Hybrid MPI: Efficient Message Passing for Multi-core Systems

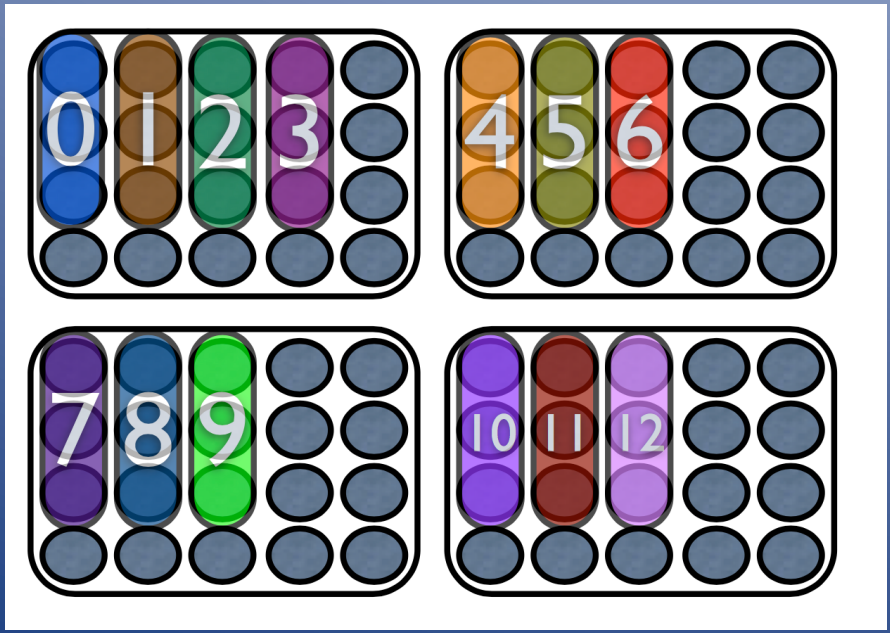
Speaker: Hualiang Li

Department of Electrical Engineering

University of Hawaii, Manoa

Background

- Modern multi-core system
- MPI communication overhead
- OPENMP thread-safety issue




```
#SBATCH -n 13
#SBATCH −c 3
#SBATCH −N 4
```

Outline

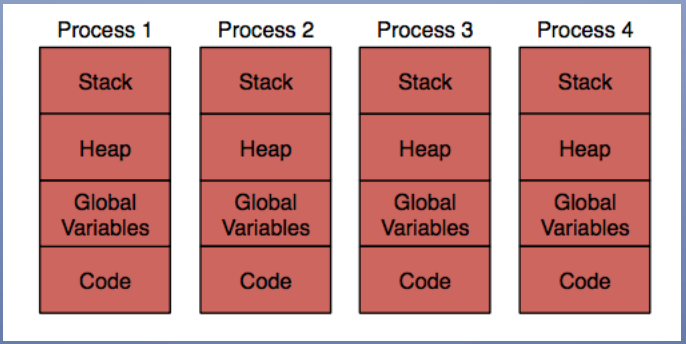
1. Objective of HMPI
2. Current MPI implementation
3. HMPI architecture
4. Communication protocols
5. Result analysis
6. Conclusion
7. Q&A

Objective

- Intra-node communication performance
  - Share memory between ranks
- Portability
  - No extra knowledge needed for MPI user
- Without root access to modify system

