

•
•
•
•

Designing Multi-core Aware Middleware for HPC and Data-Centers



Dhabaleswar K. (DK) Panda
Department of Computer Science and
Engineering
The Ohio State University
E-mail: panda@cse.ohio-state.edu
<http://www.cse.ohio-state.edu/~panda>

• • • • • • • • • •



Presentation Outline



- Introduction
- Multi-core Aware Middleware
- Designs for MPI and Evaluation
- Designs for Data-Centers and Evaluation
- Conclusions and Future work

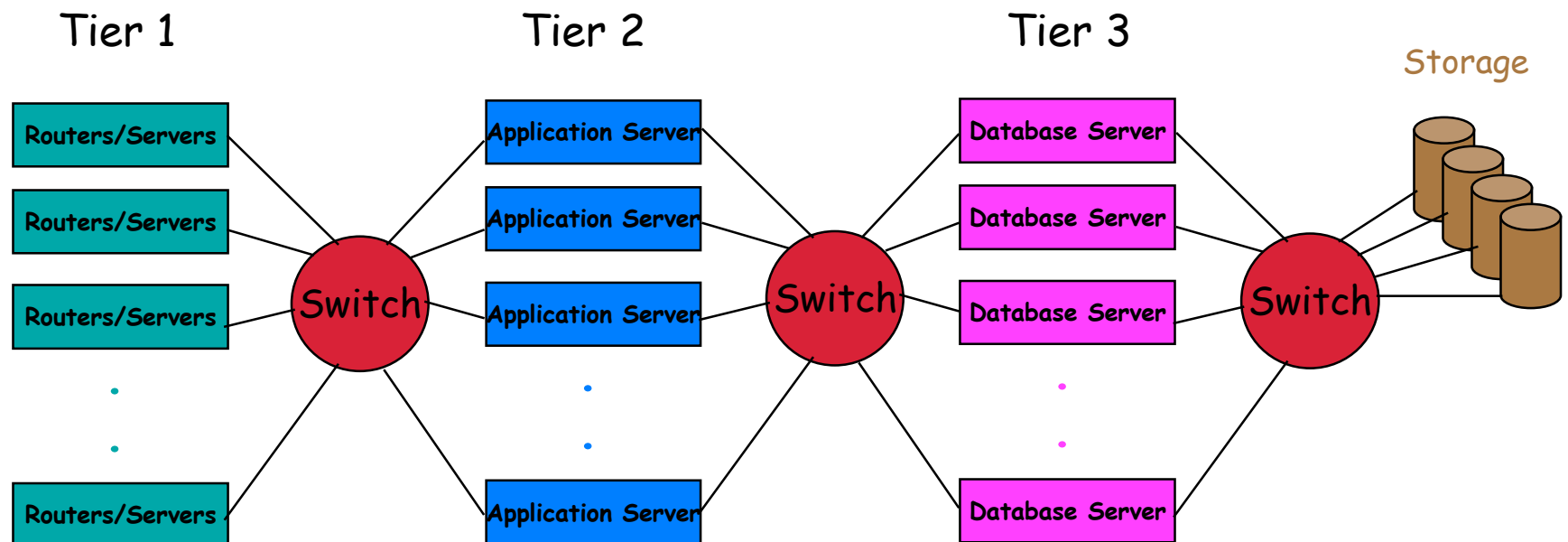


•
•
•

Trends for Computing Clusters in the Top 500 List

- Top 500 list of Supercomputers (www.top500.org)
 - June 2001: 33/500 (6.6%)
 - Nov 2001: 43/500 (8.6%)
 - June 2002: 80/500 (16%)
 - Nov 2002: 93/500 (18.6%)
 - June 2003: 149/500 (29.8%)
 - Nov 2003: 208/500 (41.6%)
 - June 2004: 291/500 (58.2%)
 - Nov 2004: 294/500 (58.8%)
 - June 2005: 304/500 (60.8%)
 - Nov 2005: 360/500 (72.0%)
 - June 2006: 364/500 (72.8%)
 - Nov 2006: 361/500 (72.2%)

Increasing Use of Clusters for Multi-Tier Data Centers





- All major search engines and e-commerce companies are using clusters for multi-tier datacenters
 - Google, Amazon, Financial institutions,



Multi-core Processors - The Wave of the Future



- Node architectures are changing rapidly
 - Especially with multi-core processors
 - Use of Multi-core systems in the TOP500
 - Opteron dual-core - 15.6%
 - Xeon 51xx (Woodcrest) - 6.2%
 - Introducing new **memory-hierarchy**
 - Core-core communication
 - Within a socket (on-chip), Intra-CMP
 - Across a socket (off-chip), Inter-CMP
 - Inter-node communication (off-chip)
 - Are we getting maximum performance on these systems?
 - Can we design better software to extract the maximum performance?
- 
- 



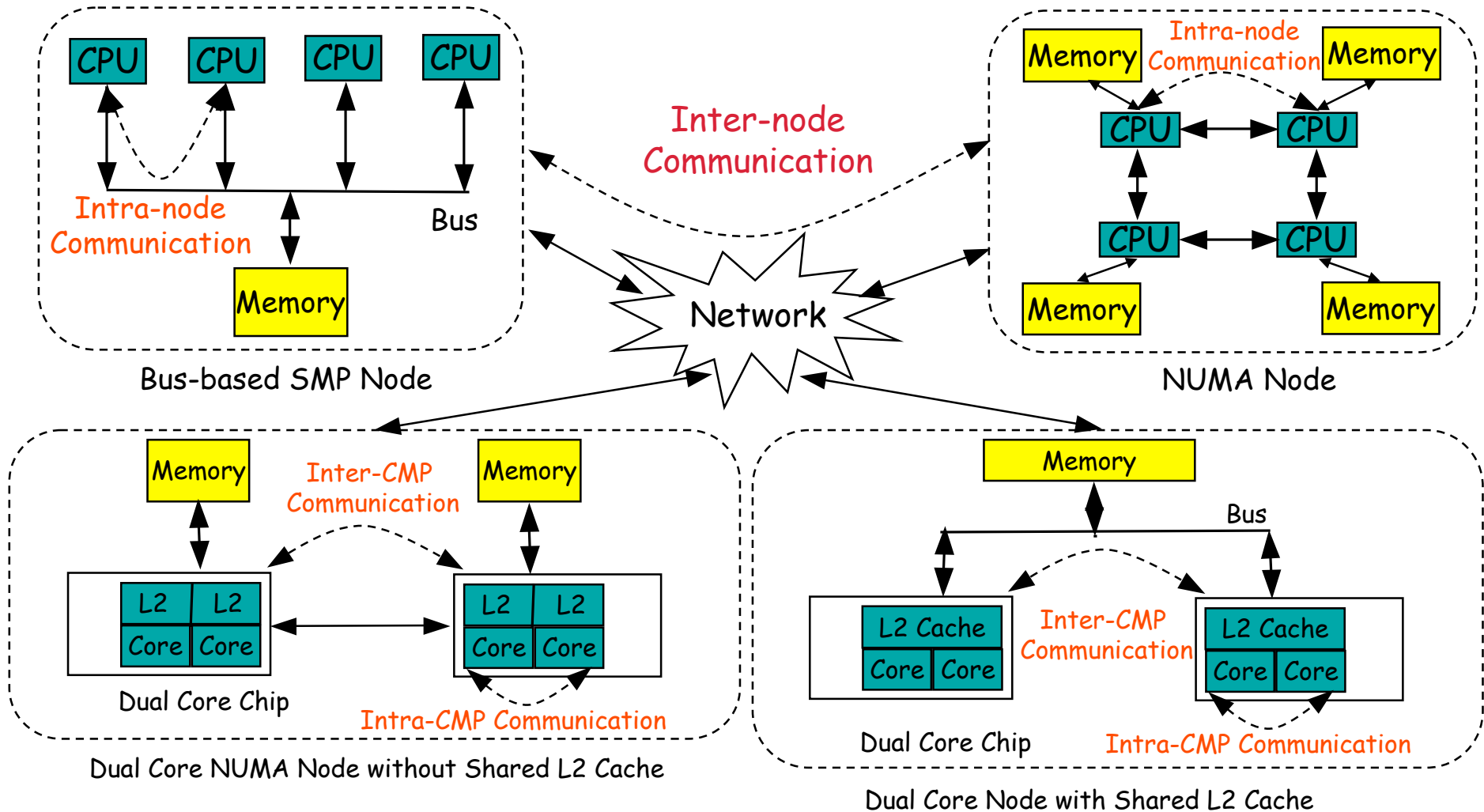
Presentation Outline



- Introduction
- *Multi-core Aware Middleware*
- Designs for MPI and Evaluation
- Designs for Data-Centers and Evaluation
- Conclusions and Future work



