NETWORK-BASED COMPUTING LABORATORY

Designing An Efficient Kernel-level and User-level Hybrid Approach for MPI Intra-node Communication on Multi-core Systems

Lei Chai **Ping Lai** Hyun-Wook Jin*
Dhabaleswar. K. Panda

Computer Science & Engineering Department
The Ohio State University
*Department of Computer Science and Engineering
Konkuk University, Seoul, Korea

OHIO STATE

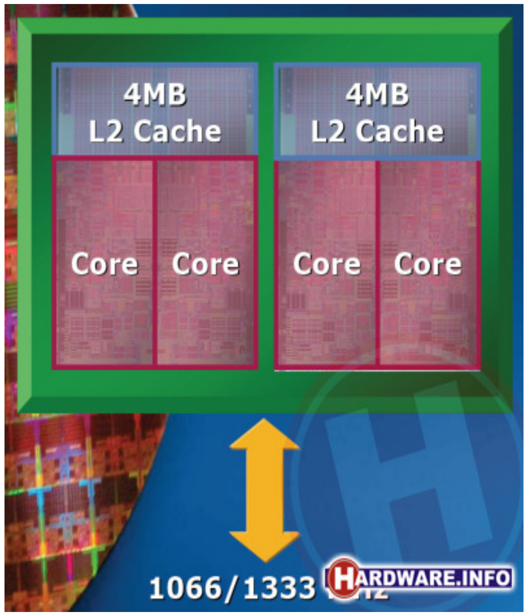
Outline

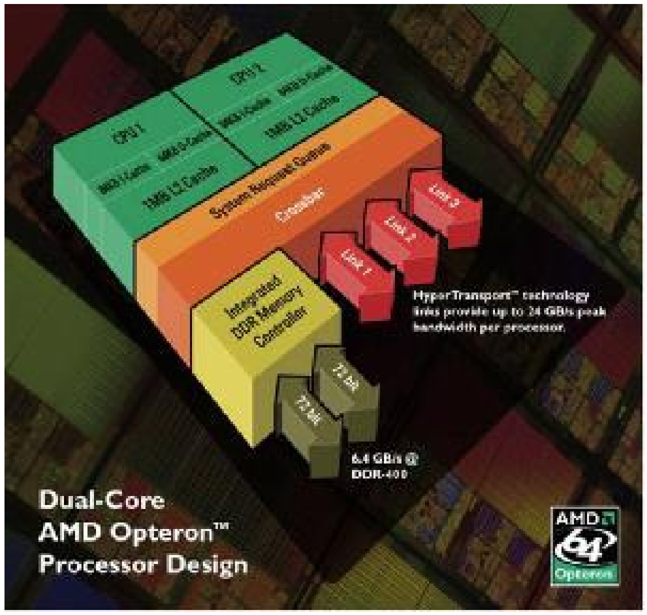
- Motivation and introduction
- Initial evaluation and analysis
- Design of the hybrid approach
- Performance results
- Conclusions and Future Work

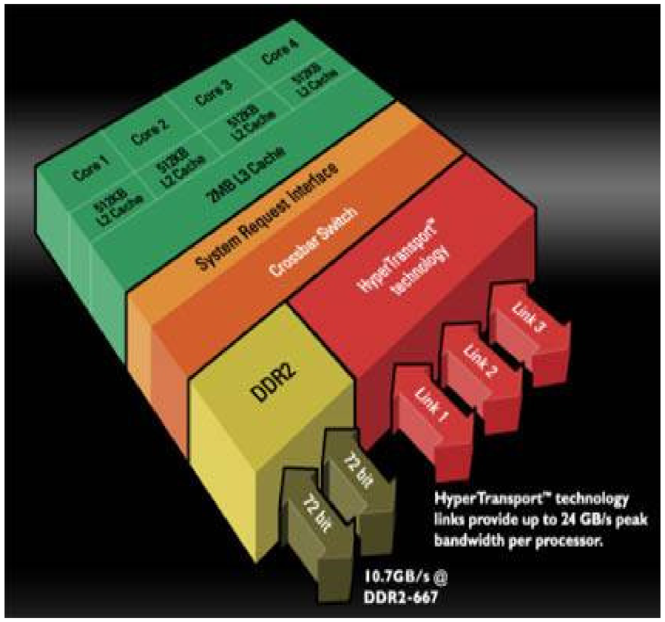
Motivation

- Single-core processors are reaching physical limits
  - Overheat problem
  - Power consumption
  - Design complexity
- Multi-core processor
  - Two or more processing cores on the same chip
  - Dividing workload to different cores
  - Chip-level Multi-Processor (CMP)
- Widely deployed in clusters
- More than 80% sites in Top500 supercomputer list use multi-core processors, June 2008
- MPI intra-node communication is more critical than ever!