Current MPI implementation

1. Process-based MPI



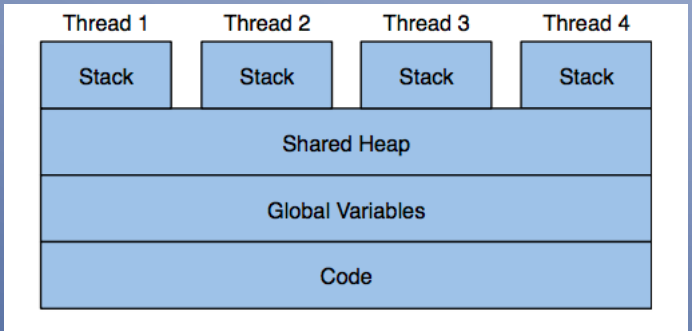
Advantage: No synchronization

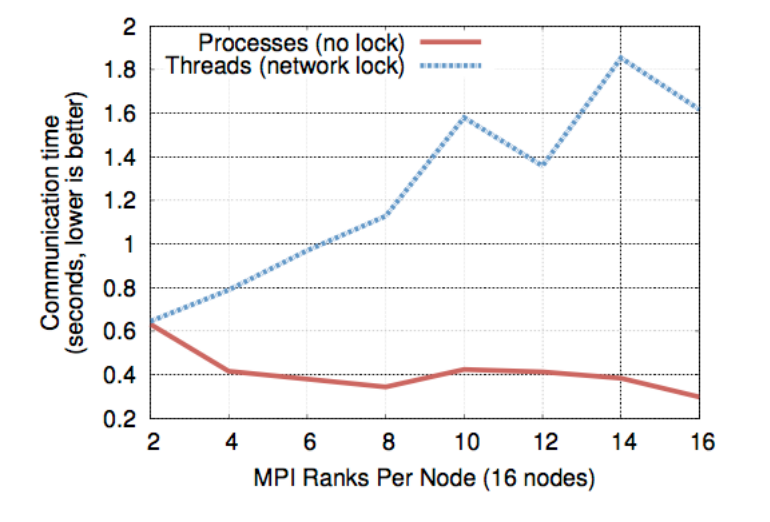
Disadvantage: 1. Two-copy per message (pipeline)

2. Pair-wise memory allocation (memory consuming)

Current MPI implementation

2. Thread-based MPI





Advantage: Shared memory

Disadvantage: Race condition (global variable)

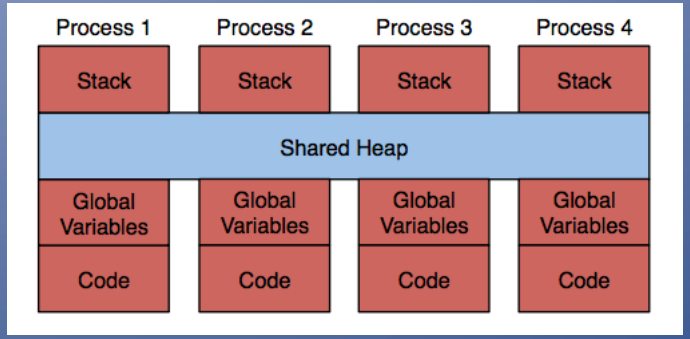
i. Complier transformation tool

ii. Thread-safe API only

HMPI architecture

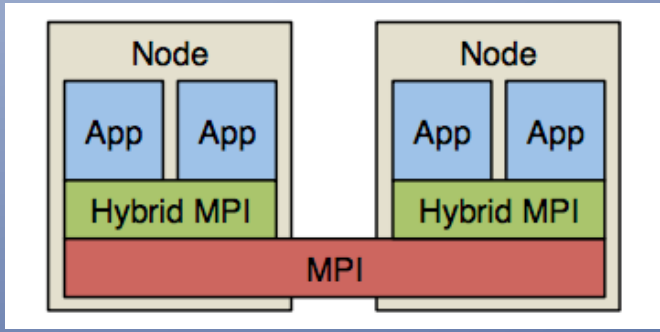
1. Shared memory allocation
   a) Some systems allow memory mapping from other process (require root access)

   b) In HMPI, *mmap* and *sbrk* are used to modify *dlmalloc*
      - virtual memory
      - process owned share memory
      - one 16G physical memory, up to 16G virtual memory could be allocated for each process



