Linpack Evaluation on a Supercomputer with Heterogeneous Accelerators

Toshio Endo, Akira Nukada
Satoshi Matsuoka, Naoya Maruyama
Tokyo Institute of Technology, Japan

IPDPS 2010, Atlanta
In HPC systems, power consumption has been/will remain a major concern.

GPU and Accelerators are promising for their excellent Flops/Watt ratio.

<table>
<thead>
<tr>
<th></th>
<th>ClearSpeed X620</th>
<th>NVidia GeForce GTX285</th>
<th>ATI Radeon HD4870</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed (SP)</td>
<td>1063GFlops</td>
<td>1200GFlops</td>
<td></td>
</tr>
<tr>
<td>Speed (DP)</td>
<td>80GFlops</td>
<td>88GFlops</td>
<td>240GFlops</td>
</tr>
<tr>
<td>Memory BW</td>
<td>6.4GB/s</td>
<td>159GB/s</td>
<td>115GB/s</td>
</tr>
<tr>
<td>Power</td>
<td>25W</td>
<td>183W</td>
<td>160W</td>
</tr>
</tbody>
</table>
Heterogeneous Systems

Heterogeneous architectures that combines general purpose CPUs and accelerators will be attractive for

- Generality by general purpose CPUs
  - Typically x86/x86-64 CPUs

- Higher Flops/Watt ratio by accelerators
  - GPUs, Cell processor, ClearSpeed...

Example:

- LANL RoadRunner: 1.4PF with 12240 PowerXCell8i
- NUDT Tianhe-1: 1.2PF with 5120 Radeon HD 4870
- TokyoTech TSUBAME: 160TF with 680 Tesla S1070 GPUs+648 ClearSpeed
Our Contribution

- Demonstrated scalability of a heterogeneous system, TSUBAME
- A Linpack implementation that uses cooperatively:
  - 10,368 Opteron cores
  - 612 Tesla GPUs
  - 648 ClearSpeed accelerators
  - 640 Xeon
- A different strategy than on Roadrunner or Tianhe-1 is required
- 87.01TFlops $\rightarrow$ #56 in Top500 ranking
LANL RoadRunner (2008)

The largest heterogeneous system
The first PetaFlops machine in the world!

- 6120 dual-core Opterons and 12240 PowerXCell 8i
- IBM blades

Peak performance is 1.4PFlops
- >90% comes from Cell

#2 in Top500 ranking
Linpack performance is 1.042PFlops
Tokyo-Tech TSUBAME Supercomputer

Tokyo-Tech Supercomputer and Ubiquitously Accessible Mass-storage Environment

“TSUBAME” also means “swallow”, the symbol mark of Tokyo-Tech
TSUBAME Basic Data

- 655-node Linux cluster
  - Sun Fire X4600
  - 8 Dual-core Opteron 880 (=16cores) per node
  - 32GB DDR memory per node
  - And Tesla S1070 GPU and ClearSpeed accelerators

- ~1.1MW power consumption, 350 m² footprint

- SUSE Linux Enterprise 10

- Jobs are managed by a batch scheduler
  - A customized version of Sun N1 Grid Engine
  - A production system used by >1,500 users
Accelerators Installed (1): NVIDIA Tesla S1070

- 4GPUs in 1U box
  - 800 watts/box
- Each GPU has:
  - 30 Multi Processors x 8 Stream processors
  - 86GFlops (double prec)
  - 4GB GDDR3 memory
  - 102GB/s memory bandwidth
- Connected with hosts via external PCI-Express cables
  - 2 GPUs hang on a cable
- Programming with CUDA programming language
- 320 out of 655 TSUBAME nodes are connected with 2 GPUs respectively
  - ‘Inter-node’ heterogeneity
Accelerators Installed (2): ClearSpeed X620 Accelerator

- PCI-X board
  - 2 CSX600 x 96 SIMD cores
  - 80GFlops (double prec)
  - 1GB DDR memory
  - 6.4GB/s memory bandwidth
  - 25 watts /board

- Programming with ClearSpeed C^n programming language

- Each TSUBAME node has a board
TSUBAME Node with Hybrid Accelerators

- 8 dual-core Opteron CPUs (16 cores)
- SDR InfiniBand 1GB/s x 2
- ClearSpeed
- Other nodes
- 32GB memory
- 2GPUs of Tesla
- PCI-X 1GB/s
- PCI-e gen1 x8 2GB/s
- SunFire X4600
History of TSUBAME in Top500

<table>
<thead>
<tr>
<th></th>
<th>Jun06</th>
<th>Nov06</th>
<th>Jun07</th>
<th>Nov07</th>
<th>Jun08</th>
<th>Nov08</th>
<th>Jun09</th>
<th>Nov09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linpack Speed</td>
<td>38.18 (TF)</td>
<td>47.38</td>
<td>48.88</td>
<td>56.43</td>
<td>67.70</td>
<td>77.48</td>
<td>87.01</td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>7</td>
<td>9</td>
<td>14</td>
<td>16</td>
<td>24</td>
<td>29</td>
<td>41</td>
<td>56</td>
</tr>
</tbody>
</table>

- The 3rd system as a heterogeneous system
  - From Nov 06 to Nov 07, it was the 1st
- Continuous improvement for 7 times
What is Linpack?