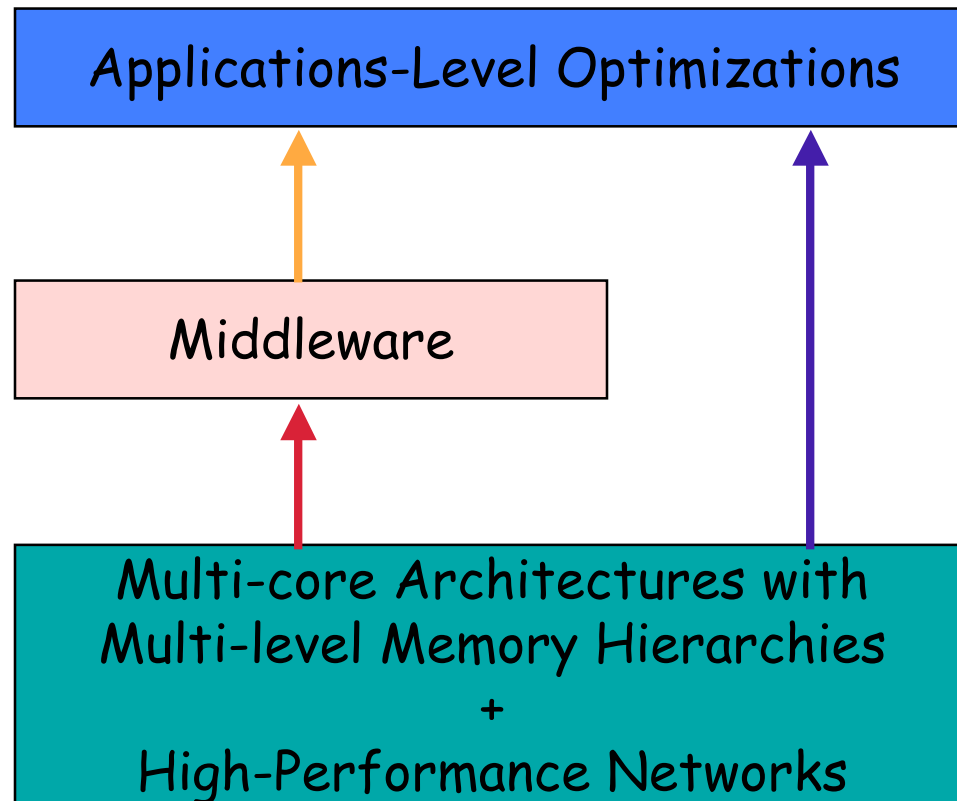
Multiple Types of Memory Hierarchies in Multi-core Clusters



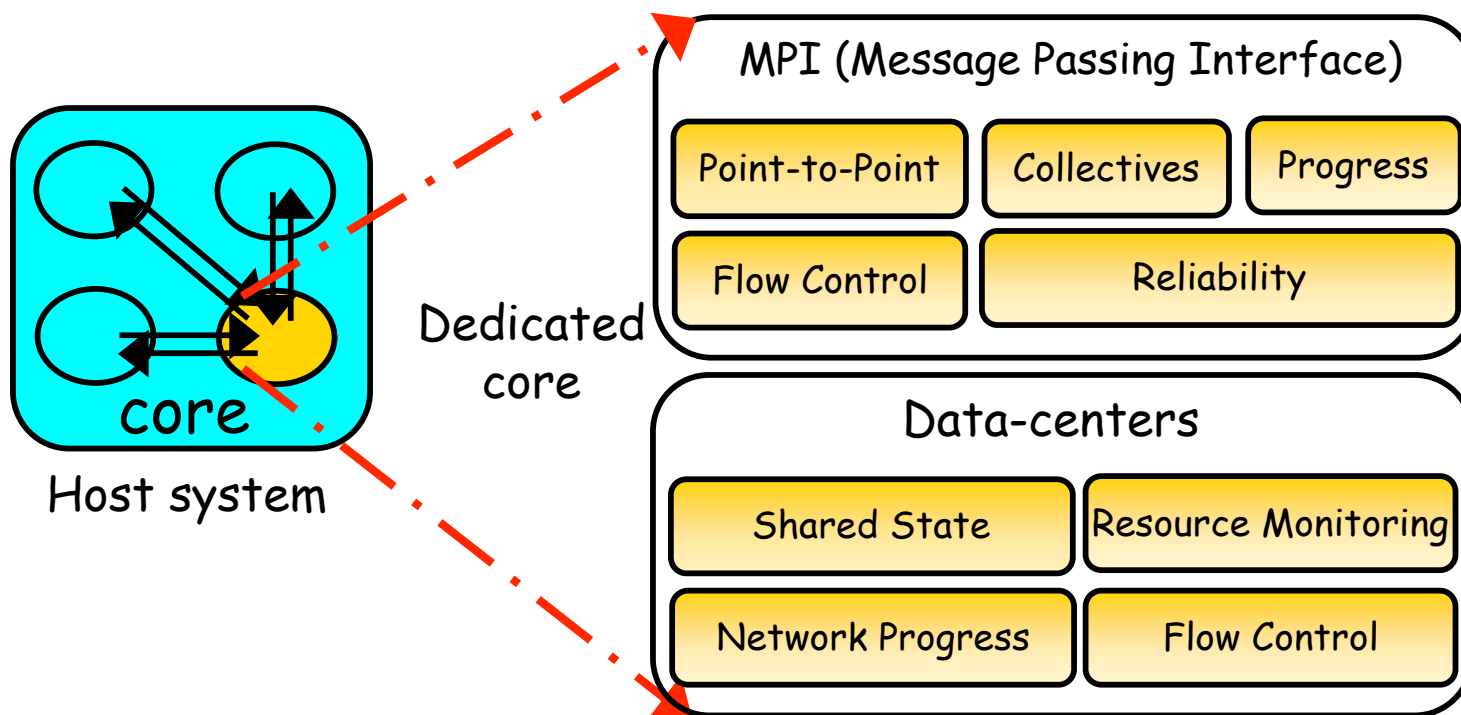
⋮

Designing Efficient Middleware with Multi-core Architectures

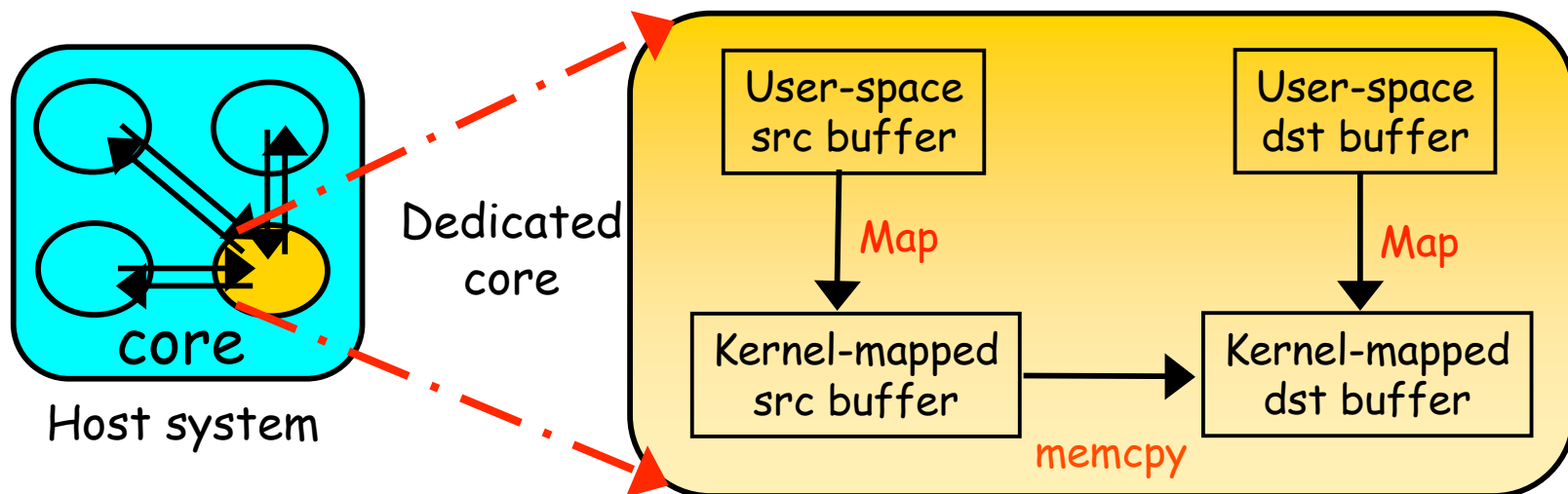
HPC and
Data-Centers



Dedicated Cores for HPC and Data-Centers



An Example: Onloading Memory Copy (MCNI)



- Many of the existing approaches use at least two copies to copy buffers across different processes
- Dedicate a core to reduce the memory copies
- Map the source and destination buffers in kernel
- Perform memory copy directly from source to destination



