INRIA/UIUC/ANL JLPC – June 2012
“Mapping and Scheduling Break Out”
Bill Kramer
Cray XE6/XK7 - 276 Cabinets

**XE6 Compute Nodes**
- 5,688 Blades – 22,640 Nodes – 362,240 Cores

**XK7 GPU Nodes**
- 768 Blades – 3,072 Nodes – 24,576 Cores – 3,072 GPUs

Supporting systems: LDAP, RSA, Portal, JIRA, Globus CA, Bro, test systems, Accounts/Allocations, CVS, Wiki
## Blue Waters XE6 Node

Blue Waters contains 22,640 XE6 compute nodes

<table>
<thead>
<tr>
<th>Node Characteristics</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Core Modules*</td>
<td>16</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>313 Gflops/sec</td>
</tr>
<tr>
<td>Memory Size</td>
<td>64 GB per node</td>
</tr>
<tr>
<td>Memory Bandwidth (Peak)</td>
<td>102 GB/sec</td>
</tr>
<tr>
<td>Interconnect Injection Bandwidth (Peak)</td>
<td>9.6 GB/sec per direction</td>
</tr>
</tbody>
</table>

*Each core module includes 1 256-bit wide FP unit and 2 integer units. This is often advertised as 2 cores, leading to a 32 core node.*
Cray XK7 and a Path to the Future

Blue Waters contains 3,072 NVIDIA Kepler (GK110) GPUs

<table>
<thead>
<tr>
<th>XK7 Compute Node Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Processor</td>
</tr>
<tr>
<td>Host Processor Performance</td>
</tr>
<tr>
<td>Kepler Peak (DP floating point)</td>
</tr>
<tr>
<td>Host Memory</td>
</tr>
<tr>
<td>Kepler Memory</td>
</tr>
</tbody>
</table>
Blue Waters 3D Torus
Size
23 x 24 x 24

Service Nodes spread Throughout the torus

Blue Waters High Speed Network

Compute Nodes
- Cray XE6 Compute
- Cray XK7 Accelerator

Service Nodes
- Operating System
  - Boot
  - System Database
- Login/Network
  - Login Gateways
  - Network
- Lustre File System
  - LNET Routers

Interconnect Network

InfiniBand
Login Server(s)
Network(s)
GigE
Fibre Channel
Infiniband
SMW
Boot Raid
Lustre
BW Sustained Petascale Performance Measures

- Original NSF Benchmarks
  - Full Size – QCD (MILC), Turbulence (PNSDNS), Molecular Dynamics (NAMD)
  - Modest Size – MILC, Paratec, WRF
- SPP – is a time to solution metric that is using the planned applications on representative parts of the Science team problems
  - Represents end to end problem run including I/O, pre and post phases, etc.
  - Coverage for science areas, algorithmic methods, scale
- SPP Application Mix (details and method available)
  - NAMD – molecular dynamics
  - MILC, Chroma – Lattice Quantum Chromodynamics
  - VPIC, SPECFEM3D – Geophysical Science
  - WRF – Atmospheric Science
  - PPM – Astrophysics
  - NWCHEM, GAMESS – Computational Chemistry
  - QMCPACK – Materials Science
- At least three SPP benchmarks run at full scale
- XK nodes have to add 15% more SPP
• Original NSF Benchmarks
  • Full Size – \textbf{QCD (MILC)}, \textbf{Turbulence (PNSDNS)}, Molecular Dynamics (NAMD)
  • Modest Size – MILC, Paratec, WRF
• SPP – is a time to solution metric that is using the planned applications on representative parts of the Science team problems
  • Represents end to end problem run including I/O, pre and post phases, etc.
  • Coverage for science areas, algorithmic methods, scale
• SPP Application Mix (details and method available)
  • NAMD – molecular dynamics
  • \textbf{MILC}, Chroma – Lattice Quantum Chromodynamics
  • VPIC, SPECFEM3D – Geophysical Science
  • \textbf{WRF} – Atmospheric Science
  • \textbf{PPM} – Astrophysics
  • NWChem, GAMESS – Computational Chemistry
  • QMCPACK – Materials Science
• At least three SPP benchmarks run at full scale
• \textbf{Bold} = codes known to benefit from topology optimization
Effective Use of Complex Systems