◆ A numerical benchmark used in Top500 supercomputer ranking (www.top500.org)
  • Solves a dense linear equation $Ax = b$ of order $N$
  • A direct solver; total computation cost is $O(N^3)$
  • Users can configure $N$; In TSUBAME, $N \approx 1,000,000$

◆ HPL (High-performance Linpack) by A. Petitet
  • A famous MPI parallel implementation, designed for uniform systems
  • Based on blocked LU-decomposition, with partial pivoting
  • The most time consuming part is matrix-multiplication (DGEMM)
  • Used as a basis of our implementation
HPL Algorithm

LU decomposition of $N \times N$ matrix $A$

for $(k = 0; k < N; k += B)$
   Panel factorization with partial pivoting to obtain $L$
   Broadcast $L$
   Row exchange, and compute $U$
   Update the rest part of matrix
   
   $A' = A' - L \times U$

DGEMM is the most time consuming
Data Decomposition in HPL

- Matrix A is uniformly distributed with 2D block-cyclic distribution among processes.

Matrix distribution on 6 (=2x3) processes

Each process has a “partial-matrix”

\[ A_L' = A_L' - L_L \times U_L \]
Design Issues on Heterogeneous Systems

- **Who computes?**
  - Kernel (DGEMM, DTRSM)
    - Accelerators? Both CPU and accelerators?
  - Non-kernel

- **Where are matrix data placed?**
  - Host memory? Accelerator memory?

  - Strategies depend on system architecture
  - We compare our decision with that on Roadrunner [PPoPP09]
    - More challenging on TSUBAME
Who Computes?

- Non-kernel
  - Only CPUs are used for MPI communication, pivoting...

- Kernel functions
  - On Roadrunner, Cells contribute 96% of performance
    - Ratio of CPUs is 4%
  - Only Cells are used

- On TSUBAME, CPUs contribute 35%
  - Omitting any type of processors heavily degrades performance

⇒ All of CPUs, GPUs, ClearSpeed are used
Where are matrix data placed? (1)

A RR node

A TSUBAME node

Host mem : Device mem
16GB = 4GB x 4

Host mem : Device mem
32GB > 4GBx2 + 1GB
Where are matrix data placed? (2)

- In Linpack, the matrix size should be larger to gain speed in Flops
  - ⇒ it should be as large as host memory

- On RR,
  - (1) Device memory = Host memory
  - (2) Kernel computation is done only by Cells
  ⇒ Matrix data are **on Cell device memory**

- On TSUBAME,
  - Device memory < Host memory
  ⇒ Matrix data are **usually on host memory**
Executing Kernel Functions on Accelerators

- Matrix data is on host memory, when DGEMM function is called
- Pipelined DGEMM execution:
  1. A part of input data is moved from host to device
  2. Computes DGEMM on accelerators
  3. The results are moved back to host, then repeats for next partial matrix

More frequent and larger amount of PCI-e/PCI-X communications are required than on RR
Challenging Issues on TSUBAME

- **Intra-node heterogeneity:**
  - CPU/GPU/ClearSpeed are used for kernel
  - On RR, using only Cell is sufficient

- **Inter-node heterogeneity:**
  - Half the nodes have GPUs, while others don’t
  - On RR, nodes are uniform

- **Frequent PCI-e/PCI-X communication:**
  - The whole input/output is moved via PCI
  - On RR, matrix data always resides in Cell device memory

How can we run HPL, originally designed for uniform systems, efficiently?
Coping with intra-node Heterogeneity

- We ‘virtualize’ heterogeneous processors at BLAS layer
- Processors are providers of DGEMM performance
- **We control mapping between processes and processors**
  - An MPI process divides its own sub-matrix with a proper ratio and throws DGEMM tasks to CPUs and accelerators
  - All processes should be mapped with processors of similar performance

Example of mapping during DGEMM
Coping with **Inter-node Heterogeneity**

- We control the number of processes among nodes
  - cf. CHARM++, AMPI from UIUC

- We can keep kernel workload of each process uniform (good for HPL), while maintaining heterogeneity
Mapping between Processes and Processors (1)

- When processing non-kernels
  - Panel factorization, MPI communication etc.

