The Case of the Missing Supercomputer Performance

Achieving Optimal Performance on the 8,192 Processors of ASCI Q

Fabrizio Petrini, Darren J. Kerbyson, Scott Pakin
Los Alamos National Laboratory
The Problem
ASCII Q running SAGE is not performing as well as it should

• ASCII Q
  - 8192 processors
  - Installed at Los Alamos National Laboratory (LANL)
  - 2nd fastest supercomputer (2003)

• SAGE
  - Eulerian hydrodynamics application
  - 150,000 lines of Fortran and MPI code
How the authors

I. determined that ASCI Q was not performing well
II. identified the source of the performance loss
III. improved performance
IV. remeasure the performance
• A performance model of SAGE (verified on many systems to predict performance within 10% error)

• Measured ASCI Q one half (4096 processors) at a time, the two halves are consistent

• SAGE performs significantly worse than was predicted by the model

Figure 1: Expected and measured SAGE performance
• Performance of using 4 processors/node is different

Figure 2: Difference between modeled and measured SAGE performance when using 1, 2, 3, or 4 processors per node
• > 256 nodes, using 4 processors/node is worse than using 3 processors/node
• > 512 nodes, using 4 processors/node is worse than using 2 processors/node

Figure 3: Effective SAGE processing rate when using 1, 2, 3, or 4 processors per node
SAGE performs a constant amount of work per cycle and could be expected to take a constant amount of time to finish.

Cycle time ranges from 0.7 to 3 seconds, greater than a factor of 4 in variability.

Figure 4: SAGE cycle-time measurements on 3,584 processors.
• Collective-communication operations: allreduce, reduction, account for the increase in cycle time

Figure 5: Profile of SAGE’s cycle time
Using 4 processors per node
• $\leq 3$ processors / node, latency $< 300$ us

• A problem arises when using all 4 processors within a node, latency $> 3$ms

Figure 6: allreduce latency as a function of the number of nodes and processes per node
• Synthetic parallel benchmark, alternatively computes for 0, 1, or 5ms then performs either an allreduce or a barrier

• Ideal: grow logarithmically with increasing number of nodes, insensitive to computational granularity

• Actual: grow linearly with number of nodes, and increase with computational granularity

Figure 7: allreduce and barrier latency with varying amounts of intervening computation
• Improved performance of allreduce 7x better
• SAGE spends 51% of time in allreduce, should lead to 78% performance gain in SAGE
• Actual: only marginal improvement
• Eliminates MPI implementation and network as source of performance loss
• Hypothesis: periodic system activities were interfering with application execution, causing performance variability (“noise”)
• A simple benchmark of running synthetic computation for 1000 seconds in the absence of noise

• Slowed down experienced by each process is low, < 2.5%

• Contradicts the “noise” hypothesis
A new benchmark of running 1 million iterations of synthetic computation, each iteration precisely 1ms in the absence of noise, total 1000 seconds

Result is the same as previous benchmark

Aggregate the four processor measurements taken on each node

Found regular pattern of noise: every 32 nodes contain some nodes that are noisier
• All nodes suffer a moderate amount of noise

• Node 0 (cluster manager), node 1 (the quorum node), node 31 (the RMS cluster monitor) suffer more

Figure 12: Slowdown per node within each 32-node cluster
Table 2: Summary of noise on each 32-node cluster

<table>
<thead>
<tr>
<th>Source of noise</th>
<th>Duration (ms)</th>
<th>Location (nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>0–3</td>
<td>✔️ ✔️ ✔️ ✔️ ✔️</td>
</tr>
<tr>
<td>RMS daemons</td>
<td>5–18</td>
<td>✔️ ✔️ ✔️ ✔️ ✔️</td>
</tr>
<tr>
<td>TruCluster daemons</td>
<td>&gt;18</td>
<td>✔️ ✔️ ✔️</td>
</tr>
</tbody>
</table>

Figure 13: Identification of the events that cause the different types of noise
A delay in a single process slows down the whole application.

Not possible or cost effective to remove daemons or kernel threads.

Solution: coschedule the activities, pay penalty only once.

Figure 14: Illustration of the impact of noise on synchronized computation.
• Developed a simulator, taking account into all events

• Each event: <F, L, E, P>

• Frequency F, average duration L, the distribution E, the placement (set of nodes) P

• Remove noise on either node 0, 1 or 31, only 15% improvement

• Remove all three nodes, 35%

• Remove kernel noise: significant improvement

• More performance is lost to short but frequent noise on all nodes than to long but less frequent noise on just a few nodes

Figure 15: Simulated vs. experimental data with progressive exclusion of various sources of noise in the system
• The authors undertook some optimizations on ASCI Q

• Removed about ten daemons from all nodes

• Decreased the frequency of RMS monitoring by a factor of 2 on each node (30 -> 60 seconds)

• Moved several TruCluster daemons from node 1 and 2 to node 0 on each cluster

• Expected speed improvement is a factor of 2.2
• 3 different computational granularity - 0, 1, 5ms (length of computation between two barriers)

• Only shows performance improvement of micro benchmark

• Will this improve performance of SAGE?

Figure 16: Performance improvements obtained on the barrier-synchronization microbenchmark for different computational granularities
• Jan-27-03 and May-01-03 are measured after noise removal

• May-01-03 (min) is min cycle time of over 50 cycles

• There is room for further improvement: remove one processor from node 0 and node 31, run system tasks

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Usable processors</th>
<th>Cycle time</th>
<th>Processing rate (10^6 cell updates/sec.)</th>
<th>Improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unoptimized system</td>
<td>8,192</td>
<td>1.60</td>
<td>69.1</td>
<td>— N/A —</td>
</tr>
<tr>
<td>3 processes/node</td>
<td>6,144</td>
<td>0.64</td>
<td>129.3</td>
<td>1.87</td>
</tr>
<tr>
<td>Without node 0</td>
<td>7,936</td>
<td>0.87</td>
<td>123.1</td>
<td>1.78</td>
</tr>
<tr>
<td>Without nodes 0 and 31</td>
<td>7,680</td>
<td>0.86</td>
<td>120.6</td>
<td>1.75</td>
</tr>
<tr>
<td>Without nodes 0 and 31 (best observed)</td>
<td>7,680</td>
<td>0.68</td>
<td>152.5</td>
<td>2.21</td>
</tr>
<tr>
<td>Model</td>
<td>8,192</td>
<td>0.63</td>
<td>178.4</td>
<td>2.58</td>
</tr>
</tbody>
</table>

Figure 17: SAGE performance: expected and measured after noise removal
• Categorize the relative impact of each of the three primary sources of noise

• The computational granularity of the application “enter in resonance” with noise of a similar harmonic frequency and duration

![Graph](image)

Figure 18: Cumulative noise distribution for barrier synchronizations with different computational granularities

• X axis: duration of an individual occurrence of system noise

• Y axis: cumulative amount of barrier performance lost to noise

• 0 - 3ms: kernel activity, 5 - 18ms: RMS daemons, >18ms TruCluster daemons