HMPI architecture

2. HMPI inheritance



- Hybrid MPI library

  Advantages:

  a) portability: Just link it to application

  b) transparency: No code transformations, or library modifications are needed

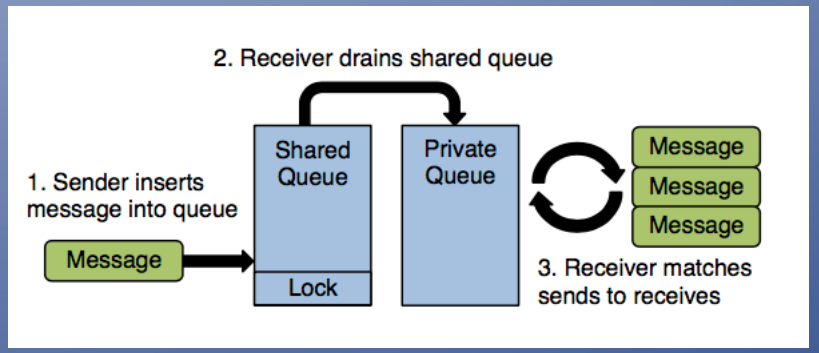
HMPI architecture

3. Message matching

   a) two message queues per receiver, one private, one global.

   b) MCS lock (FIFO)

   c) Drain from global to private in constant time

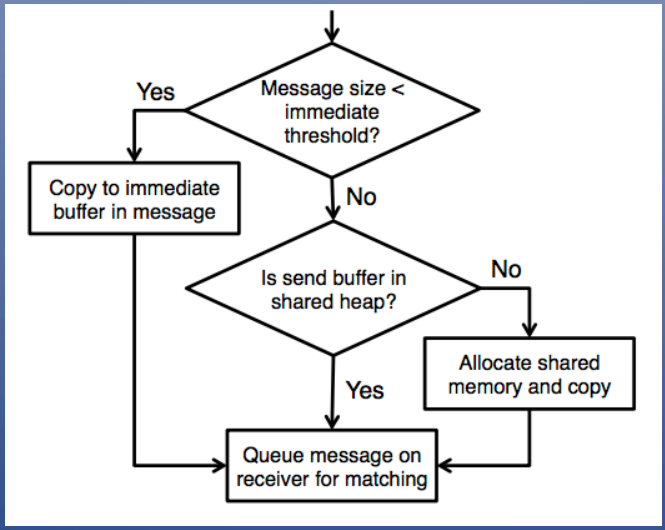


Communication protocols

1. Immediate Transfer Protocol

   memcpy is not always first choice

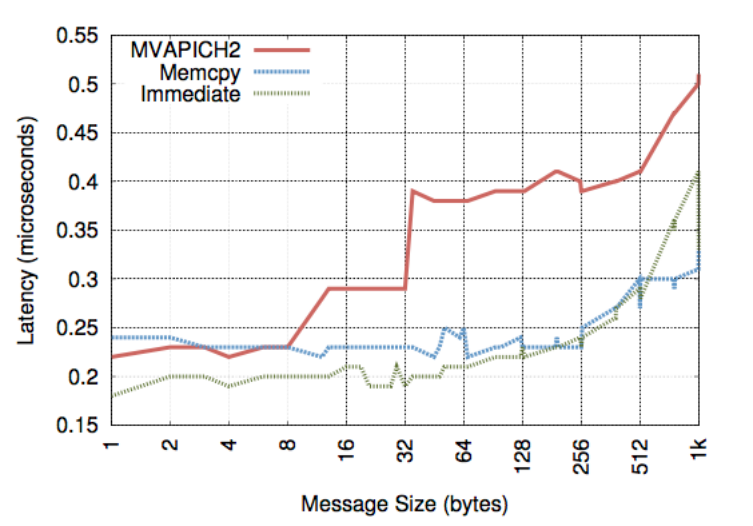
   For small message, use two-copy method