- Only a small subset of BW applications use Charm++ (sorry Sanjay). The vast majority are MPI or MPI/OpenMP.
- Increasing performance requires dramatic increases in parallelism that then generates complexity challenges for science and engineering teams.
- Goals that require improved topology mapping and scheduling:
  - Scaling applications to large core counts on general-purpose CPU nodes.
  - Effectively using parallel IO systems for data-intensive applications and innovative storage and data paradigms.
  - Effectively using limited bandwidth of interprocessor network.
  - Enhancing application flexibility to increasing effective, efficient use of systems.
Performance and Scalability

• The problem is fewer applications are able to scale in the face of limited bandwidths. Hence the need to work with science teams and technology providers to
  • Develop better process-to-node mapping using for graph analysis to determine MPI behavior and usage patterns.
  • Topology Awareness in Applications and in Resource Management
  • Improve use of the available bandwidth (MPI implementations, lower level communication, etc.).
  • Consider new algorithmic methods
  • Considering alternative programming models that improve efficiency of calculations
Performance and Scalability

• Use of heterogeneous computational units
  • While more than ½ of the science has some GPU based investigations, only a few are using GPUs in production science
    • Many applications are GPUized only in a very limited way
    • Few are using GPUs at scale (more GPU resources are relatively small scale with limited networks)
  • Help the science teams to make more effective use of GPUs consists of two major components.
    • Introduce compiler and library capabilities into the science team workflow to significantly reduce the programming effort and impact on code maintainability.
      • OpenACC support is the major path to more general acceptance
    • Load balancing at scale

• Storage Productivity
  • Interface with improved libraries and middle ware
  • Modeling of I/O
  • On-line and Near-line transparent interfaces
Application Flexibility

- Using both XE and XK nodes in single applications
  - For multi-physics applications that provide a natural decomposition into modules is to deploy the most appropriate module(s) different computational units.
  - For applications use the Charm++ adaptive runtime system, heterogeneity can be handled without significant changes to the application itself.
  - Some applications naturally involve assigning multiple blocks to individual processors include multiblock codes (typically in fluid dynamics), and the codes based on structured adaptive mesh refinement.
  - The application-level load balancing algorithms can be modified to deal with the performance heterogeneity created by the mix of nodes.

- Malleability
  - Understanding topology given and maximizing effectiveness
  - Being able to express desired topology based on algorithms
  - Middleware support
Flexibility - Application Based Resiliency

- Multiple layers of Software and Hardware have to coordinate information and reaction
- Analysis and understanding is needed before action
- Correct and actionable messages need to flow up and down the stack to the applications so they can take the proper action with correct information
- Application Situational Awareness - need to understand circumstances and take action
- Flexible resource provisioning needed in real time
- Interaction with other constraints so sub-optimization does not adversely impact overall system optimization
Application Topology Challenges - Opportunities

- Deciding best/improved topology layout for major applications
  - Project Improvements in run time
- Interface and adjustments for MPI task topology
- Improved monitoring/measurement of communication and performance as topology changes
- Flexibility expressing desired topology – can not over specify and expect to have good throughput
- Visualization of layouts and traffic
System Topology Challenges - Opportunities

- Ability to express both logical and physical topology layout/options
  - Current resource manager just presents a logical list of available nodes – not related to logical or physical topology
- Estimating overhead and increase wait time to accumulate better topologies.
- Providing interfaces for different topology aware scheduling algorithms
  - With multidimension weighting
- Schedule for I/O performance/topology
- Visualization of layouts
Topology Awareness and Scheduling

Summary

• Topology aware use and scheduling is critical to improving application performance for current and future architectures
• Application and System challenges have to be overcome at multiple levels
• Deterministic solutions are not likely except in small scale or dedicated time
  • More likely there will be flexible “proximity” refinements