MPI Forum May 2012 Meeting @ Tsukuba:
Fujitsu extensions of Open MPI for K computer

Takahiro Kawashima, Tomoya Adachi, Shinji Sumimoto
Fujitsu Limited
Outline of This Talk

- RIKEN and Fujitsu are jointly developing K computer at RIKEN AICS, Kobe, Japan
  - Public operation will start on autumn of 2012

Outline of This Talk

- K computer and its interconnect Tofu
- Our challenges to MPI implementation
- Performance evaluation
K computer

- 88K nodes connected by Tofu interconnect
- Each node equips:
  - one SPARC64 VIIIfx processor (8 cores, 2.0GHz)
  - one Tofu interconnect controller
- We got:
  - No.1 of Nov. 2011 TOP500 (10.51PFLOPS with 93.2% efficiency)
  - No.1 in Four benchmarks at 2011 HPC Challenge Awards
  - Gordon Bell Prize for Peak Performance at SC11

Our Goals
- Challenging to Realize World’s Top 1 Performance
- Keeping Stable System Operation over 80K Node System
Software View of Tofu Interconnect

- 3D Torus View for User’s Job
  - Combining 6D torus/mesh axes to 3 rings

- 4 RDMA Engines (NIC) per Node
  - RDMA Interface: Put and Get
  - Minimum latency to neighbor nodes: < 1μs
  - Unidirectional bandwidth per engine: 5GB/s
  (Total bidirectional bandwidth: 40GB/s)
Fujitsu MPI and Our Challenges

- Fujitsu MPI for the K computer: Based on Open MPI with
  - Original Process Management and High Performance Point-to-Point Communication for Tofu Interconnect
  - Enhanced Collective Communication

Our Challenges

- High Performance
  - High Bandwidth
  - Low Latency
  - Effective Collective Communication
  - Simple Tofu RDMA Interface

- Reduced Memory Consumption
  - 12KiB/process communication buffer needs 1GiB for 88K processes!
  - Trade-off with communication performance

- Usability
  - Hide complex 6D torus/mesh view and provide arbitrary-sized 1D/2D/3D torus view

Topics of This Talk
Low Latency: Issues and Our Solution

**Issues**

- Software overhead should be $<< 1\mu s$ for short message PingPong in order to utilize Tofu hardware latency ($< 1\mu s$).
- Rich functionalities of MPI (communicator, datatype, blocking/non-blocking, ...) involve software overhead.

**Our Solution: Tofu LLP (Tofu Low Latency Path)**

- Optimized path dedicated to blocking send of short & contiguous message.
- In the Open MPI terms, we created new “LLP framework” and “tofu LLP component”.
Low Latency: Open MPI Stacks of K computer

- Tofu LLP bypasses creating a request object, packing data for complex datatype, creating a BTL descriptor, ...
- Falls back to normal path if condition doesn’t meet.

```
Point-to-Point

Extension

Collective

MPI Interface Layer

ob1 PML

tofu LLP

r2 BML

tofu BTL

Common Functions for
tofu BTL, tofu LLP, tuned COLL, tbi COLL

Shortcut Pass for Low Latency
Communication (send, rsend, bsend)

Common Functions for
tofu BTL, tofu LLP, tuned COLL, tbi COLL

Tofu Interconnect

Tofu Library
```
Software overhead is relatively small (though we are not satisfied yet)

~0.3μs
Effective Collective Communication: Issues and Our Solution

- Issues of Collective Communication Implementation of Open MPI:
  - Not topology-aware, but rank-based
  - Not multi-NIC-aware
    - Frequent message collision (results in low bandwidth)

- Send/Receive model
  - Large software overhead
    (e.g. rendezvous on each pipeline segment)

- Our Solution:
  - Using topology- and multi-NIC-aware algorithms with one-sided RDMA-based communication
Effective Collective Communication: Topology- and multi-NIC-aware algorithms

- **Collision-freeness**: Communicating only with neighbor nodes in a pipelined or bucket manner
- **Multi-NIC-awareness**: Dividing messages into multiple parts and transferring via different paths

**Examples**

- **Bcast (pipeline)**

  phase 1  
  phase 2  
  phase 3  
  phase 4

- **Allgather (bucket)**

  phase 1  
  phase 2  
  phase 3
Effective Collective Communication: Bcast Bandwidth

Bcast Bandwidth (48x6x32)

- Algorithm for Tofu (for long message)
- Algorithm for Tofu (for short message)
- Algorithms in Open MPI

Multi-NIC-aware collision-free tree algorithm
Multi-NIC-aware binary tree algorithm
11x faster
Effective Collective Communication: Allgather Bandwidth

Allgather Bandwidth (48x6x32)

- Algorithm for Tofu
- Algorithms in Open MPI

Multi-NIC-aware collision-free bucket algorithm

4x faster
Simple Tofu RDMA Interface: Issues and Our Solution

**Issues:**
- Rich and portable functionalities of MPI point-to-point communication and one-sided communication involve various overheads.
- Users cannot fully control communication by MPI calls.
  - True RDMA
  - Multi-NIC
  - Specific Hardware Feature
  - Communication Path in Torus/Mesh
- Applications may get more performance by low-level hardware control.

**Our Solution: Simple Tofu Specific RDMA Interface**
- Fujitsu-specific API (FJMPI_Rdma_prefix)
- Low-level RDMA communication
  - Able to control NIC, communication path, and memory registration directly.
  - Able to use Tofu specific feature (remote process notification on RDMA)
- Simplified API; only RDMA (Put/Get), no communicators, no datatypes.
- Abstract API; can be implemented for widely-used InfiniBand.

Application-specific tuning with extended RDMA interface in real application is expected.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Achieved (GUPS)</th>
<th>System</th>
<th>Affiliation</th>
<th>Peak (TFLOPS)</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>121</td>
<td>K computer (1/5 scale)</td>
<td>RIKEN AICS</td>
<td>2359</td>
<td>18432</td>
</tr>
<tr>
<td>2</td>
<td>117</td>
<td>IBM BG/P</td>
<td>LLNL</td>
<td>446</td>
<td>32768</td>
</tr>
<tr>
<td>3</td>
<td>103</td>
<td>IBM BG/P</td>
<td>ANL</td>
<td>557</td>
<td>32768</td>
</tr>
<tr>
<td>4</td>
<td>38</td>
<td>Cray XT5</td>
<td>ORNL</td>
<td>2320</td>
<td>111556</td>
</tr>
</tbody>
</table>

http://www.hpcchallenge.org/
Summary

- MPI scaled to 88K nodes and achieved LINPACK 10PFLOPS on K computer.
- Fujitsu MPI enhanced implementation of Open MPI to utilize the performance of Tofu interconnect.
- Application-specific tuning with extended interface shows good performance on Global RandomAccess.

- We thank Open MPI development team very much for providing very stable MPI software.
- We would like to make some contribution to Open MPI community. (undergoing)