Architectures of Multi-core Processors





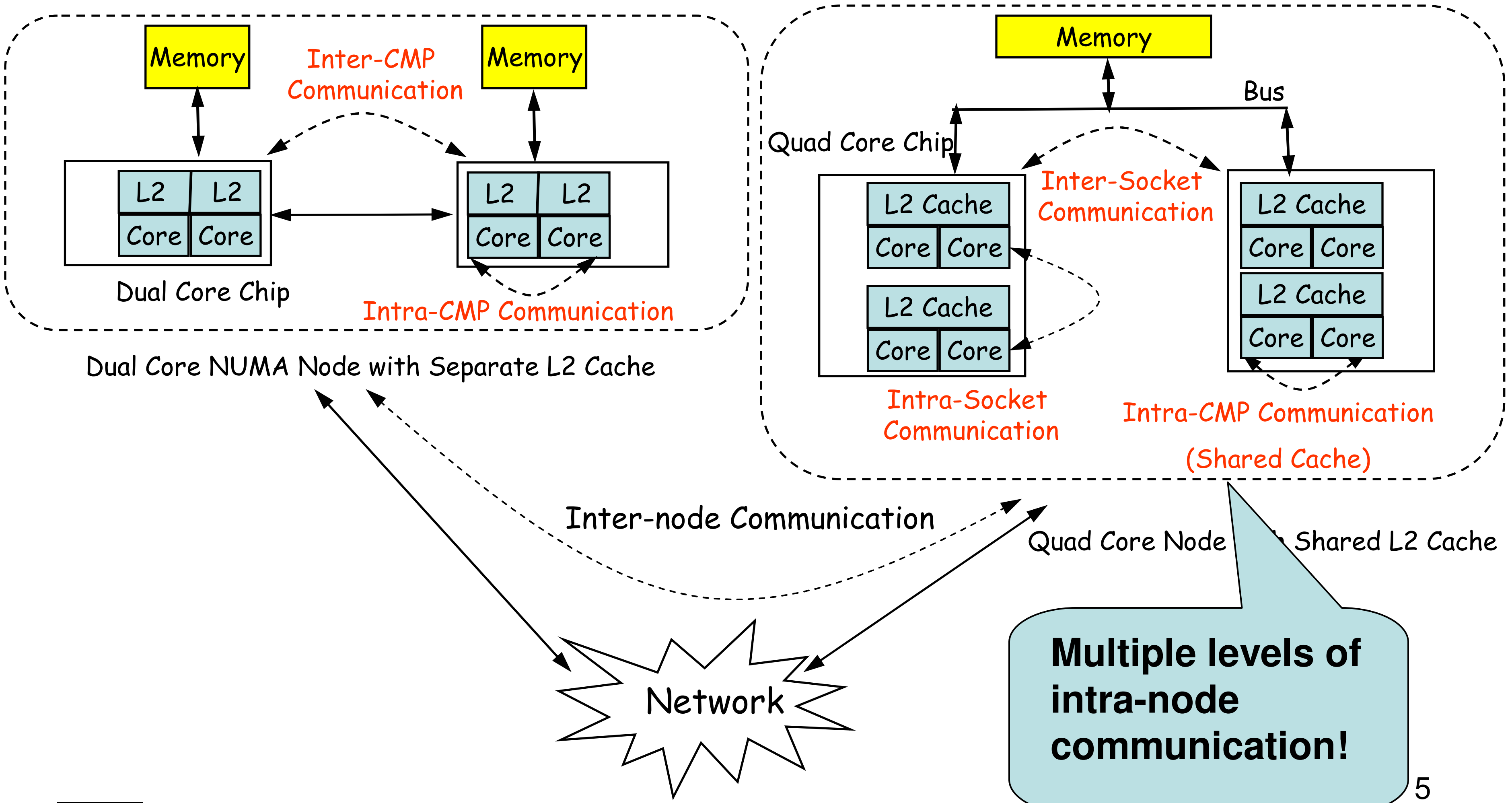


Intel Clovertown processor
Quad-core
Built from two Woodcrest sockets
Shared L2 cache

Dual-core Opteron
Separate L2 cache

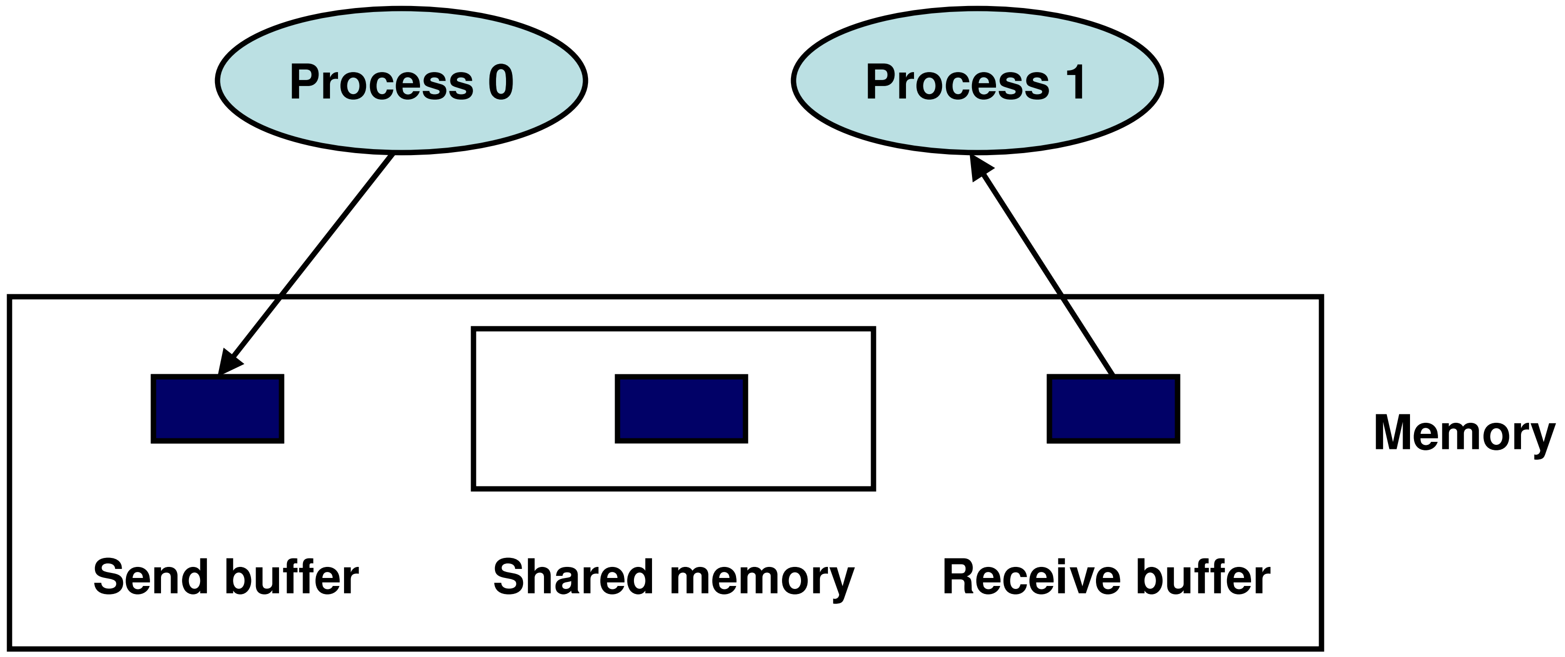
Quad-core Opteron
Separate L2 cache
Shared L3 cache

Multi-core Cluster Setup



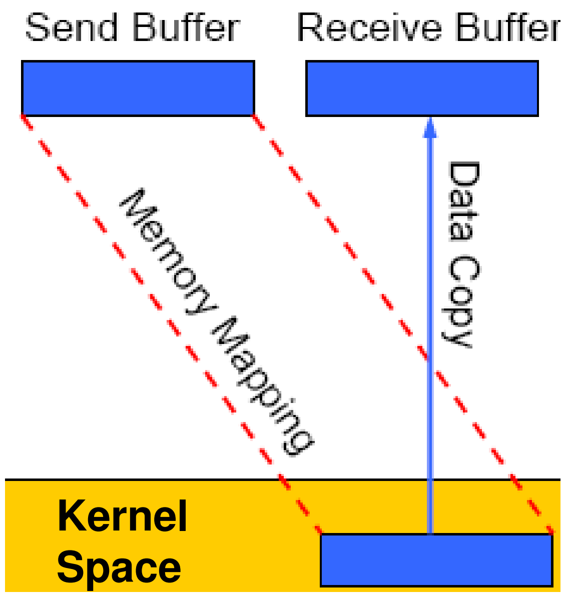
Multiple levels of intra-node communication!

Shared Memory Approach



- All the processes *mmap* a temporary file as the shared memory buffers for communication
- Minimal startup time
- Two copies

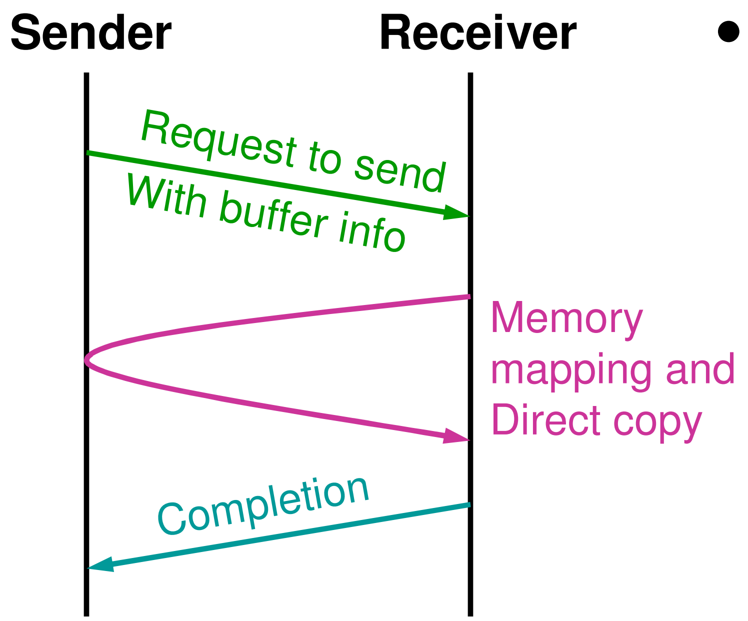
Kernel Assisted Direct Copy



- H. -W. Jin, S. Sur, L. Chai, and . K. Panda, LiMIC: Support for High Performance MPI Intra-node Communication on Linux Cluster, International Conference on Parallel Processing (ICPP) 2005.