Node w/ Tesla:
4 procs/node
16 Opteron cores

Node w/o Tesla:
2 procs/node
16 Opteron cores
Mapping between Processes and Processors (2)

- When processing DGEMM kernels
  - Each process uses several cores and accelerators

Node w/ Tesla: 4procs/node
- 16 Opteron cores
- Clear Speed
- Tesla
- Tesla
- Some cores are dedicated for PCI comm

Node w/o Tesla: 2procs/node
- 16 Opteron cores
- Clear Speed
Coping with PCI Communication overhead

- Since matrix data is allocated on host memory, kernel performance heavily depends on the matrix size.

- For the sizes in the figure,
  - Computation: $O(M'N'B)$
  - PCI-Communication: $O(M'N' + M'B + N'B)$

- To reduce effects of PCI communication, $M'$, $N'$, $B$ should be large enough.

- In Linpack, we should keep the block size $B$ large enough.

We decided to use $B=1152$, which achieves 241GFlops per node.
Evaluation Conditions

- 648 TSUBAME nodes
  - 312 nodes are connected with Tesla GPUs → 624 GPUs are used in total
- 80 8-core Xeon nodes
- Modified HPL + Voltaire MPI + GOTO BLAS + CSXL BLAS + NUBLAS
  - NUBLAS is our own DGEMM kernel for Tesla GPUs
- Total number of processes is 2000
  - 2000 = 312 nodes x 4 procs + 336 x 2 + 80 x 1
  - Process grid (P x Q) = (40 x 50)
- Matrix size N = 1,059,839, block size B = 1,152
## Evaluation Result

<table>
<thead>
<tr>
<th>T/V</th>
<th>N</th>
<th>NB</th>
<th>P</th>
<th>Q</th>
<th>Time</th>
<th>Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>WC10R2R4</td>
<td>1059839</td>
<td>1152</td>
<td>40</td>
<td>50</td>
<td>9121.18</td>
<td>8.701e+04</td>
</tr>
</tbody>
</table>

\[
\|Ax-b\|_\infty / (\epsilon (\|A\|_\infty \|x\|_\infty + \|b\|_\infty )N) = 0.0119654 \ldots \text{ PASSED}
\]

- 87.01 TFlops is achieved
  - #56 in the Top500
  - #3 performance as a heterogeneous supercomputer
- 2.6 hours
## Discussion on Efficiency

<table>
<thead>
<tr>
<th></th>
<th>Peak (TFlops)</th>
<th>Linpack (TFlops)</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoadRunner</td>
<td>1376</td>
<td>1042</td>
<td>76%</td>
</tr>
<tr>
<td>Tianhe-1</td>
<td>1206</td>
<td>563</td>
<td>47%</td>
</tr>
<tr>
<td>TSUBAME</td>
<td>163.2</td>
<td>87.01</td>
<td>53%</td>
</tr>
<tr>
<td>Opteron only TSUBAME</td>
<td>49.87</td>
<td>38.18</td>
<td>77%</td>
</tr>
</tbody>
</table>

- Why is the efficiency lower?
  - PCI overhead? Inter-node heterogeneity?
  - We will discuss it step by step
Discussion (1/5): Overhead of Core-wise DGEMM

- DGEMM performance is measured on each type of CPU core/accelerators, and totaled

\[ [A] -11\% \text{ TSUBAME} \]

- PCI overhead is not included
- We observe 11% overhead
Discussion (2/5): Overhead of Node-wise DGEMM

- DGEMM performance is measured on each type of node, and totaled

- Opteron part is suffered from the existence of cores dedicated for PCI communication

- Opteron-only and RR are almost free from PCI comm
Discussion (3/5): Overhead by Inter-node Heterogeneity

- DGEMM performance of each node type deviates from 4:2:1 little \(\rightarrow\) bottlenecked by the slowest processes

- This overhead is peculiar to TSUBAME
Discussion (4/5): Overhead Caused by DGEMM Problem Size

- In Linpack, the problem size of DGEMM kernel gets smaller as iterations proceed. We simulated changes of kernel size and measured the performance.

![Graph showing performance changes with different sizes of DGEMM kernel for Tsubame and Opteron-only systems.](image)
Discussion (5/5):
Other Overhead

- MPI communication overhead
- Computations other than kernels, including panel factorization

![Graph showing performance comparisons between TSUBAME and Opteron-only systems.](image)
Summary

- Heterogeneous supercomputers are scalable
  - 87TFlops Linpack performance is achieved on TSUBAME with >600 GPUs, >600 ClearSpeeds, >10000 Opteron cores
  - We have discussed on overheads peculiar to heterogeneous systems
    - Some are peculiar to TSUBAME

- For better performance, efficient CUDA kernels are important, but we need more!
  - Analysis of application and architecture
  - Algorithm design
  - Considering overhead of PCI comm, MPI comm
Future Plan

- A new system TSUBAME 2 will be introduced in this autumn
  - 2.4PFlops peak with ~4000 Fermi GPUs
    - Exceeds RoadRunner and Tianhe-1
  - Linpack, HPCC and other applications will be evaluated