256 bytes threshold

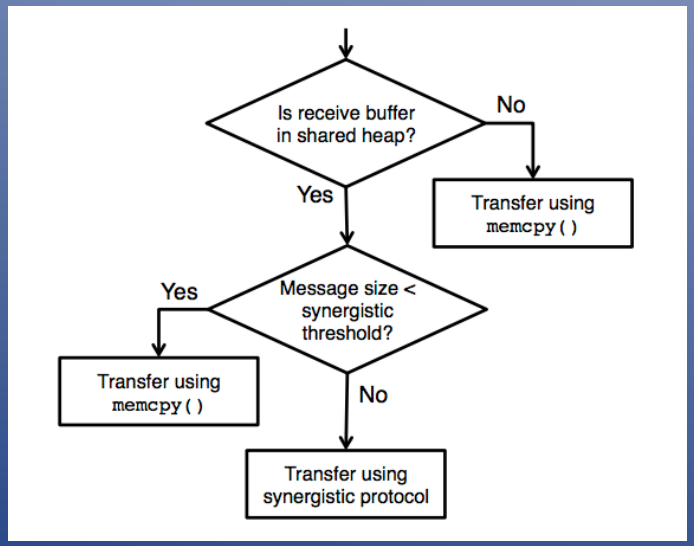
Communication protocols

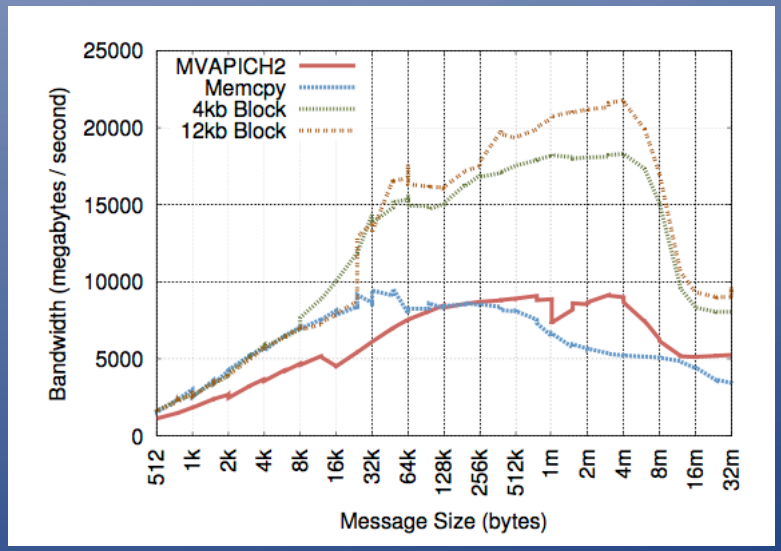
- Sender bring the message into cache
- The time saved by avoiding the cache miss more than makes up for the cost of doing two copies