- Taking help from the operating system to copy messages directly from one process's user buffer to another
- Kernel overhead
- One copy
- LiMIC
  - Stand alone communication module
  - Implements its own message queue and performs message matching etc
  - Designed for Linux kernel 2.4

Kernel Assisted Direct Copy (Cont'd)



- H. -W. Jin, S. Sur, L. Chai, D. K. Panda, Lightweight Kernel-Level Primitives for High-Performance MPI Intra-Node Communication over Multi-Core Systems. IEEE Cluster 2007 (Poster).

- LiMIC2
  - Light weight communication primitives
  - Only implements memory mapping and data movement primitives
  - Depends on the MPI library for message matching etc
  - Designed for Linux kernel 2.6
  - Uses a rendezvous protocol for send/receive
  - May potentially be affected by process skew

Problem Statement

- What are the relative performances of these two approaches?
- Can one of them be sufficient for MPI intra-node communication, especially on multi-core clusters?
- Can we design an efficient hybrid approach that optimizes performance?
- Can applications benefit from the hybrid approach?

Methodology

- Step 1: Initial evaluation
  - To study the characteristics of the two approaches
- Step 2: Designing a hybrid approach
  - To overcome the potential disadvantages of each individual approach (e.g. process skew issue in LiMIC2) and best combine the two approaches
- Step 3: A comprehensive performance evaluation on the hybrid approach
  - Micro-benchmarks
  - Collective operations
  - Applications

MVAPICH and MVAPICH-LiMIC2

- MVAPICH
  - High-performance, scalable, and fault-tolerant MPI library over
    - InfiniBand
    - 10GigE/iWARP
    - Other RDMA-enabled interconnects
  - Developed by Network-Based Computing Laboratory, OSU
  - Being used by more than 750 organizations world wide, including many of the top 500 supercomputers
  - For example, Ranger system at Texas Advanced Computing Center (TACC) ranked 4th in June '08 ranking
  - Current release versions use shared memory approach for intra-node communication
- MVAPICH-LiMIC2
  - An integrated version of MVAPICH with LiMIC2
  - Uses kernel assisted direct copy for intra-node communication
  - Will be available in future releases

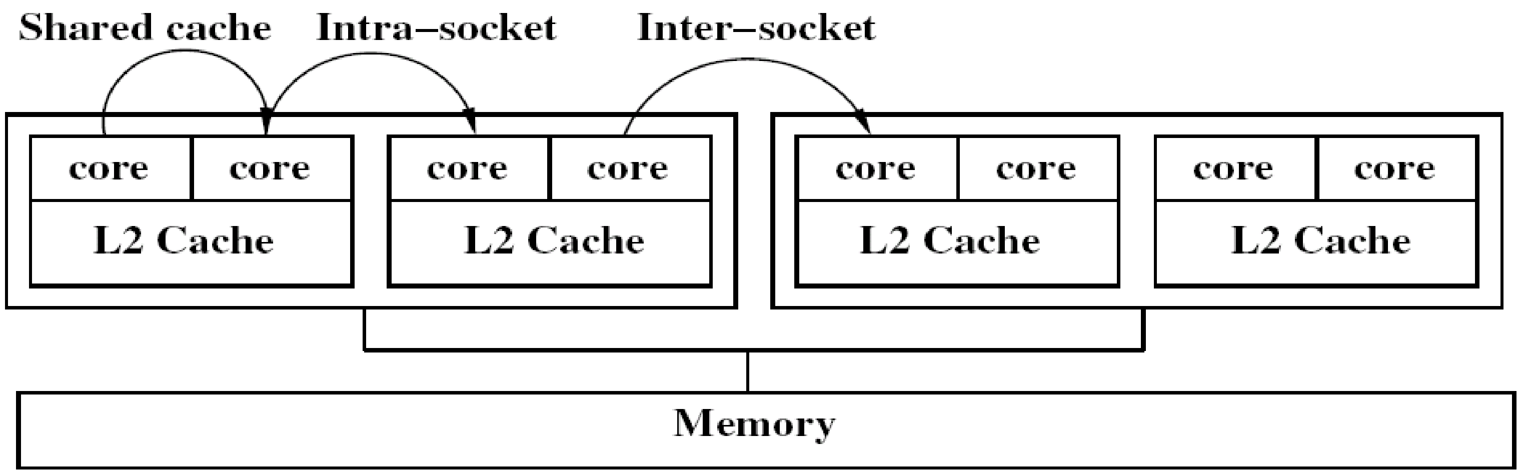
http://mvapich.cse.ohio-state.edu/

Outline

- Motivation and introduction
- Initial evaluation and analysis
- Design of the hybrid approach
- Performance results
- Conclusions and Future Work

Experimental System Setup

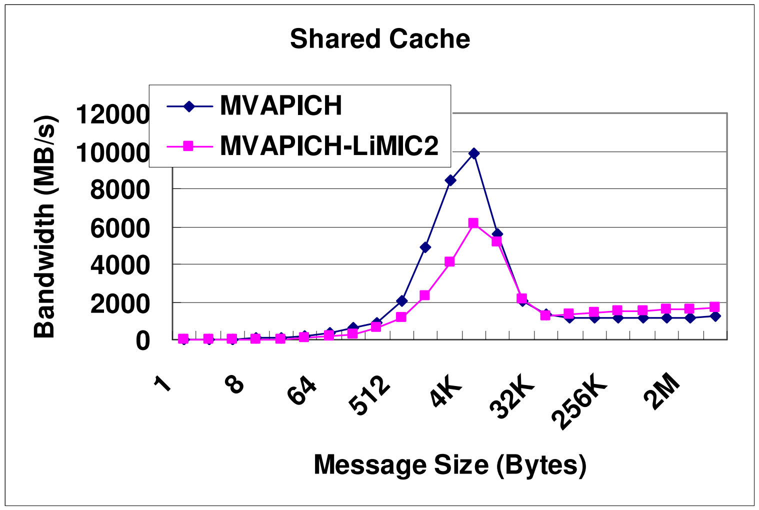
- Intel Clovertown cluster
  - Dual-socket quad-core Xeon processors, 2.0GHz
  - 8 processors per node, nodes connected by InfiniBand
  - A pair of cores on the same chip share a 4MB L2 cache
  - Linux 2.6.18

