Applications on Cray XK6

Roberto Ansaloni
Alistair Hart
Wilfried Oed

ZKI-Arbeitskreis Supercomputing
19.-20.4.2012
The great technology inflection point of mid last decade
  - Power has become a huge cost and the major design constraint
  - Future performance from \textit{parallelism}
    - Not clock rate
    - Not complexity

Conclusion
  - Need lots of very simple cores optimized for power on parallel code
  - But also need some fast cores for serial code
  \Rightarrow \text{heterogeneous processing (a.k.a. accelerators)}

GPUs are looking very promising for HPC
  - They now have good DP FP and error protection
  - They ship \textit{O(100M)} units a year
  - It looks like they can credibly support both masters
### XK6 Compute Node Characteristics

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Processor</td>
<td>AMD Series 6100 (Interlagos)</td>
</tr>
<tr>
<td>Host Processor</td>
<td>147 Gflops</td>
</tr>
<tr>
<td>Tesla X2090 Cores</td>
<td>448</td>
</tr>
<tr>
<td>Tesla X2090 Perf.</td>
<td>600+ Gflops</td>
</tr>
<tr>
<td>Host Memory</td>
<td>16 or 32GB, 1633 MHz DDR3</td>
</tr>
<tr>
<td>Tesla X2090 Memory</td>
<td>6GB GDDR5 capacity, 170 GB/sec</td>
</tr>
<tr>
<td>Upgradeable to KEPLER</td>
<td>many-core processor</td>
</tr>
</tbody>
</table>

- **10 12X Gemini Channels**
  (Each Gemini acts like two nodes on the 3D Torus)
- **High Radix YARC Router with adaptive Routing**
  168 GB/sec capacity
• Four nodes per blade
• Gemini Mezzanine
• Plug compatible with Cray XE6 blade
• Configurable processor, memory and SXM GPU card
● Upgrading Jaguar (XT5) to Titan (XK6)
● Peak performance 10~20 petaflops
● Efficiency & scalability of unprecedented scale
● Powerful hybrid of GPUs/CPUs & blazing interconnect

**Phase 5: System Configuration**
Name: Jaguar
Architecture: XK6
Processor: 16-Core AMD
Cabinets: 200
Nodes: 18,688
Cores/node: 16
Total cores: 299,008
Memory/node: 32GB
Memory/core: 2GB
Interconnect: Gemini
GPUs: 960
## Blue Waters System

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray System &amp; Storage cabinets:</td>
<td>&gt;300</td>
</tr>
<tr>
<td>Compute nodes:</td>
<td>&gt;25,000</td>
</tr>
<tr>
<td>Usable Storage Bandwidth:</td>
<td>&gt;1 TB/s</td>
</tr>
<tr>
<td>System Memory:</td>
<td>&gt;1.5 Petabytes</td>
</tr>
<tr>
<td>Memory per core:</td>
<td>4 GB</td>
</tr>
<tr>
<td>Gemini Interconnect Topology:</td>
<td>3D Torus</td>
</tr>
<tr>
<td>Usable Storage:</td>
<td>&gt;25 Petabytes</td>
</tr>
<tr>
<td>Peak performance:</td>
<td>&gt;11.5 Petaflops</td>
</tr>
<tr>
<td>Number of AMD processors:</td>
<td>&gt;49,000</td>
</tr>
<tr>
<td>Number of AMD x86 cores:</td>
<td>&gt;380,000</td>
</tr>
<tr>
<td>Number of NVIDIA GPUs:</td>
<td>&gt;3,000</td>
</tr>
</tbody>
</table>
● Cray XE6
  ● 402 TF
  ● 1496 nodes => 47872 IL 16 cores
  ● Gemini Interconnect

● Cray XK6
  ● 176 nodes, each equipped with
    ● one 16-core AMD Opteron CPU, 32 GB DDR3 memory
    ● one NVIDIA Tesla X2090 GPU with 6 GB of GDDR5 memory
Cray Vision for Accelerated Computing