Communication protocols

2. Synergistic transfer protocol
   a) Bandwidth bottleneck
   b) Receiver can copy entire message without sender





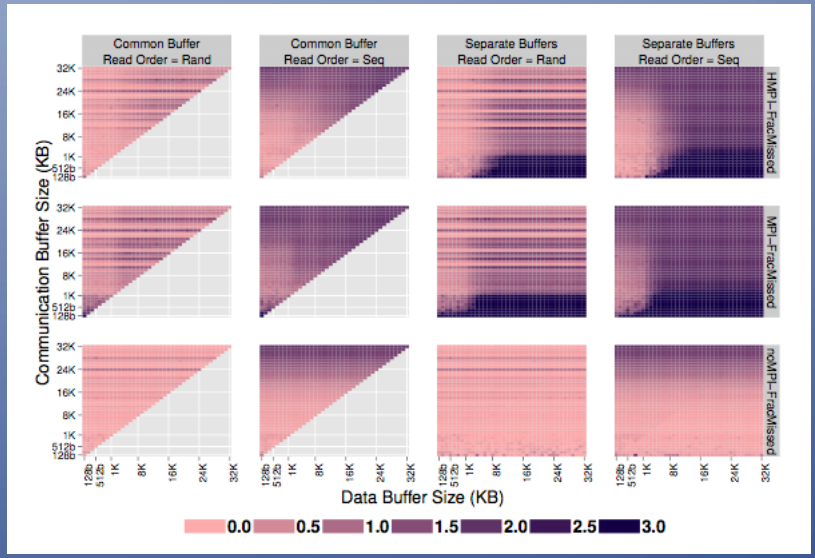
Result analysis

I. Communication analysis

a) Lowest cache miss if no communication

b) HMPI has lower cache miss than MPI

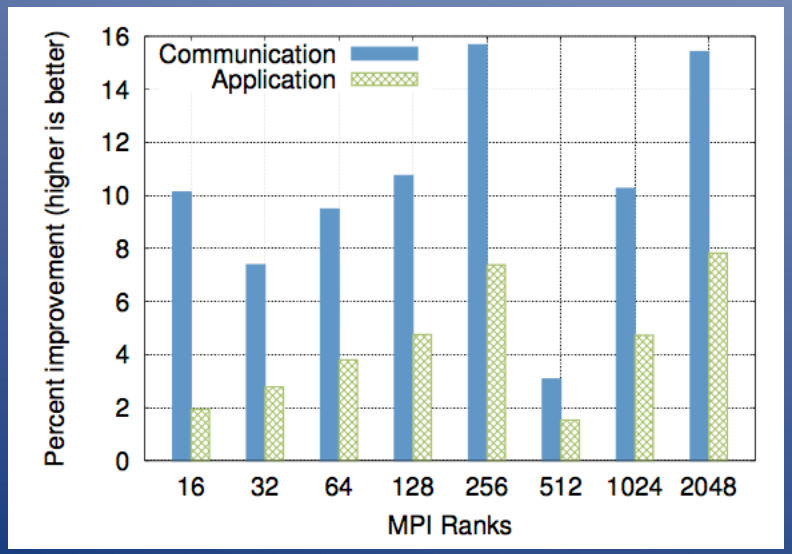
c) Random reading has a better cache hit means cache replacement algorithm not good enough.



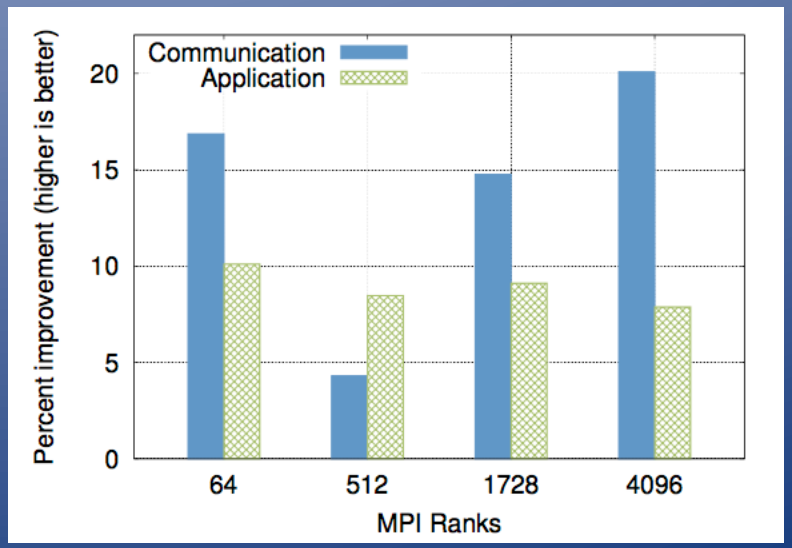
Result analysis

II. Application analysis

1. For MiniMD, communicationg speedups of 3.1-15.7%, resulting in total application time improvements of 1.5-7.9%
2. For LULESH, communication time speedups of 11-46.1% and application time speedups of 14.1% - 19.5%



MiniMD performance



LULESH performance

Conclusion

Approach

- Exploit shared memory hardware in MPI programs that are written for distributed memory systems.
- Utilize a layered model for implementing our optimizations.
- Use above two mechanisms to develop HMPI, a fast layered MPI library that optimizes hybrid shared memory communication

Result

- less on-node communication overheads
- Integrates transparently into legacy applications