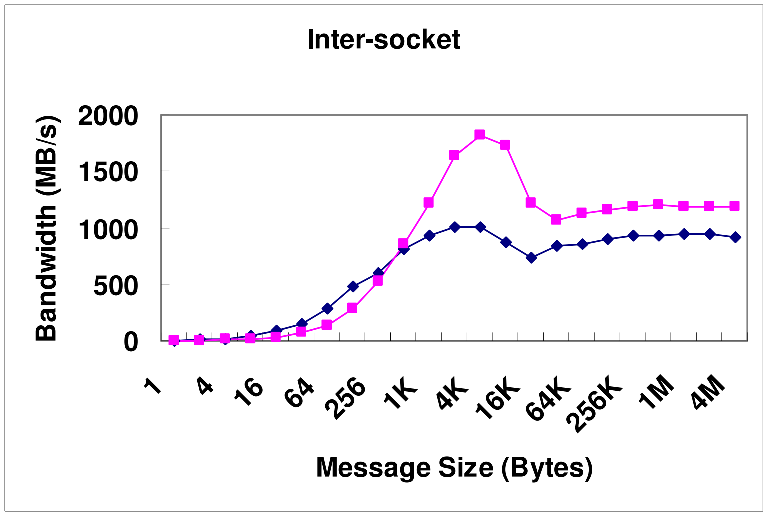


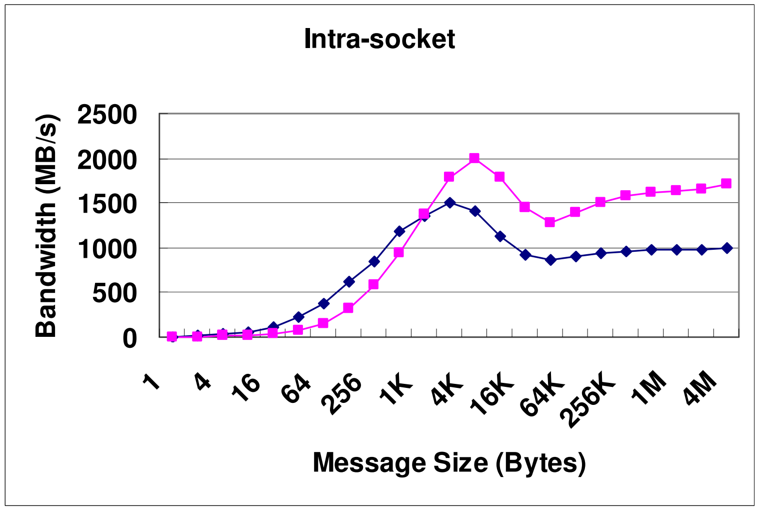
Experiments

- Micro-benchmarks
  - OSU Multi-pair bandwidth test
    - Without buffer reuse
    - With buffer reuse
  - Skew effect benchmark
- Tool to profile L2 cache utilization
  - Oprofile (http://oprofile.sourceforge.net)
- Experiments
  - Impact of processor topology
  - Impact of buffer reuse
  - L2 cache utilization
  - Impact of process skew

Impact of Processor Topology

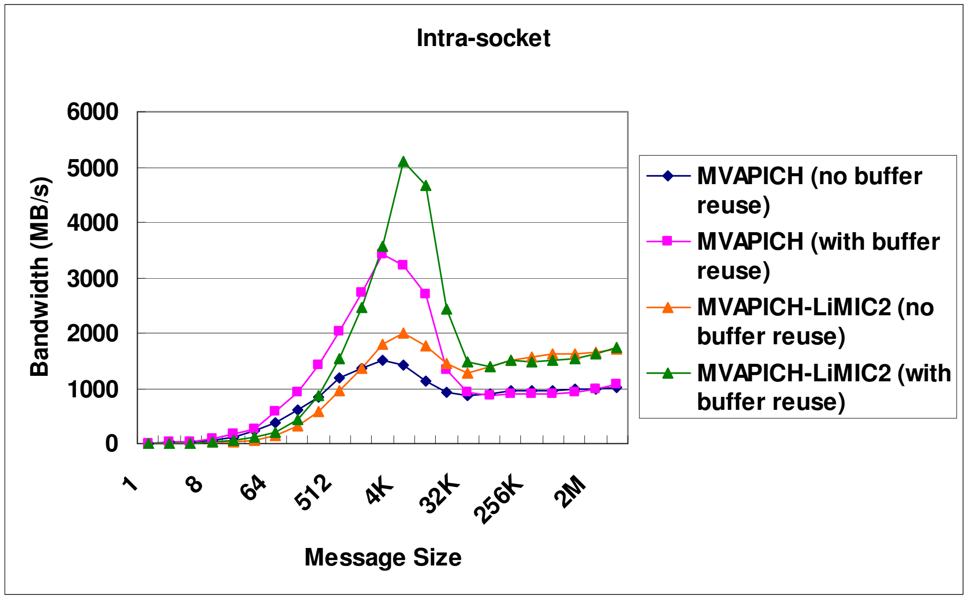






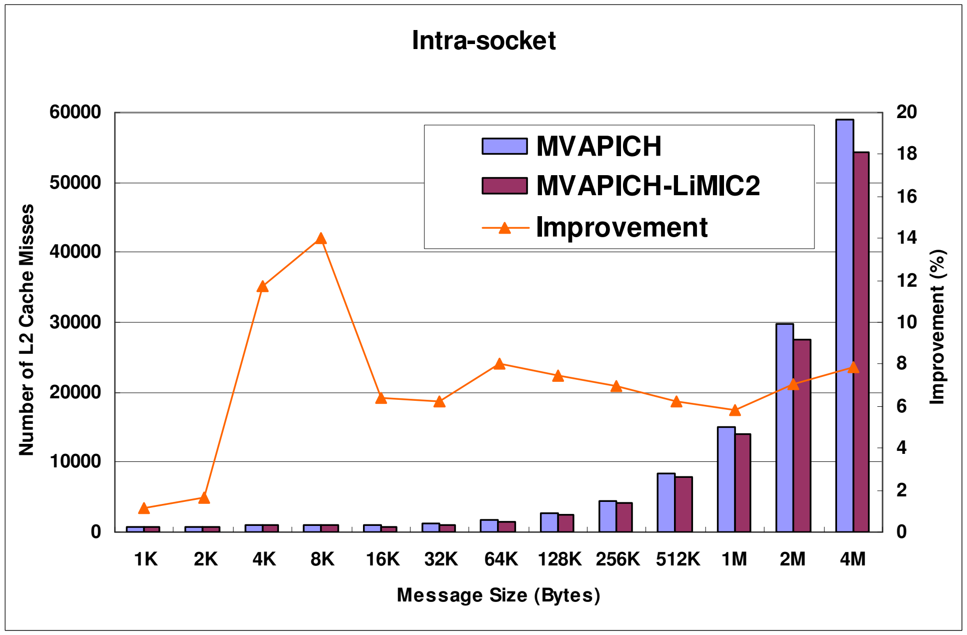
- MVAPICH-LiMIC2 performs better for medium and large messages
- Thresholds:
  - Shared cache: 32KB
  - Intra-socket: 2KB
  - Inter-socket: 1KB

