>
<th>Threads</th>
<th>Nodes Used</th>
<th>Reference Tflops</th>
<th>Time (seconds)</th>
<th>Pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>miniFE</td>
<td>49152</td>
<td>1</td>
<td>2048</td>
<td>1065.151</td>
<td>92.4299</td>
<td>0.0056</td>
</tr>
<tr>
<td>miniGhost</td>
<td>49152</td>
<td>1</td>
<td>2048</td>
<td>3350.20032</td>
<td>95.97</td>
<td>0.0170</td>
</tr>
<tr>
<td>AMG</td>
<td>49152</td>
<td>1</td>
<td>2048</td>
<td>1364.51</td>
<td>151.187</td>
<td>0.0044</td>
</tr>
<tr>
<td>UMT</td>
<td>49152</td>
<td>1</td>
<td>2048</td>
<td>18409.4</td>
<td>1514.28</td>
<td>0.0059</td>
</tr>
<tr>
<td>SNAP</td>
<td>49152</td>
<td>1</td>
<td>2048</td>
<td>4729.66</td>
<td>1013.1</td>
<td>0.0023</td>
</tr>
<tr>
<td>miniDFT</td>
<td>10000</td>
<td>1</td>
<td>417</td>
<td>9180.11</td>
<td>906.24</td>
<td>0.0243</td>
</tr>
<tr>
<td>GTC</td>
<td>19200</td>
<td>1</td>
<td>800</td>
<td>19911.348</td>
<td>2286.822</td>
<td>0.0109</td>
</tr>
<tr>
<td>MILC</td>
<td>24576</td>
<td>1</td>
<td>1024</td>
<td>15036.5</td>
<td>1124.802</td>
<td>0.0131</td>
</tr>
</tbody>
</table>

Geom. Mean= 0.0082
SSP= 52.1212

SSP performance on Trinity

<table>
<thead>
<tr>
<th>Trinity Nodes</th>
<th>9436</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Trinity SSP</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Name</td>
<td>MPI Tasks</td>
<td>Threads</td>
<td>Nodes Used</td>
<td>Reference Tflops</td>
<td>Time (seconds)</td>
</tr>
<tr>
<td>miniFE</td>
<td>49152</td>
<td>1</td>
<td>1536</td>
<td>1065.151</td>
<td>49.5116</td>
</tr>
<tr>
<td>miniGhost</td>
<td>49152</td>
<td>1</td>
<td>1536</td>
<td>3350.20032</td>
<td>1.77E+01</td>
</tr>
<tr>
<td>AMG</td>
<td>49152</td>
<td>1</td>
<td>1536</td>
<td>1364.51</td>
<td>66.233779</td>
</tr>
<tr>
<td>UMT</td>
<td>49184</td>
<td>1</td>
<td>1537</td>
<td>18409.4</td>
<td>454.057</td>
</tr>
<tr>
<td>SNAP</td>
<td>12288</td>
<td>1</td>
<td>768</td>
<td>4729.66</td>
<td>1.77E+02</td>
</tr>
<tr>
<td>miniDFT</td>
<td>2016</td>
<td>2</td>
<td>63</td>
<td>9180.11</td>
<td>377.77</td>
</tr>
<tr>
<td>GTC</td>
<td>19200</td>
<td>1</td>
<td>300</td>
<td>19911.348</td>
<td>868.439</td>
</tr>
<tr>
<td>MILC</td>
<td>12288</td>
<td>1</td>
<td>384</td>
<td>15036.5</td>
<td>393.597</td>
</tr>
</tbody>
</table>

Geom. Mean= 0.0530
SSP= 500.0177
Micro-benchmark Results

STREAM Performance at a Trinity node

<table>
<thead>
<tr>
<th>Function</th>
<th>Copy (MB/s)</th>
<th>Scale (MB/s)</th>
<th>Add (MB/s)</th>
<th>Triad (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td>108,014</td>
<td>108,653</td>
<td>118,850</td>
<td>119,077</td>
</tr>
</tbody>
</table>

PSNAP OS jitter/Noise Benchmark Results

NR: 9436  
Average Slowdown: 0.15%  
Min Slowdown: 0.13%  
Max Slowdown: 0.18%  
Maximum percentage slowdown at a core was measured to be 0.18%.
Conclusions

- Several months of effort by the Cray and Tri-Lab teams resulted in exceeding acceptance requirements.
- Based on benchmark results, we anticipate production Trinity apps, will see a gain of 2x-6x over Cielo (with same number of PEs).
- The benefit of a hybrid code (MPI+threads) clearly seen with Qbox
  - 2x the performance with 8 threads over 2 threads per task
- Use of `grid_order` resulted in good performance gains.