Reference

- [1] The OpenCL specification version 1.0, 2009.
  [2] A. Friedley, T. Hoefler, G. Bronevetsky, C.-C. Ma, and A. Lumsdaine. Ownership passing: E cient distributed memory programming on multi-core systems. February 2013. PPoPP 2013.
- [3] E. Gabriel, M. Resch, and R. RA¨ijhle. Implementing mpi with optimized algorithms for metacomputing, 1999.
- [4] B. Goglin and S. Moreaud. KNEM: a Generic and Scalable Kernel-Assisted Intra-node MPI Communication Framework. Journal of Parallel and Distributed Computing, 73(2):176–188, Feb. 2013. KNEM.
- [5] M. A. Heroux, D. W. Dorfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich. Improving performance via mini-applications. 2009.
- [6] T. Hoefler, J. Dinan, D. Buntinas, P. Balaji, B. Barrett, R. Brightwell, W. Gropp, V. Kale, and R. Thakur. Leveraging MPI's One-Sided Communication Interface for Shared-Memory Programming. In EuroMPI 2012, Vienna, Austria, volume 7490, Sep. 2012.
- [7] C. Huang, O. Lawlor, and L. V. Kal ́e. Adaptive MPI. In International Workshop on Languages and Compilers for Parallel Computing (LCPC), College Station, Texas, October 2003.
- [8] G. Inc. gperftools. https://code.google.com/p/gperftools.
- [9] R. Keller, E. Gabriel, B. Krammer, M. S. MA¨ijller, and M. M. Resch. Towards e cient execution of mpi applications on the grid: Porting and optimization issues. Journal of Grid Computing, 2003.
- [10] S. N. Laboratories. Large-scale atomic/molecular massively parallel simulator (LAMMPS). http://lammps.sandia.gov.
- [11] L. L. N. Laboratory. Simulation of underwater explosion benchmark experiments with ale3d. Technical Report UCRL-JC-123819, May 1997.
- [12] Lawrence Livermore National Laboratory. Hydrodynamics challenge problem, 2012.
- [13] D. Lea. Doug Lea's malloc (dlmalloc). http://g.oswego.edu/dl/html/malloc.html.
- [14] S. Li, T. Hoefler, , and M. Snir. NUMA-Aware Shared Memory Collective Communication for MPI. Jun. 2013. HPDC'13.
- [15] J. Liu, J. Wu, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. In ACM International Conference on Supercomputing (ICS'03), 2003.
- [16] Megan Gilge. IBM system Blue Gene solution: Blue Gene/Q application development, December 20 2012.
- [17] J. M. Mellor-Crummey and M. L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. ACM Transactions on Computer Systems, 9, 1991.
- [18] S. Negara, G. Zheng, K.-C. Pan, N. Negara, R. E. Johnson, L. V. Kale, and P. M. Ricker. Automatic MPI to AMPI Program Transformation using Photran. In 3rd Workshop on Productivity and Performance (PROPER 2010), number 10-14, Ischia/Naples/Italy, August 2010.
- [19] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008.
- [20] K. Pedretti and B. Barrett. XPMEM: Cross-Process Memory Mapping.
  [21] S. Pellegrini, T. Hoefler, and T. Fahringer. On the E<lects of CPU Caches on MPI Point-to-Point Communications. In IEEE International Conference on Cluster Computing (CLUSTER), sept. 2012.
- [22] M. P ́erache, P. Carribault, and H. Jourdren. MPC-MPI: An MPI implementation reducing the overall memory consumption. In EuroPVM/MPI, Berlin, Heidelberg, 2009.
- [23] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. JOURNAL OF COMPUTATIONAL PHYSICS, 117:1–19, 1995.
- [24] H. Tang and T. Yang. Optimizing threaded MPI execution on SMP clusters. In ACM International Conference on Supercomputing (ICS), 2001.
- [25] R. Thakur and W. Gropp. Test suite for evaluating performance of multithreaded MPI communication. Parallel Comput., 35(12), Dec. 2009.
- [26] D. Turner and X. Chen. Protocol-dependent message-passing performance on linux clusters. In IEEE International Conference on Cluster Computing, CLUSTER '02, Washington, DC, USA, 2002.
- [27] M. Woodacre, D. Robb, D. Roe, and K. Feind. The SGI Altix 3000 global shared-memory architecture. 2005.
- [28] H. wook Jin, S. Sur, L. Chai, and D. K. Panda. Limic: Support for high-performance mpi intra-node communication on linux cluster. In In International Conference on Parallel Processing (ICPP), pages 184–191, 2005.

Questions?