Kernel-assisted MPI Communication on Multi-core Clusters

Teng Ma

ICL, Knoxville, July-15-2011
Outline

• Motivation
• Kernel-assisted collective comm. (linear KNEM Coll)
• Topology-aware approach
• Dodging algorithm
• Conclusion
Motivation

- Increase node complexity
- Unsatisfied programming models.
Shared-memory approach

- Shared-memory region
- Copy-in/copy-out
- Problems
Related work

• SMARTMAP (Catamount)

• Single-copy large message comm.
  - BIP-SMP, Myricom MX, Qlogic IPath, and Bull MPI

• Kernel-assisted approach
  - KNEM (Kernel-nemesis)
  - LIMIC
KNEM: inter-process single-copy

MPICH2 since v1.1.1
mpich2-Nemesis LMT

Open MPI since v1.5
SM/KNEM BTL.
Outline

Motivation

• Kernel-assisted collective comm. (linear KNEM collective)

Topology-aware approach

Dodging algorithm

Conclusion
Architecture & Operations

- MPI_Bcast, MPI_Gather(v), MPI_Scatter(v),
- MPI_Allgather(v) and MPI_Alltoall(v)

Figure 2. Open MPI collective communication framework
Linear KNEM Bcast

1. root process declares sbuf to KNEM device to get a cookie back.

```c
mca_coll_knem_create_region(buff, count, datatype, knem_module, PROT_READ, &cookie);
```

2. root process distributes this cookie around non-root processes.

3. each non-root process triggers a KNEM copy simultaneously.

```c
mca_coll_knem_inline_copy(buff, count , datatype, knem_module, cookie, 0, 0)
```

4. each non-root process sends an ack back to root process.

5. root process deregisters sbuf from KNEM device
Linear KNEM Scatter & Gather

MPI_Scatter ( void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm )

• The same with Bcast’s 5 steps except:
  • 1) mca_coll_knem_create_region(buff, count*N, datatype, knem_module, PROT_READ, &cookie);
  • 3) mca_coll_knem_inline_copy(buff, count, datatype, knem_module, cookie, rank* count*datatype_size, 0);
Linear KNEM Allgather & Alltoall

Linear KNEM Allgather

= KNEM Gather + Bcast
Experiment: Platform

- Zoot (4-socket with a quad-core Intel Tigerton)
- Saturn (2 sockets with an octo-core Intel Nehalem-EX)
- Dancer (2 sockets with a quad-core Intel Nehalem-EP)
- IG (8 sockets with a six-core AMD Opteron)
Experiment: software

- KNEM version 0.9.2
- IMB-3.2(enable cache_off)
- Open MPI trunk version 1.7a1 and MPICH2-1.3.1
- Always use the same mapping between cores and processes.
- Process number is the same with core number inside each node.
Broadcast & Gather

(a) on Zoot  
(b) on Dancer  
(c) on Saturn  
(d) on IG  

Broadcast  

Gather
Alltoallv & Allgather

(a) on Zoot
(b) on Dancer
(c) on Saturn
(d) on IG

Alltoallv
Allgather
Outline

Motivation

Kernel-assisted collective comm. (linear KNEM collective)

• Topology-aware approach

Dodging algorithm

Conclusion
Kernel-assisted topology-aware hierarchical collective

• Building a collective topology according to underlying hardware, especially memory hierarchies inside nodes. (by HWLoc.)
  ➢ MPI_Bcast( hierarchical tree algorithm)
  ➢ MPI_Allgather(v) ( ring algorithm)

Target
◆ Balance memory accesses across NUMA nodes.
◆ Minimize remote memory accesses As Small As Possibly.
Broadcast on IG (48 cores)
Allgather

![Diagram of Allgather process](Image)

Figure 5. An example of distance-aware Allgather with 8 processes on a quad-socket dual-core node with random binding between processes and cores.

Hierarchical effect and tuning the pipeline size (Bcast, IG, 48 cores)

Figure 4. Performance comparison between linear KNEM Broadcast, hierarchical KNEM without pipeline, and different pipeline sizes in the hierarchical pipelined KNEM Broadcast on the IG platform. Results are normalized to the runtime of hierarchical KNEM Broadcast without pipeline (lower is better).
Bcast on IG(48 cores).
Allgather on IG (48 cores).
CPMD on IG (48 cores)

CPMD’s methan-fd-nosym test

Outline

Motivation
Kernel-assisted collective comm. (linear KNEM collective)
Topology-aware approach
• Dodging algorithm
Conclusion
Broadcast on dancer's 64 cores.

Broadcast BW compare between Open MPI's hierarchy, Tuned and dodging algorithm on dancer's 64 cores.
Scale with core # in nodes (dancer 8 nodes)

Bandwidth of broadcast operations under different collective components by changing core # per node in the communicator.
Reduce on Dancer (64 cores)
Conclusion

• Kernel-assisted approach provides a scalable solution for MPI on multi-core nodes.
• Kernel-assisted approach can cooperate with other algorithms.
• Kernel-assisted approach can be used to dodge intra-node communication overhead of MPI on multi-core clusters.