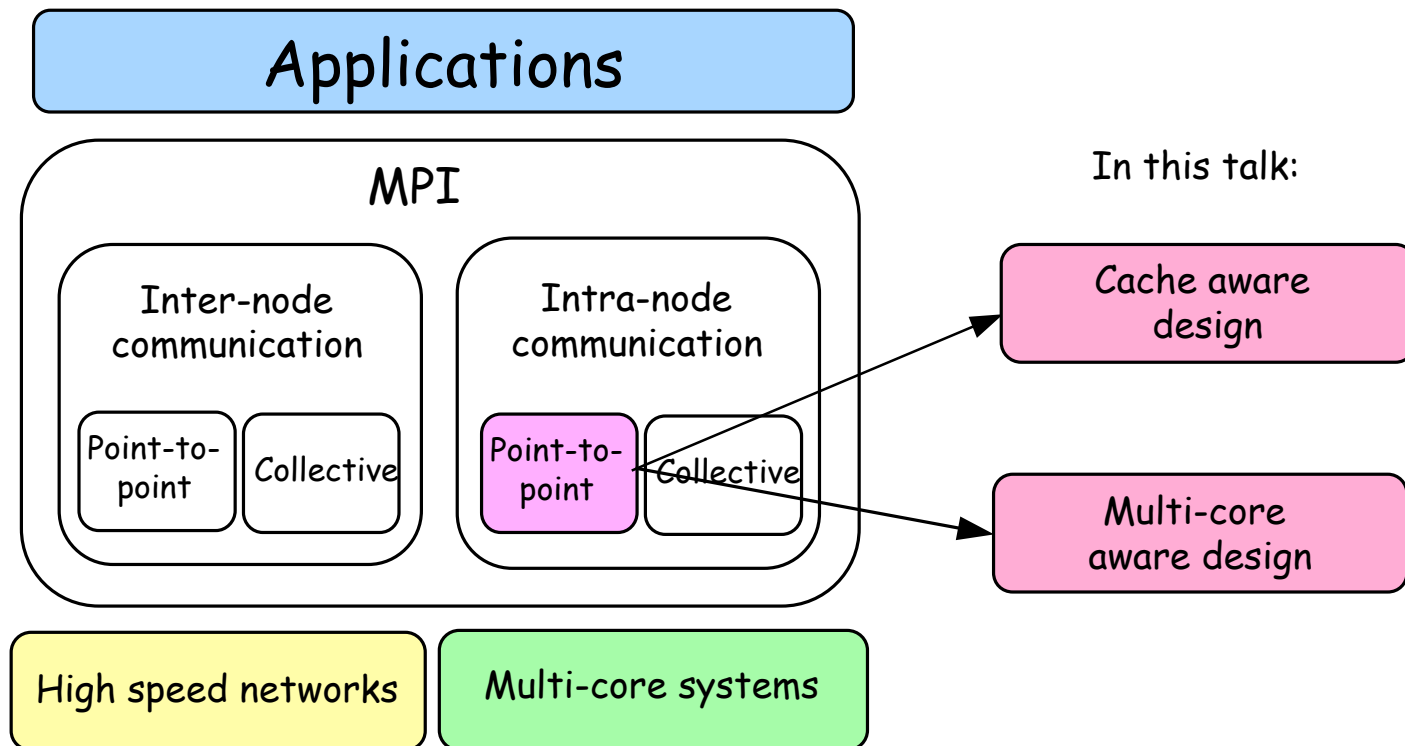
Presentation Outline



- Introduction
- Multi-core Aware Middleware
- Designs for MPI and Evaluation
- Designs for Data-Centers and Evaluation
- Conclusions and Future work





Overview of Memory Copy Optimizations on MPI Stack



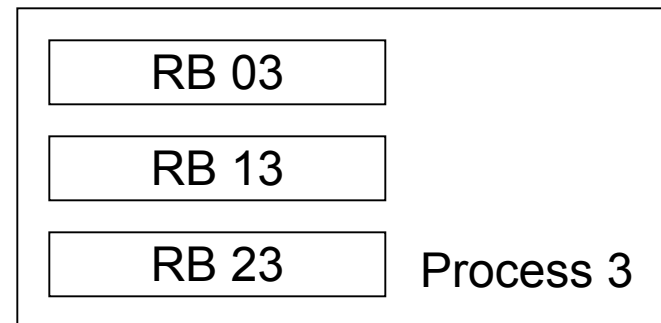
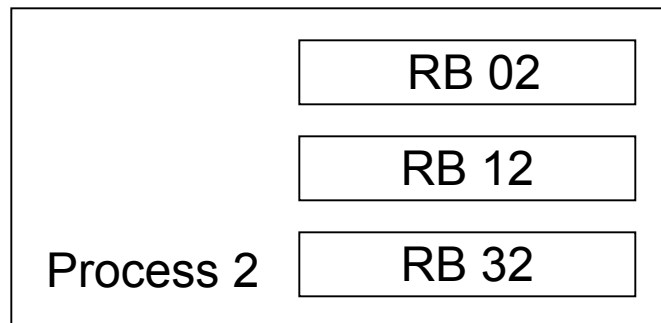
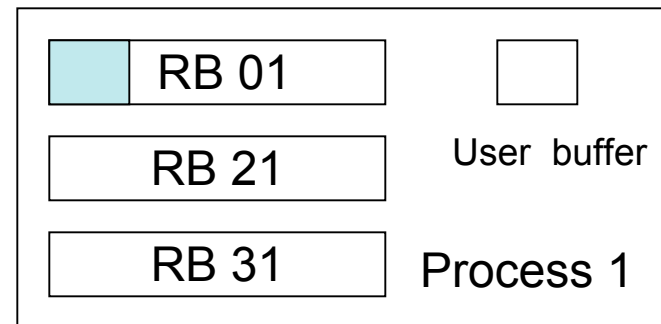
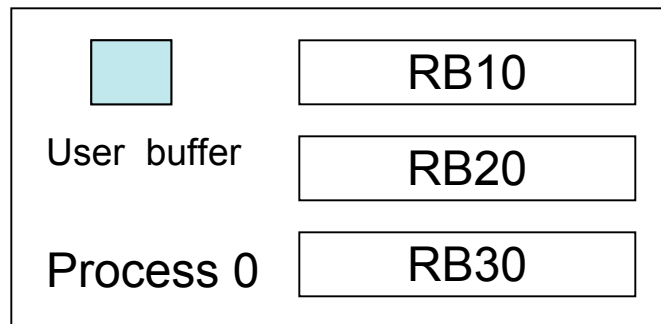


MPI Library (MVAPICH and MVAPICH2)



- High Performance MPI Library for InfiniBand Clusters
 - MVAPICH (MPI-1) and MVAPICH (MPI-2)
 - Used by more than 455 organizations in 30 countries
 - Empowering many TOP500 clusters
 - Available with software stacks of many InfiniBand and server vendors including the OpenIB and Open Fabrics Enterprise Distribution (OFED)
 - <http://nowlab.cse.ohio-state.edu/projects/mpi-iba/>
 - Already has good support for intra-node MPI point-to-point communication (with copying) over shared memory
- 
- 

Basic Intra-node Communication Design



- RB_{xy}: Receive Buffers
 - Buffers shared between processes
 - x: sender, y: receiver