Impacts of Buffer Reuse



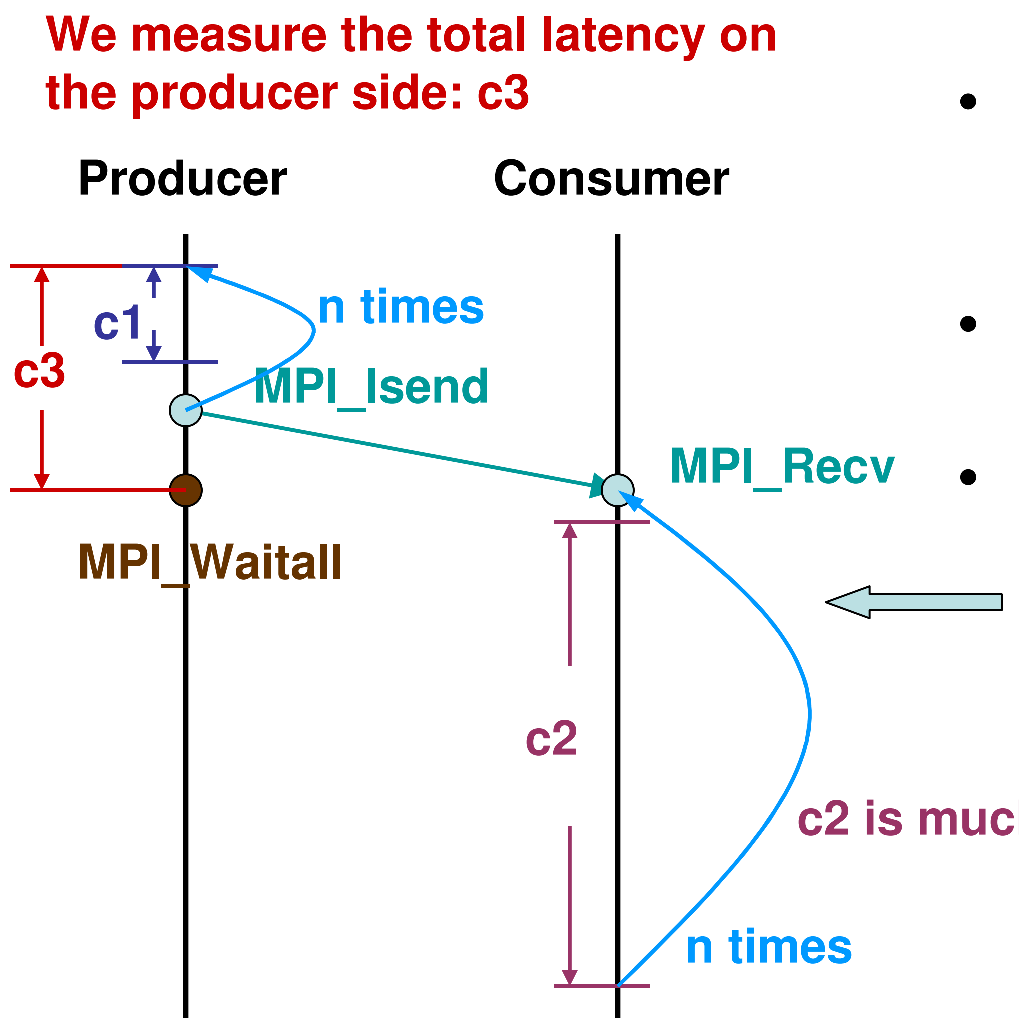
- Send/Receive multiple iterations over the same buffer
- Buffer reuse helps the performance of both MVAPICH and MVAPICH-LiMIC2
- MVAPICH-LiMIC2 benefits from buffer reuse better for medium messages because of better cache utilization
- Applications that have more buffer reuse will potentially benefit more from MVAPICH-LiMIC2

L2 Cache Utilization



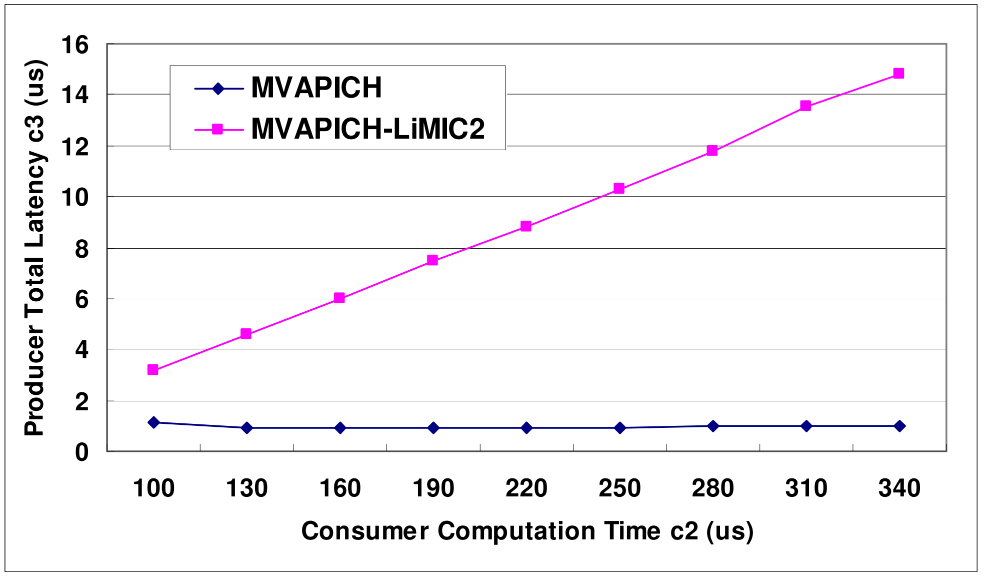
- Experiments are with buffer reuse
- Number of cache misses increase as message size increases
- MVAPICH-LiMIC2 has fewer cache misses than MVAPICH because it eliminates the intermediate copy

Impacts of Process Skew


We measure the total latency on the producer side: c3
Producer
Consumer
c1
c3
n times
MPI_Isend
MPI_Recv
MPI_Waitall
c2
c2 is much larger than c1
n times

- Process skew can potentially degrade application performance
- Skew tolerance is important!
- We designed a benchmark to measure process skew effect on performance

Process Skew Effects



- MVAPICH-LiMIC2: Send cannot complete until the corresponding receive is posted
- MVAPICH: Uses an intermediate buffer and is potentially more skew tolerant
- Theoretically:
  - *c3(MVAPICH) = (c1 + t(MPI Isend)) * window size n + t(MPI Waitall)*
  - *c3(MVAPICH-LiMIC2) = (t(MPI Recv) + c2) * window size n + t(MPI Waitall)*
- Experimental results conform to the expectation

