Kernel-Level Support for Scalable Intra-Node Collective Communications

Hyun-Wook Jin and Joong-Yeon Cho
System Software Laboratory
Dept. of Computer Science and Engineering
Konkuk University
jinh@konkuk.ac.kr
Contents

• MPI intra-node communication

• Intra-node collective communication
  – MPI_Bast()
  – MPI_Gather()

• Conclusions and future work
# Multi/Many-Core Processors

<table>
<thead>
<tr>
<th>Xeon 5100 Series (Woodcrest)</th>
<th>Xeon 5500 Series (Gainestown)</th>
<th>Xeon 5600 Series (Westmere-EP)</th>
<th>Xeon E5-2600 Series (Sandy Bridge-EP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xeon E5-2600 v2 Series (Ivy Bridge-EP)</th>
<th>Xeon E5-2600 v3 Series (Haswell-EP)</th>
<th>Xeon E7 v4 Family (Broadwell)</th>
<th>Xeon Platinum Series (Skylake)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>18</td>
<td>24</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xeon Phi X100 Series (Knights Corner)</th>
<th>Xeon Phi 7200 Series (Knights Landing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>72</td>
</tr>
</tbody>
</table>
MPI Intra-Node Communication

• **Loopback**
  - NIC provides a loopback path
  - Two DMAs

• **Shared memory**
  - Communicate through a memory area shared between MPI processes
  - Two data copies
MPI Intra-Node Communication

• **Memory mapping**
  – Directly move a message from source to destination buffer by means of kernel-level support
  – Single data copy
    • Beneficial for large messages
Kernel-Level Support for MMMapping

• LiMIC2
  – Opened the era of one-copy intra-node communication
  – LiMIC2-0.5 was publicly released with MVAPICH2-1.4RC1 (Jun. 2009)
  – LiMIC2-0.5.6 is being released with the latest MVAPICH2
    • mvapich2-src]$ ./configure --with-limic2 [omit other configure options]
    • mvapich2-src]$ mpirun_rsh -np 4 -hostfile ~/hosts MV2_SMP_USE_LIMIC2=1 [path to application]
Kernel-Level Support for MMMapping

• CMA
  – In-kernel implementation + New system calls
    – Default intra-node communication channel for large messages in MVAPICH2

• XPMEM
  – Supports memory mapping to user-level address space
Intra-Node Collective Communication

• **MPI_Bcast()**
  – Broadcasts a message from the root to all other processes of the communicator
    • One-to-Many: Root -> Other processes
  – MVAPICH2 (version 2.3) uses the collective-aware shared memory

• **MPI_Gather()**
  – Gathers together values from a group of processes
    • Many-to-One: All processes -> Root
  – MVAPICH2 (version 2.3) uses the kernel-level support (either CMA or LiMIC2) for large messages
MPI_BCAST
MPI_Bcast() in MVAPICH2 (v.2.3)

1. Copies 8KB data blocks to the shared memory (by the root process)

2. Copies 8KB data blocks to the destination buffer (by the other processes)
How bad is LiMIC2 for MPI_Bcast()?

- Experimentally applied LiMIC2 instead of shared memory
  - Shows higher latency up to 548%
Why not to use LiMIC2 in MPI_Bcast()? 

• What we expected...

![Diagram showing MPI_Bcast process]
Why not to use LiMIC2 in MPI_Bcast()? 

• What actually happened...

```
MPI_Bcast(
Send Descriptor
Memory Mapping (get_user_pages())
Data Copy
Memory Unmapping
return
```
MPI_Bcast() with LiMIC2-overlap

- The root performs memory mapping and the others reuse (share) the mapped area.
Preliminary Measurement Results

- **20-core system**
  - Intel Xeon Haswell Deca-Core x 2
  - LiMIC2-overlap reduces the latency up to 68%

- **120-core system**
  - Intel Xeon IvyBridge 15-Core x 8
  - LiMIC2-overlap reduces the latency up to 84%
What’s going on in MVAPICH2 Bcast?

1. Copies 8KB data blocks to the shared memory (by the root process)

2. Copies 8KB data blocks to the destination buffer (by the other processes)
What’s going on in MVAPICH2 Bcast?

- **Collective-aware shared memory**
- **LiMIC2-overlap**

* Message size: 256KB
* Some profiling overheads are included
What’s going on in MVAPICH2 Bcast?

- Data copy operations are not overlapped as much as expected
MPI_GATHER
MPI\_Gather() in MVAPICH2 (v.2.3)

1. Allocates an intermediate buffer
2. Moves messages to the intermediate buffer via point-to-point communication
3. Copies the gathered messages to the destination buffer

Root Process (P0) → Intermediate Buffer → Destination Buffer

Process P1
Source Buffer

Process P2
Source Buffer

... Process P(N-1)
Source Buffer
Is it OK to use LiMIC2 in MPI_Gather()?

* Message size: 256KB
* Some profiling overheads are included
Is it OK to use LiMIC2 in MPI_Gather()?
Is it OK to use LiMIC2 in MPI_Gather()?
Why not to use LiMIC2 in MPI_Gather()?
MPI_Gather() with LiMIC2-overlap
Preliminary Measurement Results

- **20-core system**
  - LiMIC2-overlap reduces the latency up to 88%

- **120-core system**
  - LiMIC2-overlap reduces the latency up to 50%
  - Different algorithms matter (e.g., binomial tree algorithm)
CONCLUSIONS
Concluding Remark

• **Intra-node collective communication**
  – MPI_Bcast()
    • One-to-Many communication
    • Implemented using collective-aware shared memory
  – MPI_Gather()
    • Many-to-One communication
    • Implemented using point-to-point

• **LiMIC2-overlap**
  – New interfaces
    • Memory mapping reuse
    • Flexibility of who can perform data copy
  – 84% improvement for MPI_Bcast()
  – 88% improvement for MPI_Gather()
Ongoing Work

- **Other collectives**
  - **MPI_Scatter()**
    - LiMIC2-overlap reduces the latency up to 78% on the 20-core system
  - **MPI_Allgather()**
    - LiMIC2-overlap reduces the latency up to 38% on the 20-core system
Ongoing Work

• Overlapping between collective communication and computation
Future Work

LiMIC3
ParaMo 2019

- The 1\textsuperscript{st} International Workshop on Parallel Programming Models in High-Performance Cloud
  - Co-located with Euro-Par 2019
  - Date: August 26, 2019
  - Venue: Göttingen, Germany
Thank You!

Ministry of Science and ICT