- Provide a tightly coupled programming environment with compilers, libraries, and tools that will hide the complexity of the system
  - Focus on integration and differentiation
  - Target ease of use with extended functionality and increased automation
- Ease of use will be possible with
  - Compiler making it feasible for users to write applications in Fortran, C, C++
  - Runtime to allow for nested parallel structures and calling of routines on the accelerator
  - Tools to help users port and optimize for accelerators
  - Auto-tuned scientific libraries
Beware of fantastic speedup claims

Hence we compare the performance observed on a „standard“ Cray XE6 node against an „accelerated“ Cray XK6 node.

Corollary:

Use single precision on the GPU but double precision on the CPU. This will cut on the effective bandwidths, cache size, and peak performance of the latter and let the former shine.

Georg Hager: Thirteen modern ways to fool the masses with performance results on parallel computers
Erlangen Regional Computing Center (RRZE)
source: http://www10.informatik.uni-erlangen.de/Misc/EIHECS6/Hager.pdf
### Accelerating Computational Science Symposium 2012

<table>
<thead>
<tr>
<th>Code / testcase</th>
<th>nodes</th>
<th>Unit</th>
<th>XE6</th>
<th>XK6 CPU only</th>
<th>XK6 CPU + GPU</th>
<th>XK6 vs XE6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP2K: DBCSR matrix multiply</td>
<td>256</td>
<td>GF/node</td>
<td>76</td>
<td>38</td>
<td>114</td>
<td>1.5</td>
</tr>
<tr>
<td>CP2K: H2O 6912</td>
<td>576</td>
<td>secs/iter</td>
<td>1965</td>
<td>924</td>
<td></td>
<td>1.1</td>
</tr>
<tr>
<td>S3D: 1200³, 52 spc, size 62³, 7200 nd</td>
<td></td>
<td>time/tstep</td>
<td>6.0</td>
<td>8.4</td>
<td>5.6</td>
<td>1.1</td>
</tr>
<tr>
<td>S3D: 1200³, 52 spc, size 48³, 18000 nd</td>
<td></td>
<td>time/tstep</td>
<td>2.78</td>
<td>3.90</td>
<td>2.58</td>
<td>1.1</td>
</tr>
<tr>
<td>GTC: kernel loop</td>
<td>32</td>
<td>secs</td>
<td>448</td>
<td></td>
<td>280</td>
<td>1.6</td>
</tr>
<tr>
<td>SPECFEM3D: 110M Cells</td>
<td>896</td>
<td>GF/node</td>
<td>57.6</td>
<td>30.6</td>
<td>144</td>
<td>2.5</td>
</tr>
<tr>
<td>CAM-SE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.7</td>
</tr>
<tr>
<td>NAMD</td>
<td>768</td>
<td>ms/tstep</td>
<td>106</td>
<td>197</td>
<td>77</td>
<td>1.4</td>
</tr>
<tr>
<td>QMC: DMC sim graphite 256 electr.</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.7</td>
</tr>
</tbody>
</table>

Open Source Molecular Dynamics code (http://www.cp2k.org/)
- Initial Cray XK6 results presented by Joost VandeVondele at Accelerating Computational Science Symposium 2012
- Initial porting to GPU of some library routines: FFT and DBCSR
- DBCSR implements a distributed blocked Cannon’s matrix multiply algorithm on sparse matrices
- DBCSR two stage approach
  - Index building: on CPU
  - Computation: on GPU
- The GPU kernel computes a batch of small matrix block products
  - CUBLAS batched DGEMM reaches ~50 GF/s for a 23x23 block
  - A hand-optimized CUDA kernel has been developed for size 23x23, and achieves 170 GF/s
DBCSR and CP2K performance on Cray XK6

- DBCSR matrix-matrix multiplication benchmark on 256 nodes
  - random [0..1] matrix, quasi-realistic fill-in pattern
  - matrix dimension 159000; 50% occupation
- CP2K benchmark: 6912 H2O molecules on 576 nodes
  - Same matrix dim 159000, ~60 matrix multiplications / iter