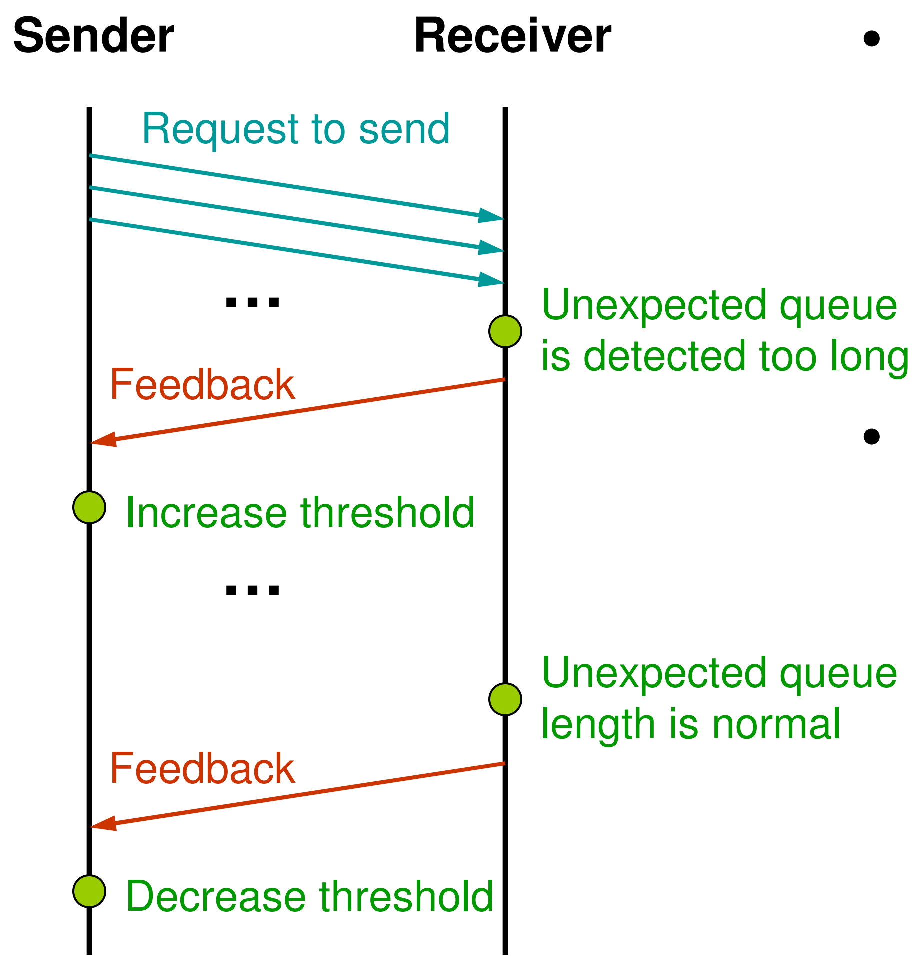
Outline

- Motivation and introduction
- Initial evaluation and analysis
- Design of the hybrid approach
- Performance results
- Conclusions and Future Work

Topology Aware Thresholds

- Thresholds to switch from shared memory to LiMIC2 need to be different for different cases
  - Shared cache: 32KB
  - Intra-socket: 2KB
  - Inter-socket: 1KB
- Parsing information exported in “sysfs” file system
  - Under */sys/devices/system/cpu/cpuX/topology/*
  - *physical package id*: Physical socket id of the logical CPU
  - *core id*: Core id of the logical CPU on the socket
- Every process knows the topology based on the above information
- Using CPU affinity to pin processes to processors

Skew Aware Thresholds

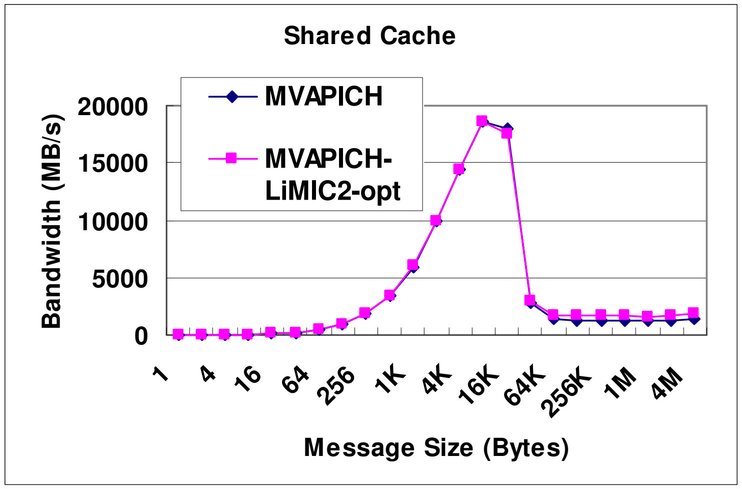


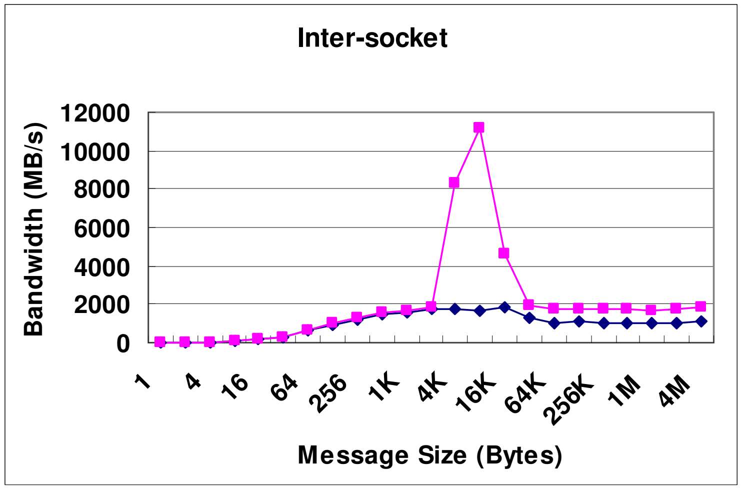
- Unexpected message queue
  - Messages received without the matching receive call being posted
  - Length indicates the process skew extent
- Skew aware
  - Receiver keeps track of the length of the unexpected message queue
  - Sends feedbacks to the sender
  - Sender adjusts thresholds accordingly

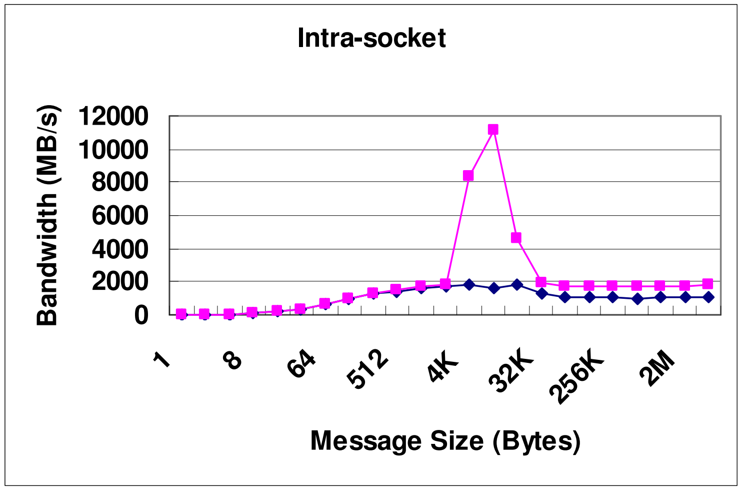
Outline

- Motivation and introduction
- Initial evaluation and analysis
- Design of the hybrid approach
- Performance results
- Conclusions and Future Work