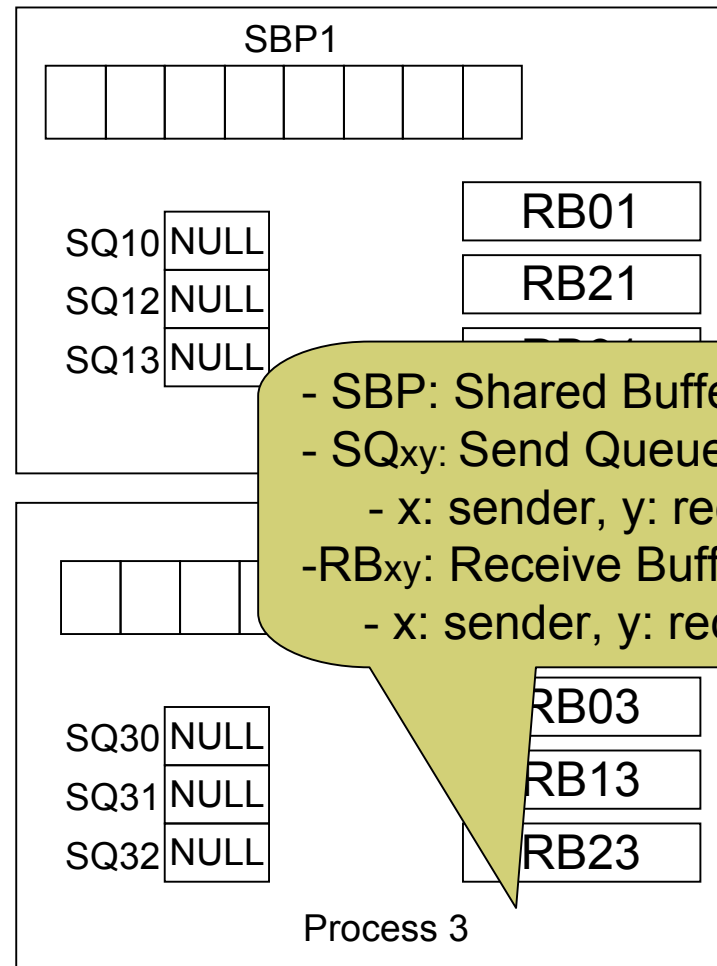
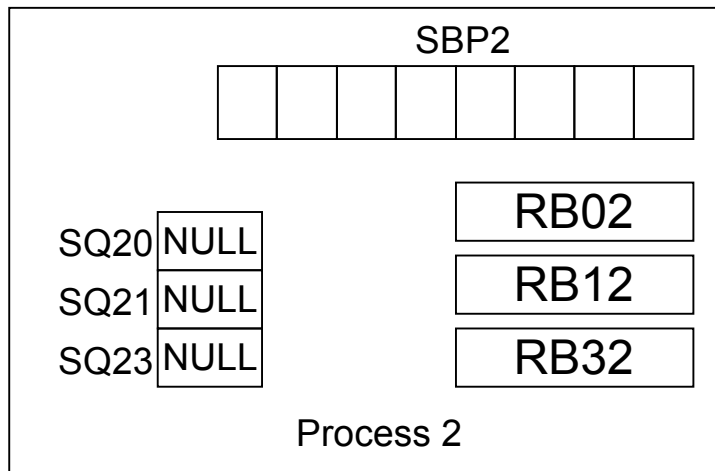
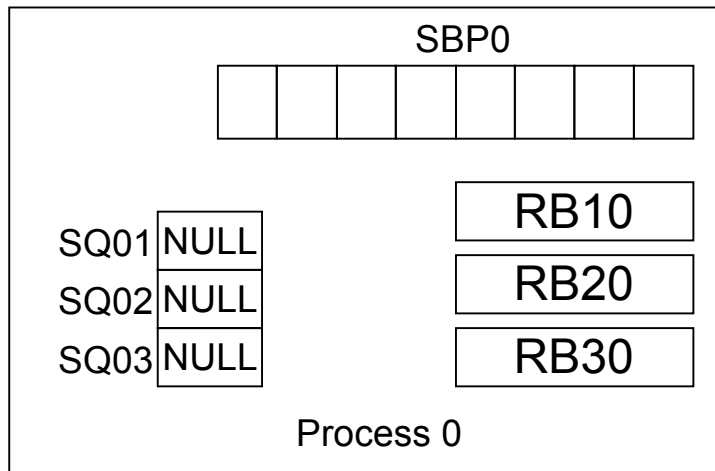


Disadvantages of the Basic Design



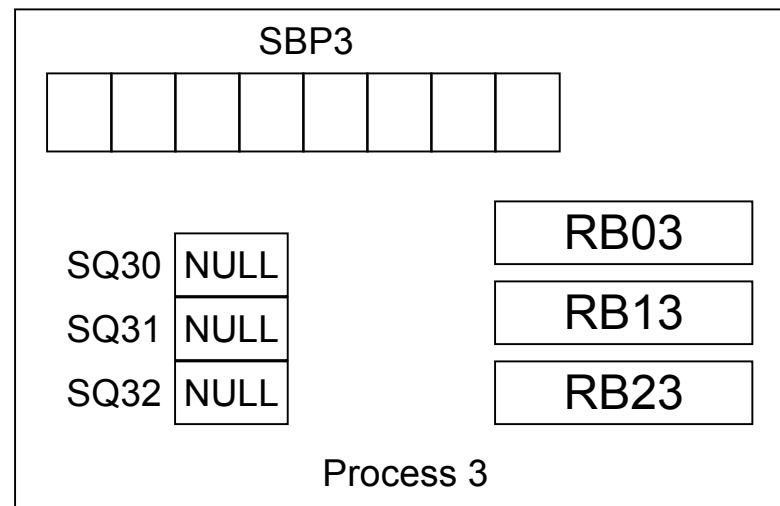
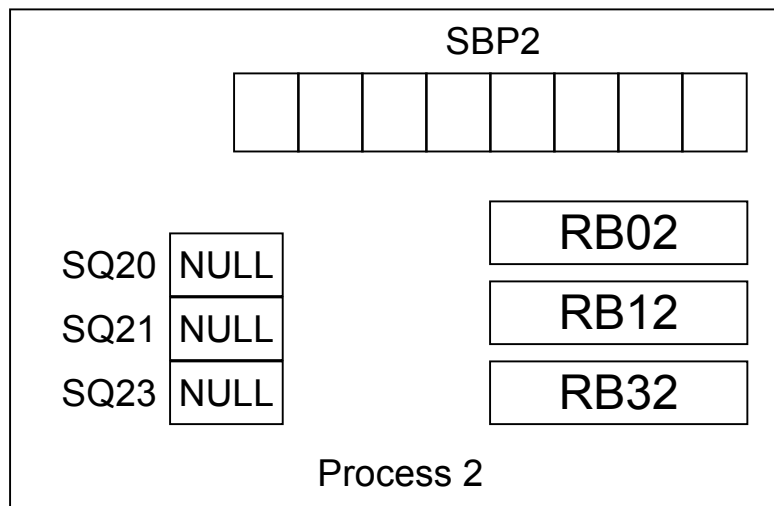
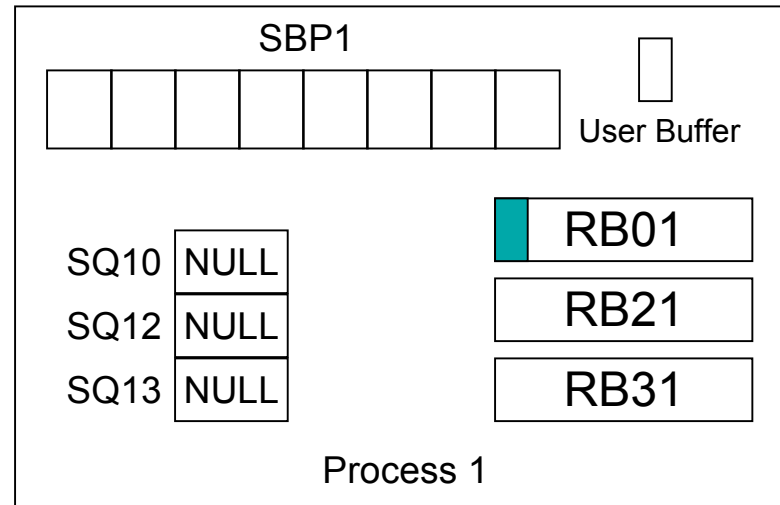
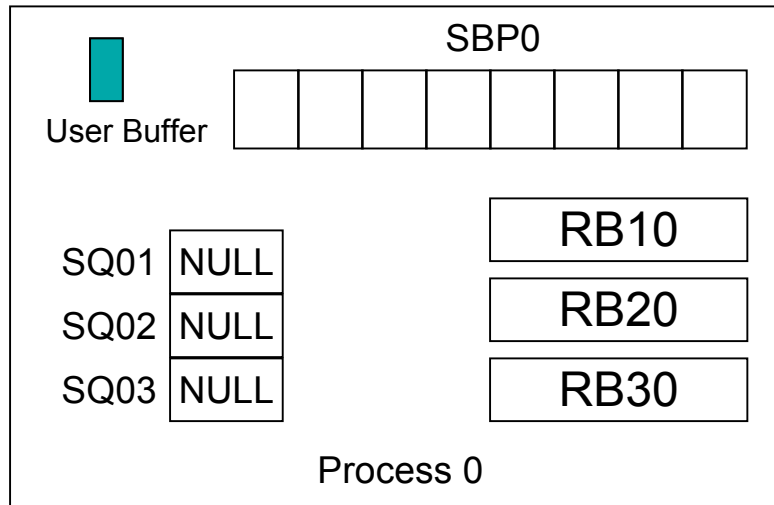
- Large memory usage
 - Not scalable
- Inefficient cache utilization
 - Need to walk through the receive buffer
 - Performance not optimized

Cache Aware Intra-node Communication Design



- SBP: Shared Buffer Pool
- SQ_{xy}: Send Queue
 - x: sender, y: receiver
- RB_{xy}: Receive Buffer
 - x: sender, y: receiver