<table>
<thead>
<tr>
<th></th>
<th>XE6</th>
<th>XK6 CPU only</th>
<th>XK6 CPU + GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCSR</td>
<td>76 GF/node</td>
<td>38 GF/node</td>
<td>114 GF/node</td>
</tr>
<tr>
<td>CP2K</td>
<td></td>
<td>1965 secs/iter</td>
<td>924 secs/iter</td>
</tr>
</tbody>
</table>

- CP2K overall gain is 1.1x on XK6 compared to XE6 (full nodes)
  - CP2K gets a 2.1x improvement due to GPU on XK6 nodes
  - DBCSR gets a 3x improvement due to GPU on XK6 nodes
  - MPI bandwidth limits CP2K performance on this testcase
- University of Edinburgh Lattice Boltzmann research code used to simulate fluid mixtures and colloids
- C/MPI, ported to GPUs using CUDA

- Ludwig Lattice Boltzmann code re-written in CUDA
  - Reordered all the data structures (structs of arrays)
  - Pack halos on the GPU
  - Streams to overlap compute
  - PCIe comms, MPI halo swaps
- 10 cabinets of Cray XK6
  - 936 GPUs (nodes)
Ludwig Weak Scaling

- 10 cabinets of Cray XK6
  - 936 GPUs (nodes)
- Only 4% deviation from perfect scaling between 8 and 936 GPUs
- Application sustaining 40+ Tflop/s
  - and still scaling...
- Strong scaling also very good, but physicists want to simulate larger systems
Himeno Benchmark

- Parallel 3D Poisson equation solver benchmark
  - iterative loop evaluating 19-point stencil
  - available from [http://accc.riken.jp/HPC_e/himenobmt_e.html](http://accc.riken.jp/HPC_e/himenobmt_e.html)
- Co-Array Fortran version of code
  - ~600 lines of Fortran
  - Fully ported to accelerators using 27 directive pairs
- Simulated XL configuration:
  - 1024 x 512 x 512 global volume
  - Strong scaling
    - Expect halo exchanges to become significant
    - Use asynchronous GPU data transfers and kernel launches to help avoid this
Himeno OpenACC Structure

**Blocking**

1. CPU
2. calculate wrk2 from p
3. pack halos from wrk2
4. update host(halos)
5. put halos over network
6. cosum(wgosu)
7. CAF barrier
8. copy wrk2 bulk to p
9. update device(halos)
10. unpack halos into p
11. next iteration

**Asynchronous**

1. CPU
2. calculate wrk2 from p
3. pack halos from wrk2
4. update host(halos)
5. put halos over network
6. cosum(wgosu)
7. CAF barrier
8. copy wrk2 bulk to p
9. update device(halos)
10. unpack halos into p
11. next iteration
Himeno Performance

![Graph showing performance (Tflops) against number of nodes for asynchronous, blocking, and CPU operations.](image)
Accelerated computing has a potential of >2x performance gain on a per node basis compared to “standard” compute node
  • This ratio is expected to continue
  • Processors will improve, so will GPUs

Power efficiency
  • Running HPL, a Cray XK6 node consumes ~10% more power than a Cray XE6 node

Current processor / GPU designs are a stepping stone to fused architectures
  • Hence, code portability is important (OpenACC)
  • Operating at scale is required to study and understand the implications
  • Steve Scott: "Computers aren't getting faster, they're only getting wider."
    [Link](http://www.hpcwire.com/hpcwire/2012-04-03/nvidia_pokes_holes_in_intel_s_manycore_story.html?page=2)

Course 2012-Cray XK at HLRS, Stuttgart
  • OpenACC Programming for Parallel Accelerated Supercomputers
  • Monday, May 14, 9:00 - Tuesday, May 15, 16:30
  • [Link](https://fs.hlrs.de/projects/par/events/2012/parallel_prog_2012/XK.html)
Thank You!