Results of Topology Aware Thresholds



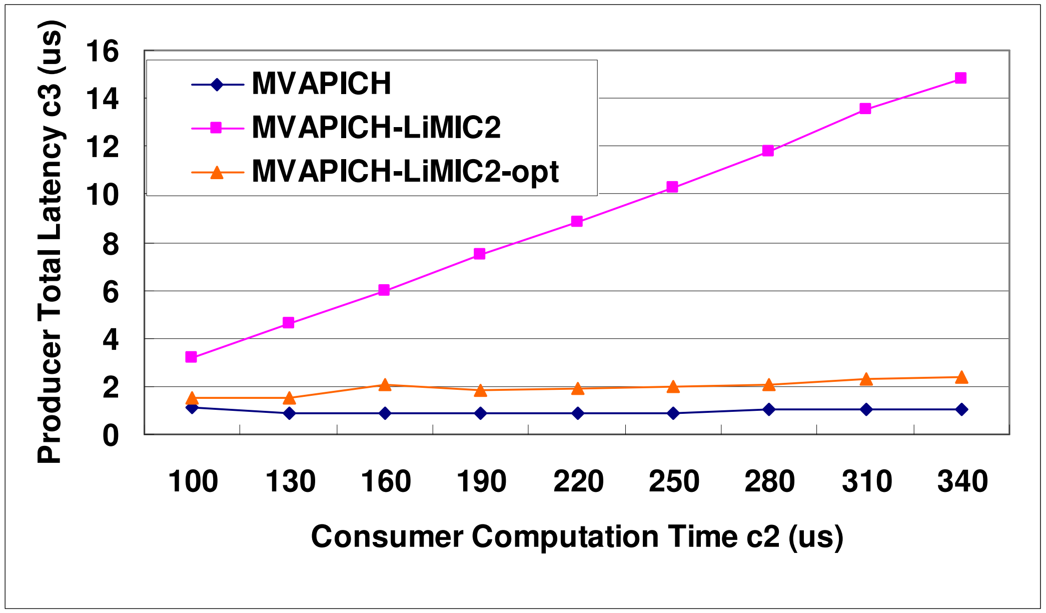




- Topology aware thresholds:
  - Shared cache: 32KB
  - Intra-socket: 2KB
  - Inter-socket: 1KB
- Performance optimized for all the three cases

NETWORK-BASED COMPUTING LABORATORY

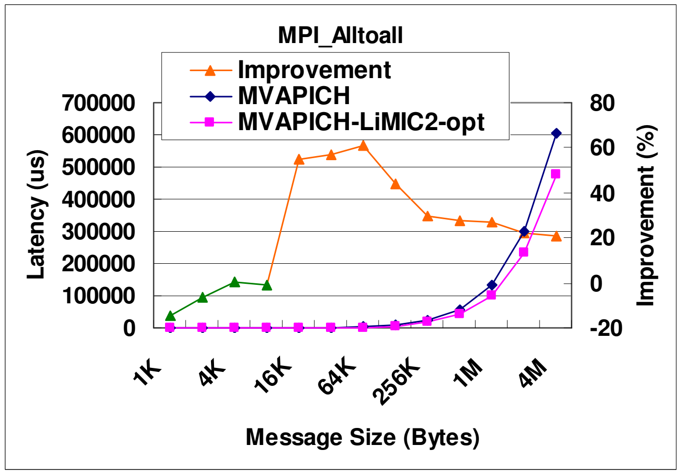
Results of Skew Aware Thresholds

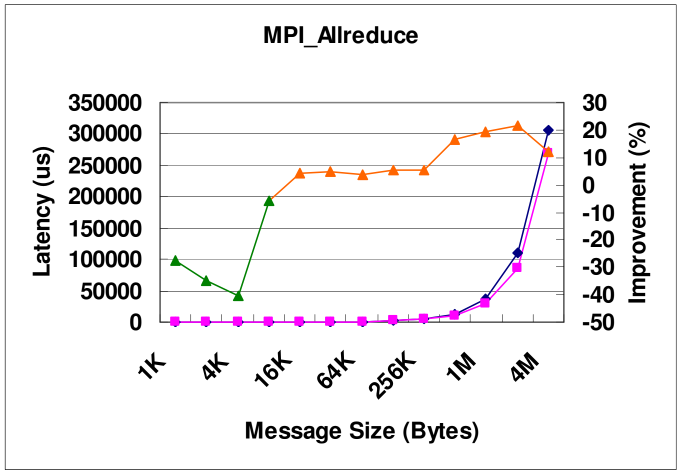


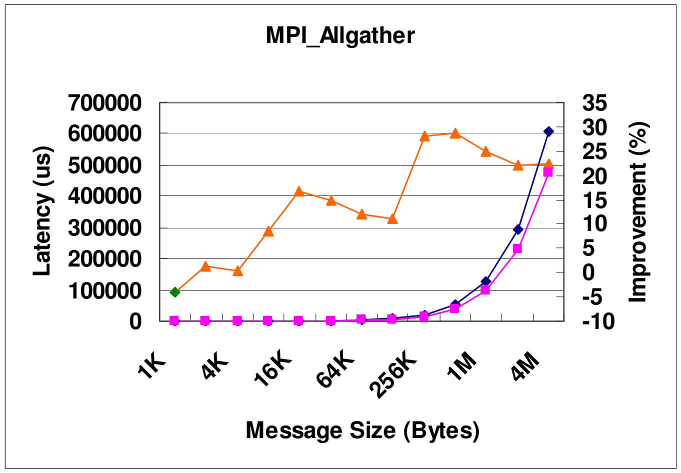
- The sending process can detect process skew quickly and adapt to it
- The producer latency is much lower with skew detection
  - Close to that of MVAPICH

OHIO STATE

Performance of Collectives





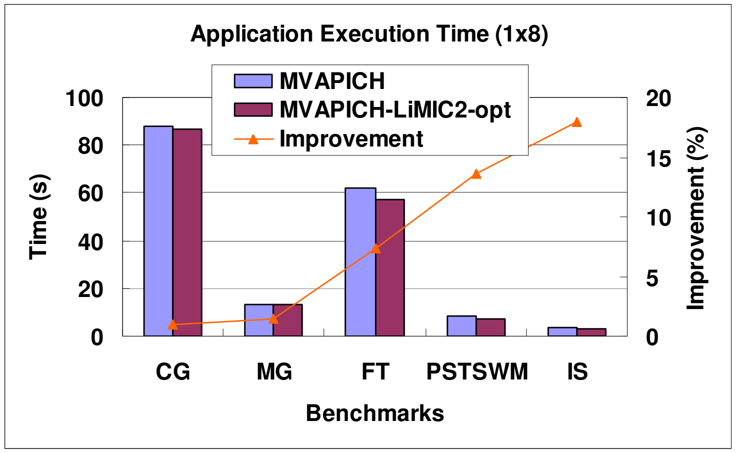


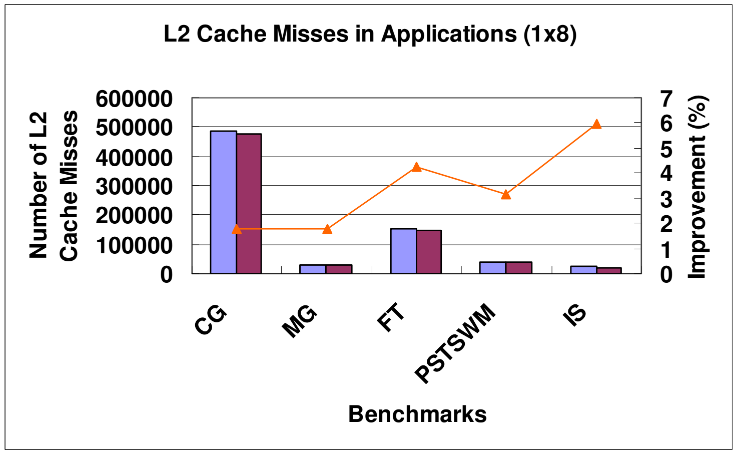
- Improvements:
  - MPI_Alltoall: 60%
  - MPI_Allgather: 28%
  - MPI_Allreduce: 21%
- MPI_Allreduce
  - MVAPICH performs better for 1KB-8KB messages
  - MVAPICH uses optimized algorithms
  - MVAPICH-LiMIC2-opt can be further optimized for collective operations