Small Message Transfer



•
•

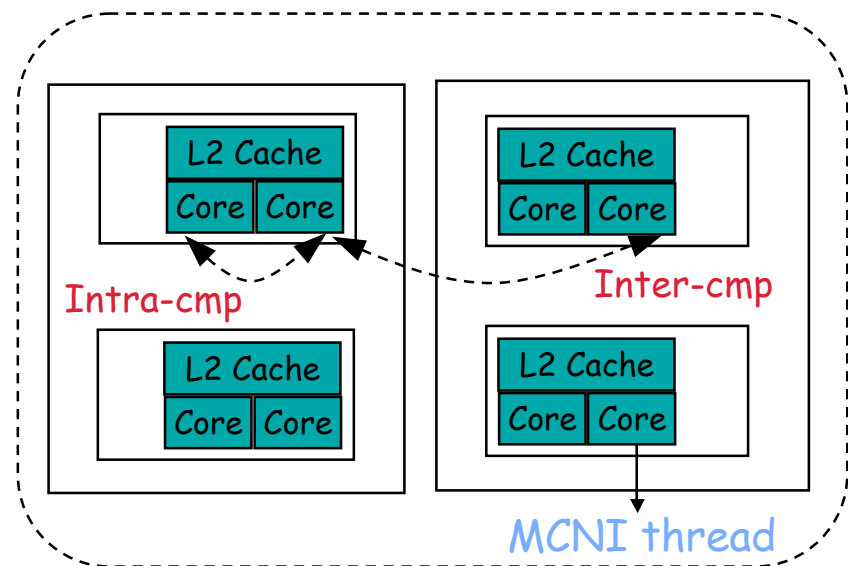
Analysis of the Cache Aware Design

- Lock free
- In-order Message Transfer
 - Control messages are going through receive buffers
- Efficient in cache utilization
 - Small messages: small receive buffer, likely in the cache
 - Large messages: chances of buffer reuse improved
- Efficient memory usage
 - Receive buffers become smaller
 - Large message buffers are shared among all the connections

L. Chai, A. Hartono and D. K. Panda, *Designing High Performance and Scalable MPI Intra-node Communication Support for Clusters*, IEEE International Conference on Cluster Computing (Cluster 2006), September 2006.

Multi-core Aware Intra-node Communication Design

- Dedicate a processing core to perform memory copy (MCNI)
- Perform the copy operation directly between two processes instead of shared memory approach for medium and large messages.
- Potential Benefits:
 - Elimination of the intermediate copy
 - Asynchronous memory copy



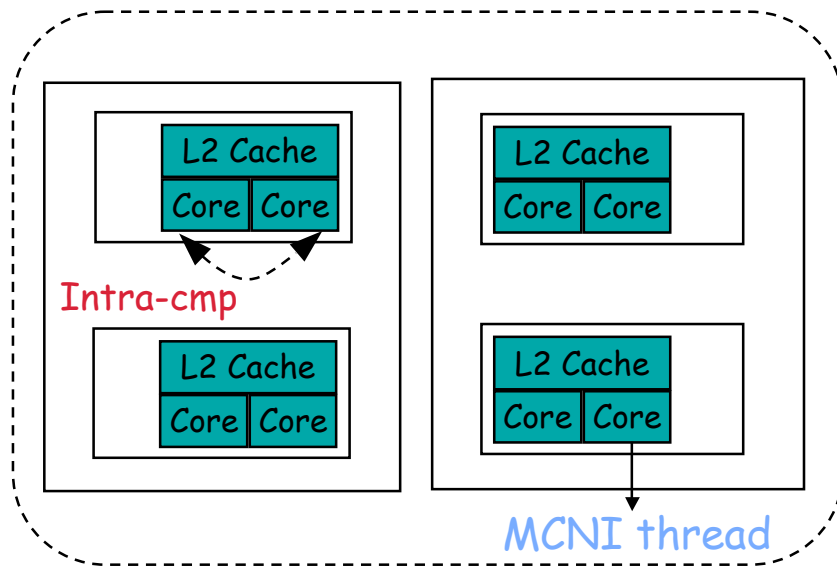


Experimental Setup

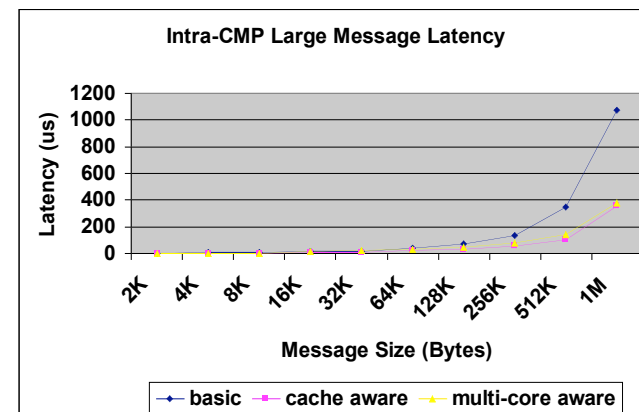
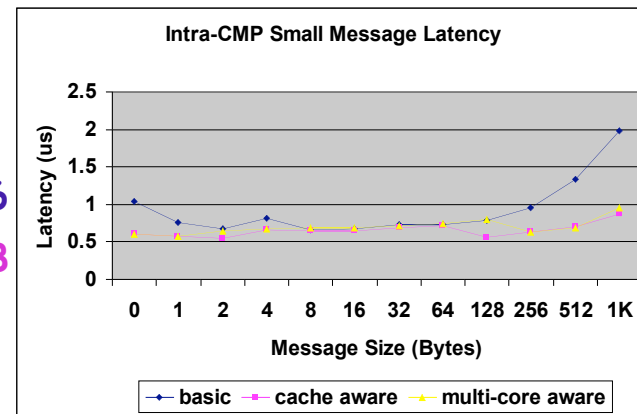


- Intel Clovertown:
 - Quad-core Clovertown processors
 - Two sockets per node -> 8 cores/node
 - 2.33GHz
 - Shared 4MB L2 cache

Intra-CMP Latency

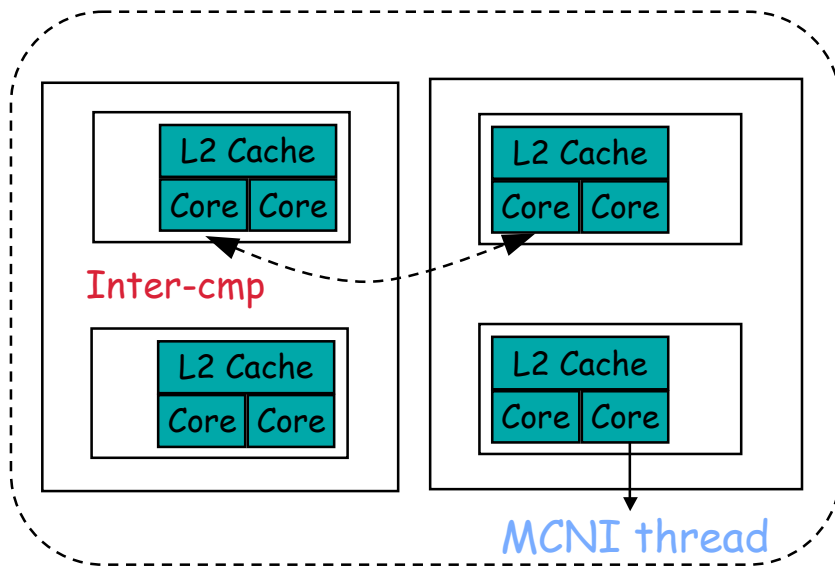


0.76
0.58

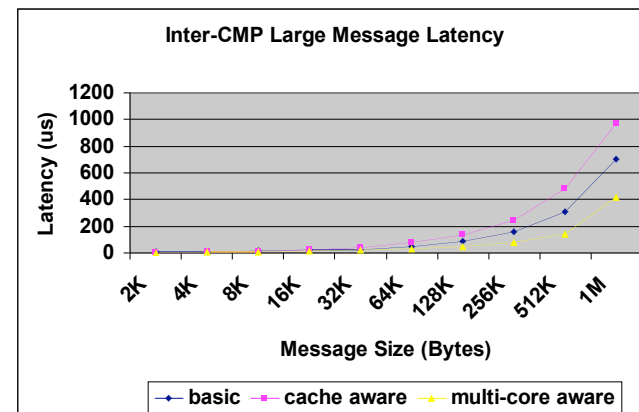
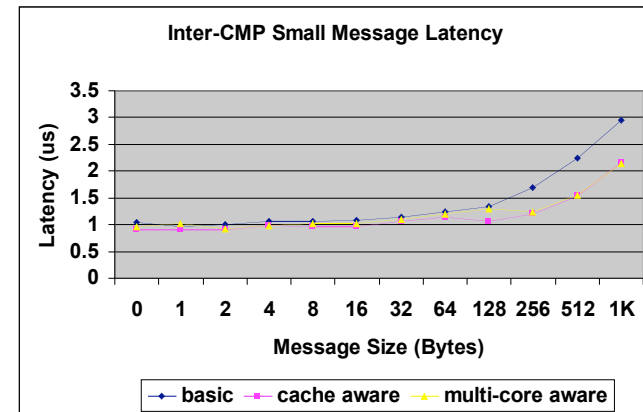


- Cache aware and multi-core aware design improves intra-cmp latency by around 70%

