Update on Topology Aware Scheduling (aka TAS)
Work done in collaboration with

- J. Enos, G. Bauer, R. Brunner, S. Islam
- R. Fiedler
- Adaptive Computing
When things look good

Image credit: Dave Semeraro
What’s the problem?

- Efficient job scheduling on a large torus is not easy.
- Over time (between large jobs, reboots) fragmented allocations appear.
- Fragmentation can lead to degraded and variable application performance.

Image credit: Robert Sisneros
• 4,116 XE node jobs run at different times.
• Run to run variability
  • makes it difficult to assign a reasonable wall clock time.
  • has an impact on job throughput.
Blue Waters Torus

- 24x24x24 gemini routers, 2 nodes each
- XE nodes not shown
- XK nodes (red) 15x6x24
- XIO nodes (yellow)
- Links along X & Z dimensions 2x faster than links along Y.

Image credit: VMD
While waiting for TAS

- Changed default node ordering to favor XZ slabs; improving aggregate interconnect bandwidth and location.
- Workload of MILC, NWCHEM, PSDNS ChaNGa, NAMD, WRF, CESM, DNS_distuf showed average improvements in runtime of 15% to 25%.
- Change does not address job-job interaction.
• Experimented with pre-defined moab features (explicit node lists) and nodesets of these features.
• Worked well for some teams to improve performance and limit job-job interference.
• Impacted job throughput (having to wait longer for specific sets of nodes).
• Responsiveness of moab adversely affected.
Impact of topology aware scheduling

• Important to scientists
  • Reduction in time to solution
  • Reduction of run-to-run variation
  • Get science done
• Important to the project and funder
  • Get Science done
  • System utilization
How to interact with TAS

- Topology aware user specifications
  - `#PBS -l geometry=X×Y×Z` with some wild cards
  - Application communication characteristics:
    - `#PBS -l comm={high|low}[::{high|low}][::{global|local}]`
    - “low” or “high” communication intensity.
      - bi-section bandwidth consideration.
    - “low” of “high” communication sensitivity.
      - allow for fragmented node allocations.
    - “global” or “local” as the dominant communication pattern.
  - Cost function for waiting for shape.
Workload Tests

- Initial tests limited to allocate convex shapes to lessen internode communication interference on other jobs (dimension ordered routing).
- The scheduler was able to try different rectangular shapes weighted by aggregate bandwidth.
Workload Test

- Synthetic workload composed of several applications
  - MILC, PSDNS, NAMD, NWChem, ChaNGa, QMCPACK, DNS_distuf, WRF, SpecFEM3D_globe.
  - Represents a broad range of communication patterns.
  - Numerous representative node counts and scaled run times based on actual Blue Waters production logs.
  - Initial conditions set by stub jobs.
- 1544 jobs (XE and XK) run in two hour window
- Good scheduler responsiveness
- Good utilization
• Top 10 jobs shown.
• XZ slabs favored.
• Some jobs specified X×Y×Z.

Image credit: Dave Semeraro
Preliminary Workload Test Results

- 324 nodes - MILC
- 3 shapes used in workload testing.
- “none” collected in batch
- 17% reduction in average runtime
- 10x reduction in CoV.
- Larger impact at larger scales.
Preliminary Workload Test Results

- Worst Application run time CoV is less than 2%
- Worst ‘Per Shape’ Application CoV is less than 1.25%
Preliminary Workload Test Results

- Speed-up from using topology aware scheduling

![Bar chart showing average speed-up for MILC 4116, NAMD 3272, and NWChem 400]
Node Selection and Task Layout

- Most codes will need to consider MPI rank ordering to take full advantage of nodes provided by topology aware scheduler.
- Topoaware: Provides task mapping for 2, 3, & 4D Cartesian grid virtual topologies.
  - Developed by Bob Fiedler, Cray.
  - In each z-pencil, extends set of selected geminis along z if needed to skip unavailable nodes
  - Determines multiple valid layouts and evaluates layout quality
  - Allows unbalanced layouts
    - Nodes on prism boundaries may have fewer tasks
    - Enables more good layouts for more virtual topology sizes
  - Scheduler ensures allocation has desired gemini count in each z-pencil
Topaware tests: Halo exchange

- Virtual topology: 32x32x32
- 10x improvement possible.
- Hop count not the only story.
- Reduction in congestion and improved bandwidth important.
- `grid_order` provided by Cray to order communication between nearest neighbors in a grid.

<table>
<thead>
<tr>
<th>Placement</th>
<th>Iter time (ms)</th>
<th>Max hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default 8x8x8</td>
<td>11.315</td>
<td>9</td>
</tr>
<tr>
<td>Grid_order 8x8x8</td>
<td>7.722</td>
<td>16</td>
</tr>
<tr>
<td>Topaware 8x8x8</td>
<td>2.771</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 11x6x11 (unbalanced)</td>
<td>1.287</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 11x8x8 (unbalanced)</td>
<td>1.147</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 8x8x11 (unbalanced)</td>
<td>1.214</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 11x7x8 (unbalanced)</td>
<td>1.782</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 8x7x11 (unbalanced)</td>
<td>1.737</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 11x8x7 (unbalanced)</td>
<td>1.580</td>
<td>2</td>
</tr>
<tr>
<td>Topaware 7x8x11 (unbalanced)</td>
<td>1.690</td>
<td>2</td>
</tr>
</tbody>
</table>
Topaware tests: MILC

- MILC
  - Virtual topology 21x2x21x24
  - 1764 nodes, 12 tasks each
  - 21x2x21 geminis
  - 2.2x faster with Topaware than with grid_order –c 2,2,2,2 on same nodes
  - grid_order can provide 2x over not using grid_order.
  - See Topology Consideration talk at December 2013 workshop.

<table>
<thead>
<tr>
<th>Placement</th>
<th>Run Time (10 iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid_order</td>
<td>254.0</td>
</tr>
<tr>
<td>Topaware</td>
<td>116.4</td>
</tr>
</tbody>
</table>
Conclusions and Next Steps

• From initial tests with topology aware scheduling we see
  • improvements in overall performance and run-to-run variability
  • promising utilization numbers
• Further tests coming and then deployment.
• What we like to see on Blue Waters …

Image credit: Dave Semeraro