Emulating Exascale using the GREMLIN Framework (With a Focus on Power)

Leogang

High Performance Computing Workshop

Matthias Maiterth

2. März 2015
1 Exascale: Challenges for the next generation of HPC
2 The GREMLIN Framework
3 Proxy Applications
4 Scaling Studies using Power GREMLINs
   Weak Scaling
   Strong Scaling
5 Future Work
### Exascale Projection

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2018</th>
<th>Factor Change</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Peak</strong></td>
<td>2 Pf/s</td>
<td>1 Ef/s</td>
<td>500</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>6 MW</td>
<td>20 MW</td>
<td>3</td>
</tr>
<tr>
<td><strong>System Memory</strong></td>
<td>0.3 PB</td>
<td>10 PB</td>
<td>33</td>
</tr>
<tr>
<td><strong>Node Performance</strong></td>
<td>0.125 Gf/s</td>
<td>10 Tf/s</td>
<td>80</td>
</tr>
<tr>
<td><strong>Node Memory BW</strong></td>
<td>25 GB/s</td>
<td>400 GB/s</td>
<td>16</td>
</tr>
<tr>
<td><strong>Node Concurrency</strong></td>
<td>12 cpus</td>
<td>1,000 cpus</td>
<td>83</td>
</tr>
<tr>
<td><strong>Interconnect BW</strong></td>
<td>1.5 GB/s</td>
<td>50 GB/s</td>
<td>33</td>
</tr>
<tr>
<td><strong>System Size (nodes)</strong></td>
<td>20 K nodes</td>
<td>1 M nodes</td>
<td>50</td>
</tr>
<tr>
<td><strong>Total Concurrency</strong></td>
<td>225 K</td>
<td>1 B</td>
<td>4,444</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>15 PB</td>
<td>300 PB</td>
<td>20</td>
</tr>
<tr>
<td><strong>Input/Output bandwidth</strong></td>
<td>0.2 TB/s</td>
<td>20 TB/s</td>
<td>100</td>
</tr>
</tbody>
</table>

---

1 ASCAC Subcommittee Report: The Opportunities and Challenges of Exascale Computing, Department of Energy, Office of Science, Fall 2010
An Introduction to the GREMLIN Framework

- Emulating Exascale by Resource Restriction
- Focus on Power, Memory, Resilience and Noise
- Scalable Framework running on current HPC systems
- Co-Design as Goal
The GREMLIN Framework
Architecture

- Implemented as MPI-Tools
- Uses MPI Profiling Interface PMPI (LD_PRELOAD)
- Combining Gremlins using P^nMPI
- Easily extensible
- Usable with C/C++ & Fortran Applications
Power GREMLINs

- Increase in #Processors $\implies$ Increase in System Power
- Less power available per processor
- Artificially restrict Package Power to study CPU behavior.
- Implementation of the Power GREMLINs:
  - Power Limits enforced by Intel’s RAPL (Running Average Power Limit) (alternative not used: DVFS)
  - Uses libraries for safe & easy access to MRSs via msr-safe and libmsr.
  - 3 GREMLINs: set per processor Power Limit, set per MPI execution Power Limit, read Power consumption
- Possible extension Power GREMLINs for Memory, Interconnect, etc.
Resilience GREMLINs

- Increased component failure in Future HPC system.
- ECC memory mean time of failure: 1,500,000 hours.
- Cray XT5 @ ORNL with 75,000 memory units: 20h
- Distributed recovery mechanisms hard to test.
- Emulating faults and errors, by fault injection using GREMLINs.
Memory GREMLINs

- Slower increase in Memory compared to concurrency (Memory Wall)
- Impact on memory size, bandwidth and caches available per core.
- 2 GREMLINs so far:
  - Cache Capacity
  - Memory Bandwidth
- Similar standalone tools exist, e.g. Cache Pirate and Bandwidth Bandit
AMG (Algebraic Multigrid) solver used in the HYPRE library

AMG solves PDEs using a hierarchy of discretization

Iterative approach: Move between fine and coarse grids
  ▶ Fine to coarse: eliminate high frequency errors using a smoother.
  ▶ At coarsest grid: Solve small linear system.
  ▶ Coarse to fine: propagate estimated solution to finer level by interpolation.
  ▶ At finest level: Terminate if residual error is small enough.

Used as solver for unstructured meshes, e.g. for plastic and elastic deformation.

Power consumption during compute phase: 103 W

Implemented using ISO-C, MPI and OpenMP.
<table>
<thead>
<tr>
<th>Processor architecture</th>
<th>Xeon 8-core E5-2670 (Intel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>TOSS</td>
</tr>
<tr>
<td>Process clock speed</td>
<td>2.6 GHz</td>
</tr>
<tr>
<td>Nodes</td>
<td>1,296</td>
</tr>
<tr>
<td>Cores per node</td>
<td>16</td>
</tr>
<tr>
<td>Total cores</td>
<td>20,736</td>
</tr>
<tr>
<td>Memory per node</td>
<td>32 GB</td>
</tr>
<tr>
<td>Total memory</td>
<td>41.5 TB</td>
</tr>
<tr>
<td>Thermal Design Power</td>
<td>115 W</td>
</tr>
</tbody>
</table>

- Kernel module and libraries for RAPL available
  *(libmsr & msr-safe)*
Weak Scaling
AMG2013

- 50 W  
- 80 W  
- 115 W  
- 65 W  
- 95 W  

time [s]

#Nodes

30 35 40 45 50 55 60

1 2 4 8 16 32 64 128 256

2. März 2015
Matthias Maiterth
Performance Variation
AMG2013
Performance Variation
AMG2013 - reordered

Node ID

<table>
<thead>
<tr>
<th>1</th>
<th>16</th>
<th>32</th>
<th>48</th>
<th>64</th>
<th>80</th>
<th>96</th>
<th>112</th>
<th>128</th>
<th>144</th>
<th>160</th>
<th>176</th>
<th>192</th>
<th>208</th>
<th>224</th>
<th>240</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
<td>-</td>
<td>50 W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 50 W

1 Node
2 Nodes
4 Nodes
8 Nodes
16 Nodes
32 Nodes
64 Nodes
128 Nodes
256 Nodes

Node ID
time[s]
Future Work
Using Power

▶ Optimizing for execution time
  (Problem size / Core count / Power )
▶ Equalizing runtime using Power Balancing
▶ Overprovisioning HPC resource
  (More CPUs than operable)
▶ Power consumption in Memory and Interconnect