Inter-CMP Latency

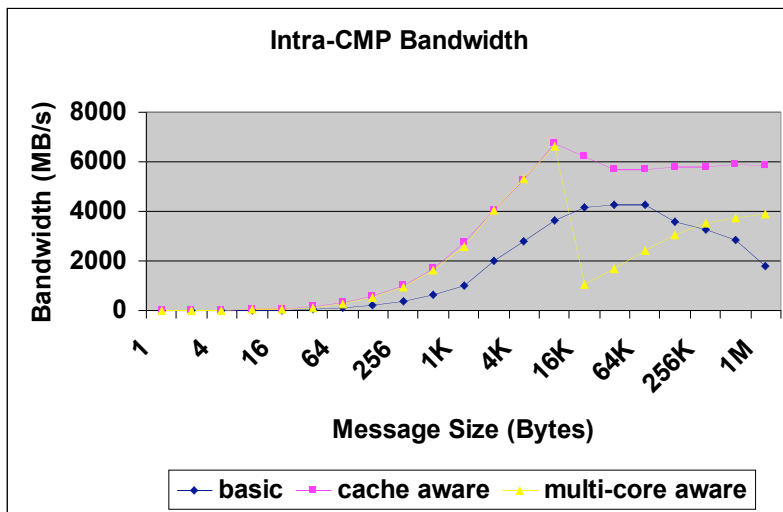


1.0
0.9

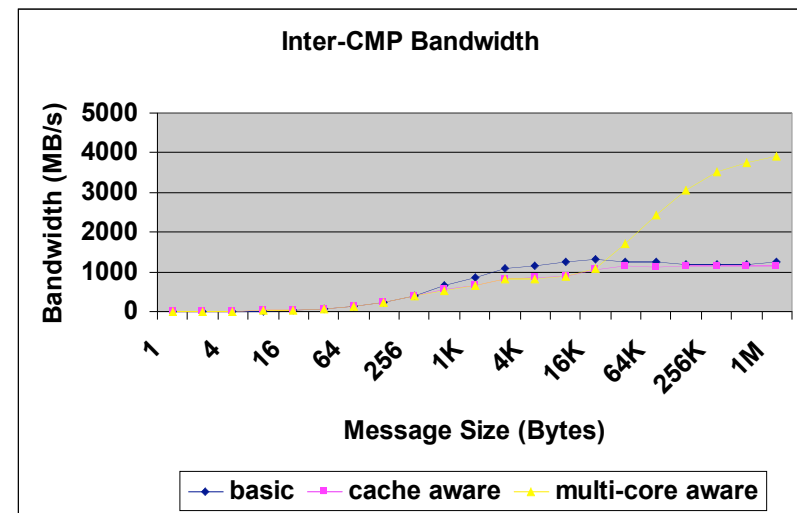


- Multi-core aware design improves inter-cmp latency by around 50%

Bandwidth Single-Pair (Intra-CMP and Inter-CMP)



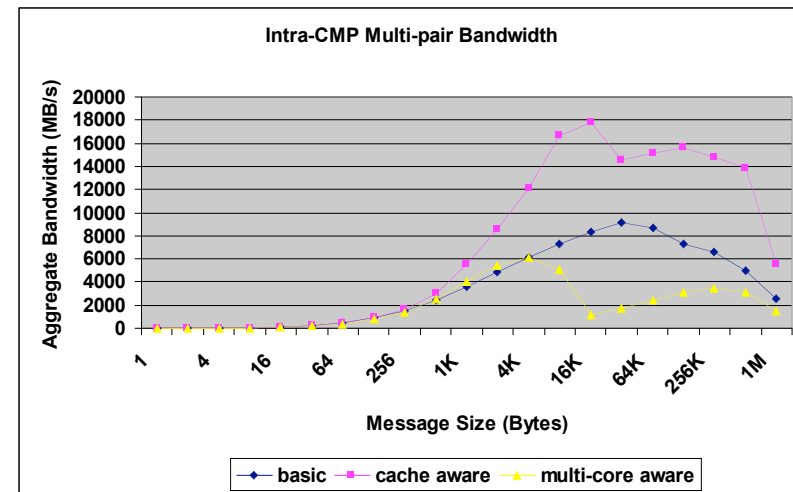
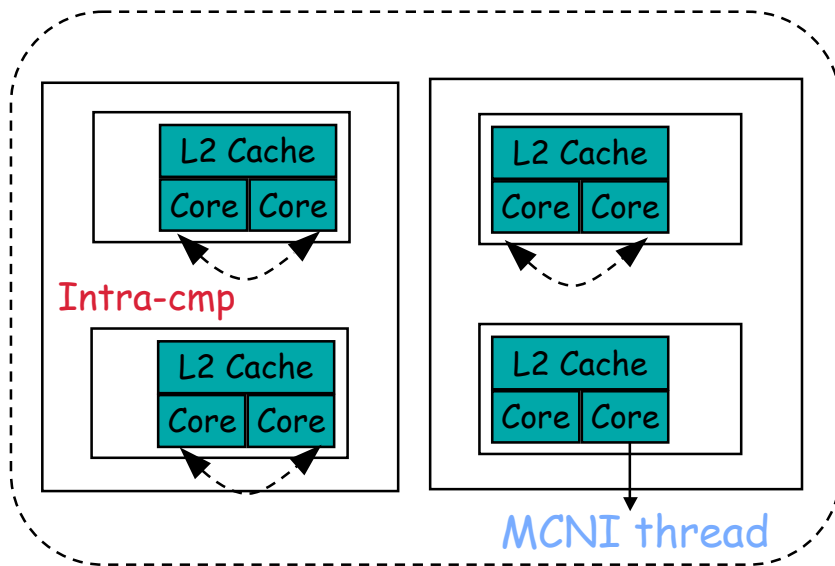
6730
4286



3910
1303

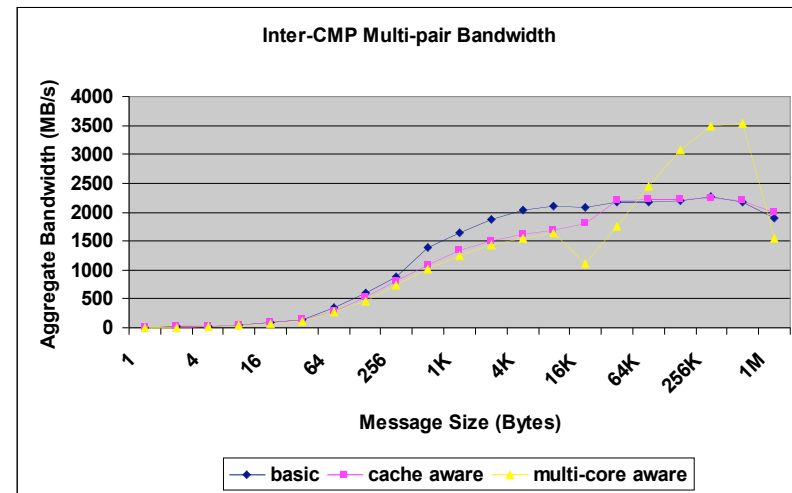
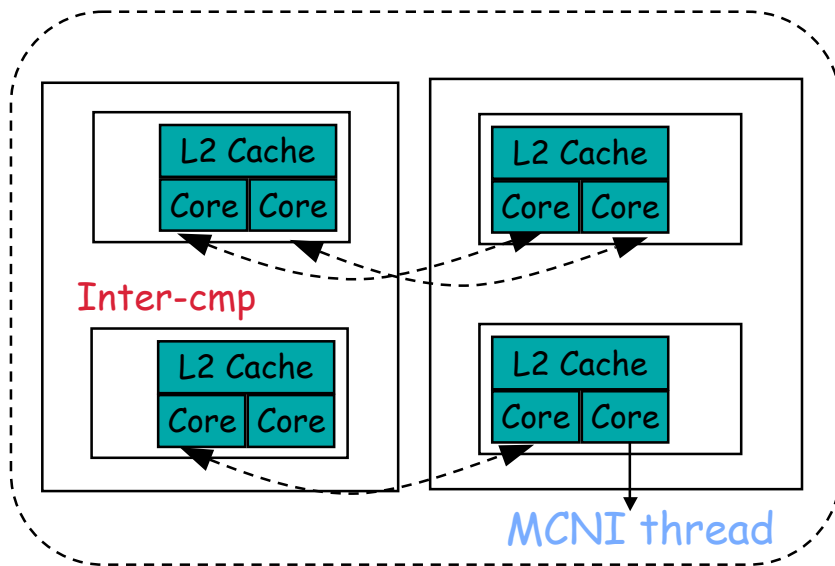
- Cache aware design improves intra-cmp peak bandwidth by 1.5 times
- Multi-core aware design improves inter-cmp peak bandwidth by 3.0 times