NETWORK-BASED COMPUTING LABORATORY

Performance of Applications (1)

- Performance on a single node with 8 processes
- MVAPICH-LiMIC2-opt improves the performance of FT, PSTSWM, and IS significantly
  - FT: 8%
  - PSTSWM: 14%
  - IS: 17%
- Cache utilization is improved by up to 6%

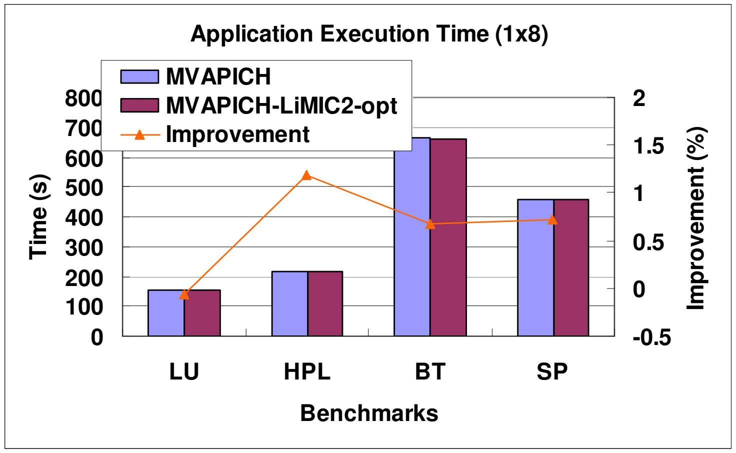


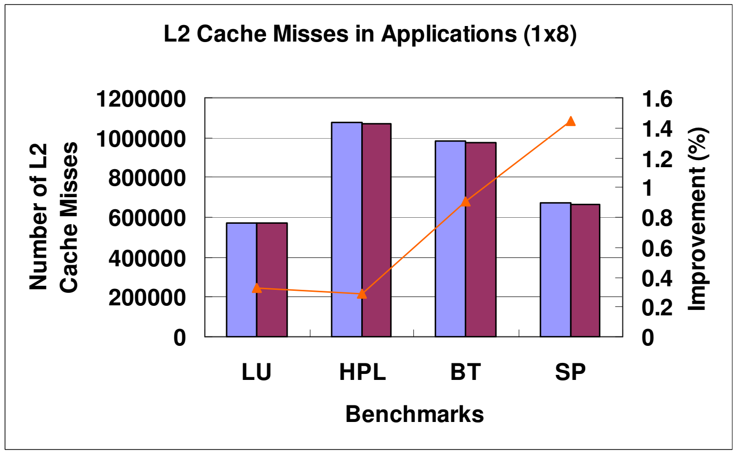


OHIO STATE

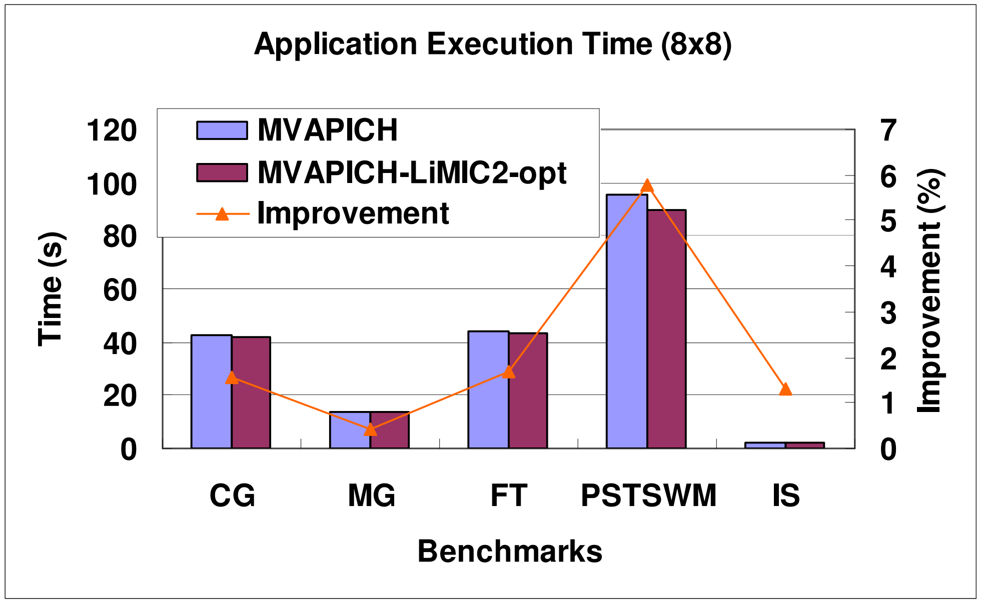
Performance of Applications (2)

- Performance on a single node with 8 processes
- Performance improvement is under 2% for LU, HPL, BT, and SP
  - Communication time is not significant, or
  - Not many large messages
- Applications that transfer more large messages can benefit more from MVAPICH-LiMIC2-opt





Performance of Applications (3)



- Performance on 8 nodes with 8x8 processes
- PSTSWM still benefits from MVAPICH-LiMIC2-opt
- MVAPICH-LiMIC2-opt is promising for clusters

Outline

Conclusions

- An efficient hybrid approach for MPI intra-node communication
  - Shared memory (MVAPICH)
  - OS kernel assisted direct copy (MVAPICH-LiMIC2)
- 1st step: Initial evaluation
  - MVAPICH-LiMIC2 provides better performance than MVAPICH depending on the message size and physical topology
    - Fewer number of copies
    - Efficient cache utilization
  - MVAPICH-LiMIC2 is less skew-tolerant than MVAPICH
- 2nd step: Designing an efficient hybrid approach
  - Topology-aware thresholds
  - Skew-aware thresholds

Conclusions (Cont'd)

- 3rd step: Comprehensive performance evaluation
  - MPI Alltoall, MPI Allgather, and MPI Allreduce performances are improved by up to 60%, 28%, and 21%, respectively.
  - FT, PSTSWM, and IS performances are improved by 8%, 14%, and 17%, respectively.
- Software distribution
  - Will be available in upcoming MVAPICH releases!

Future Work

- Optimizations on collective algorithms for MVAPICH-LiMIC2
- Evaluation on AMD dual-core and quad-core platforms
- In-depth studies on how the improvements in MPI intra-node communication benefit the application performance

Acknowledgements

Our research is supported by the following organizations

- Current Funding support by

Office of Science U.S. DEPARTMENT OF ENERGY

NSF

Mellanox Technologies

CISCO SYSTEMS EMPOWERING THE INTERNET GENERATION

QLOGIC

intel.

Linux Networx PROVEN

NetApp

Sun microsystems

- Current Equipment support by

Mellanox Technologies

intel.

AMD

Sun microsystems

APPRO HPC Cluster Solutions

advanced clustering technologies, inc.

Microway

QLOGIC

Thank you !

{chail, laipi, panda}@cse.ohio-state.edu

jinh@konkuk.ac.kr

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

MVAPICH Web Page

http://mvapich.cse.ohio-state.edu/

System Software Lab

http://sslab.konkuk.ac.kr