Intra-CMP Multi-pair Bandwidth



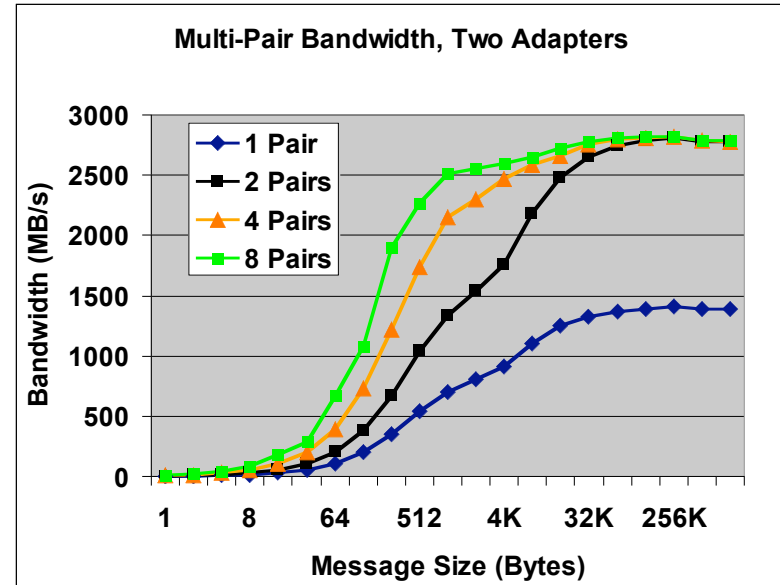
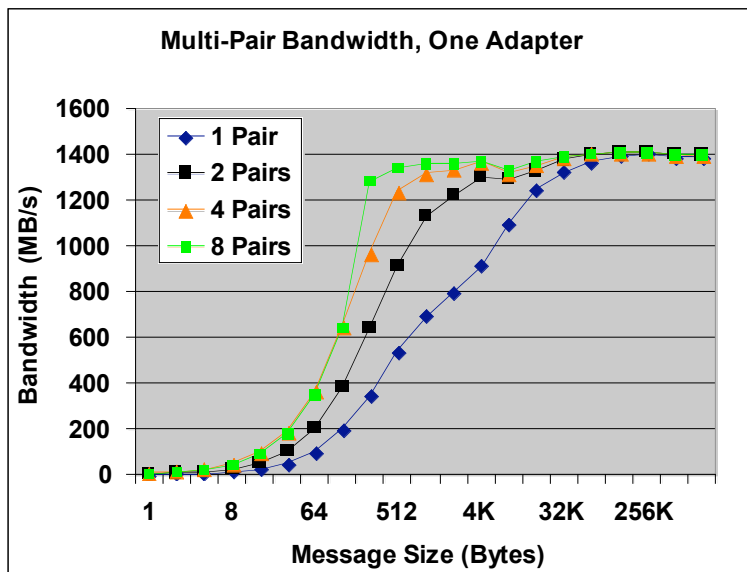
- Cache aware design almost doubles the intra-cmp multi-pair peak bandwidth

Inter-CMP Multi-pair Bandwidth



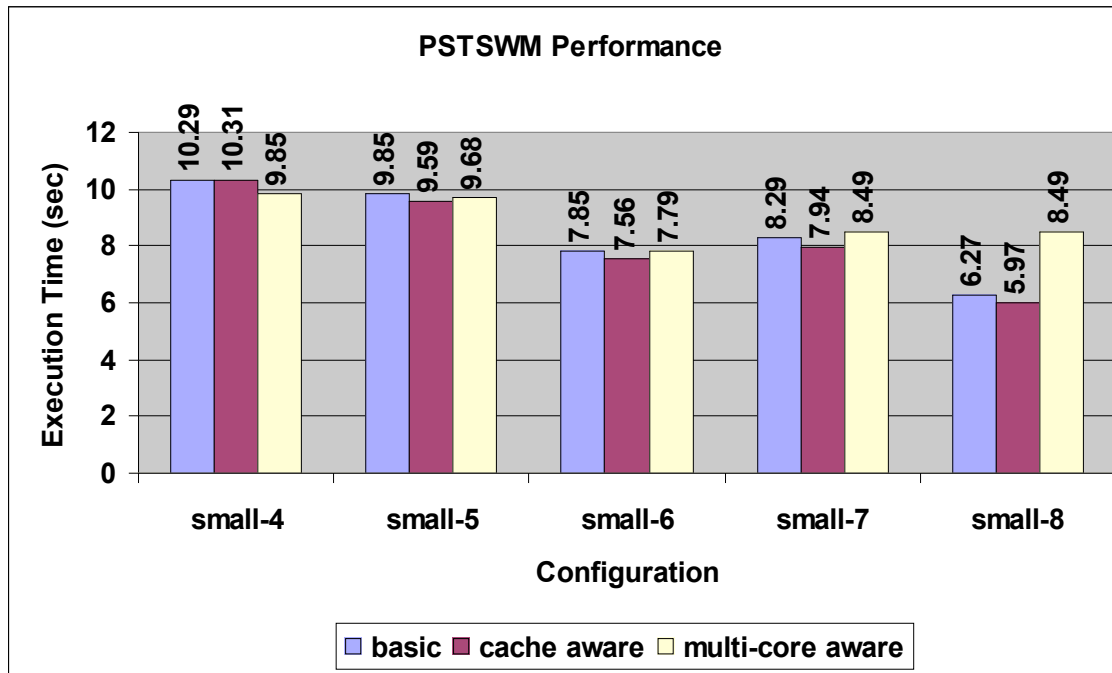
- Multi-core aware design improves inter-cmp multi-pair peak bandwidth by 56%

Inter-node Multi-Pair Bandwidth



- Memory bandwidth scales very well for small-medium messages with increasing number of pairs
- Linear bandwidth scaling is achieved for multiple rails with multiple processes.

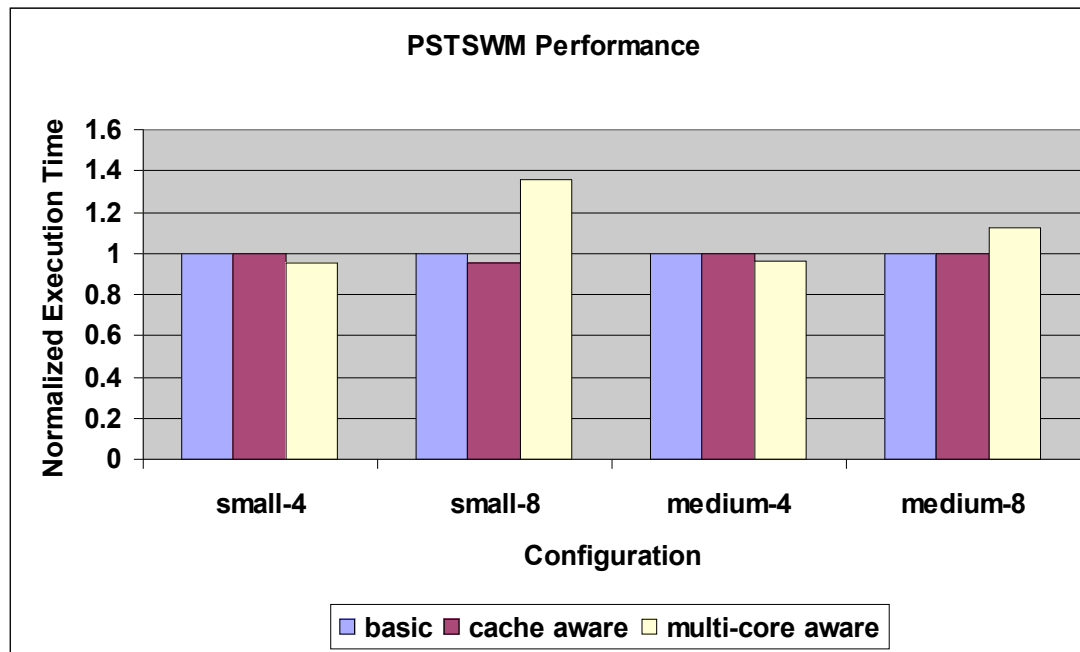
Application-Level Performance



Parallel Spectral
Transform
Shallow Water
Model (PSTSWM)

- Cache aware design improves PSTSWM performance by up to 5%
- Multi-core aware design improves PSTSWM performance for four processes by 4.2%

Application Performance (Cont'd)



- Multi-core aware design improves PSTSWM performance when there is an extra core dedicated for intra-node communication



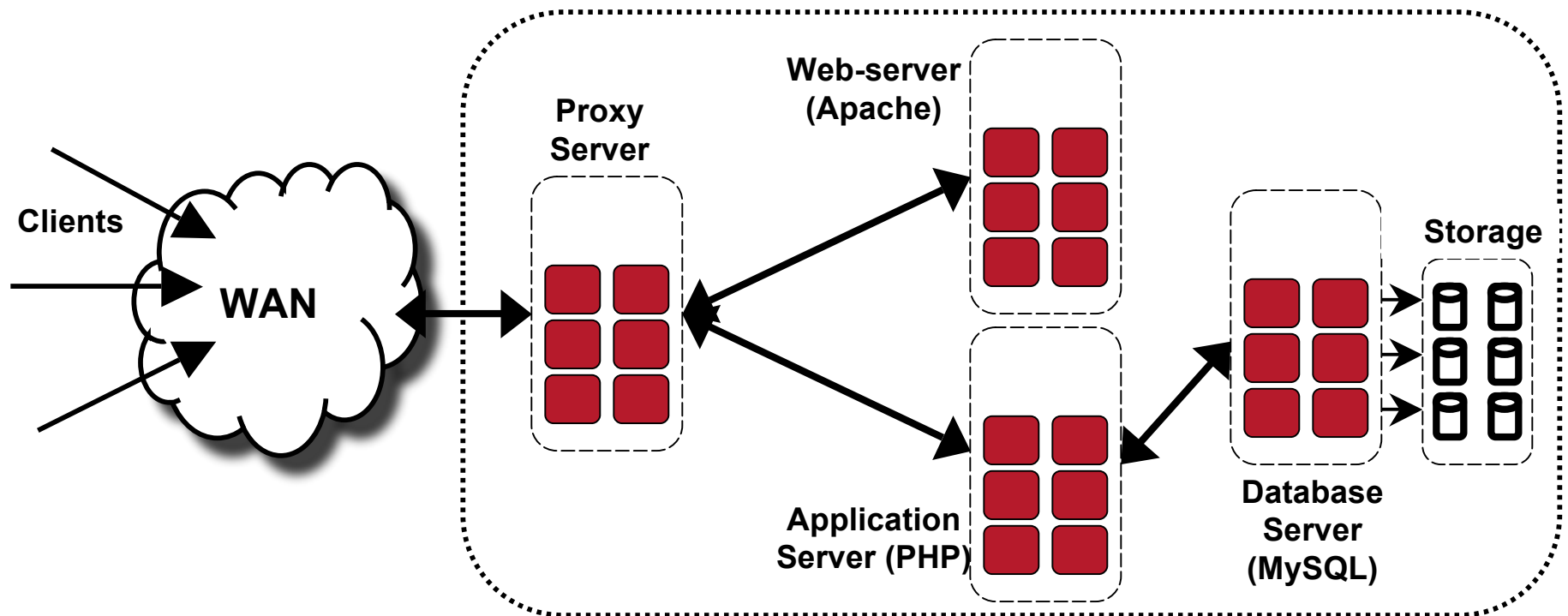
Presentation Outline



- Introduction
- Multi-core Aware Middleware
- Designs for MPI and Evaluation
- **Designs for Data-Centers and Evaluation**
- Conclusions and Future work

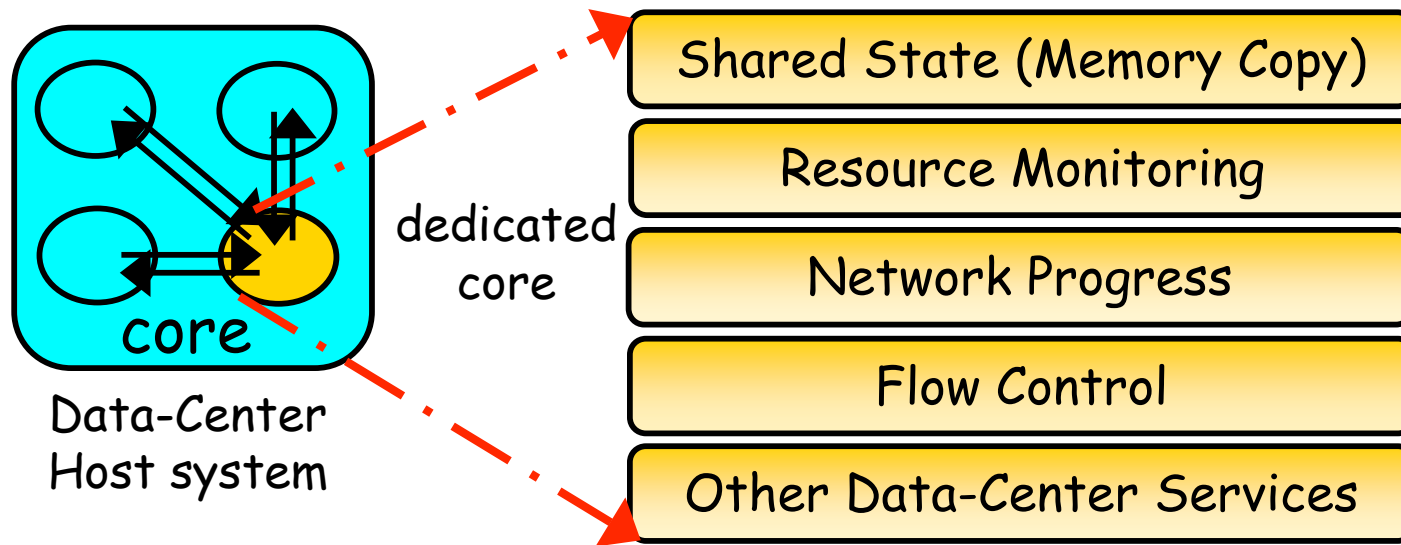


Typical Multi-Tier Data-Center Environment



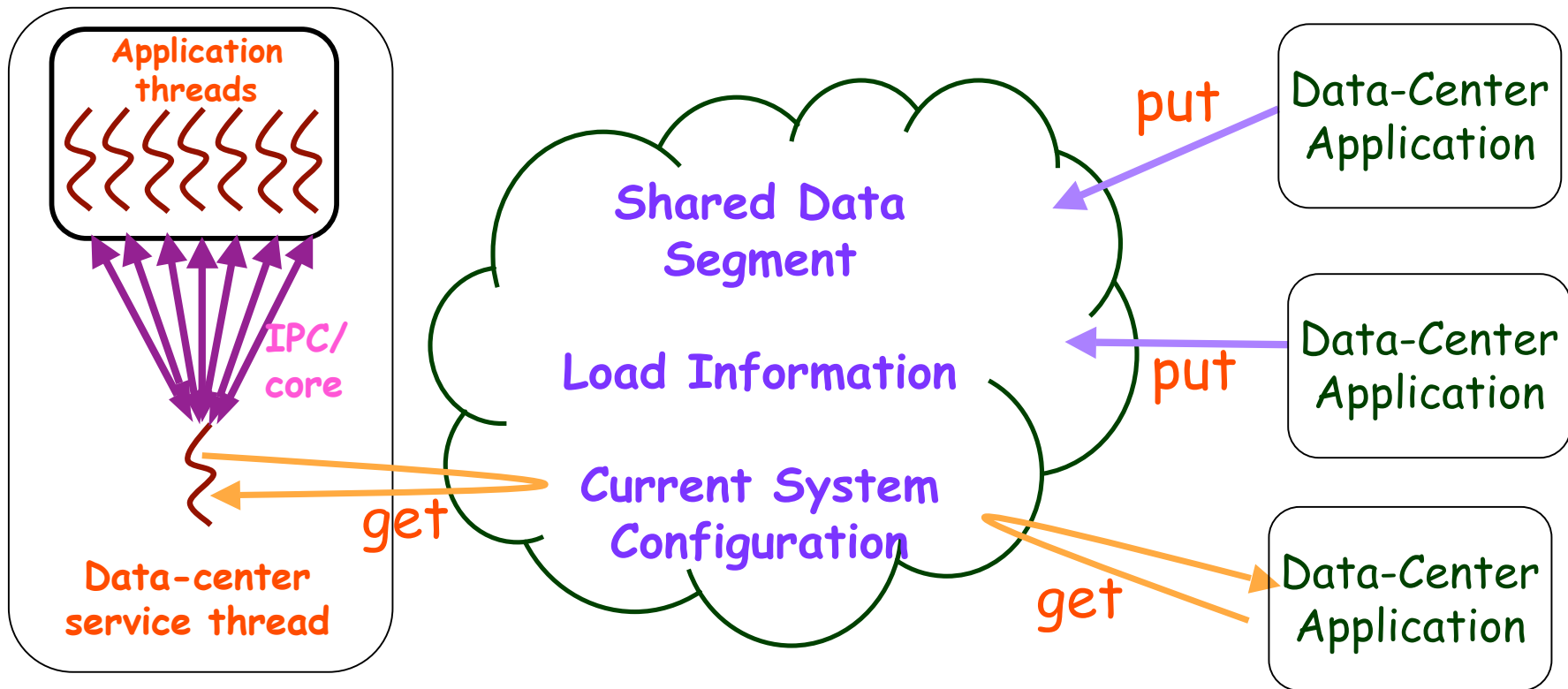
- Requests are received from clients over the WAN
- Proxy nodes perform caching, load balancing, resource monitoring, etc.
- Application server performs the business logic (CGI, Java servlets, etc.)
 - Retrieves appropriate data from the database to process the requests

Onloading Data-Center Services



- Dedicated Cores provide asynchronous progress for several data-center services and primitives

Soft Shared State



- Synchronization cost between application thread and data-center service thread can be reduced
- Can save a memory copy between application thread and data-center service thread



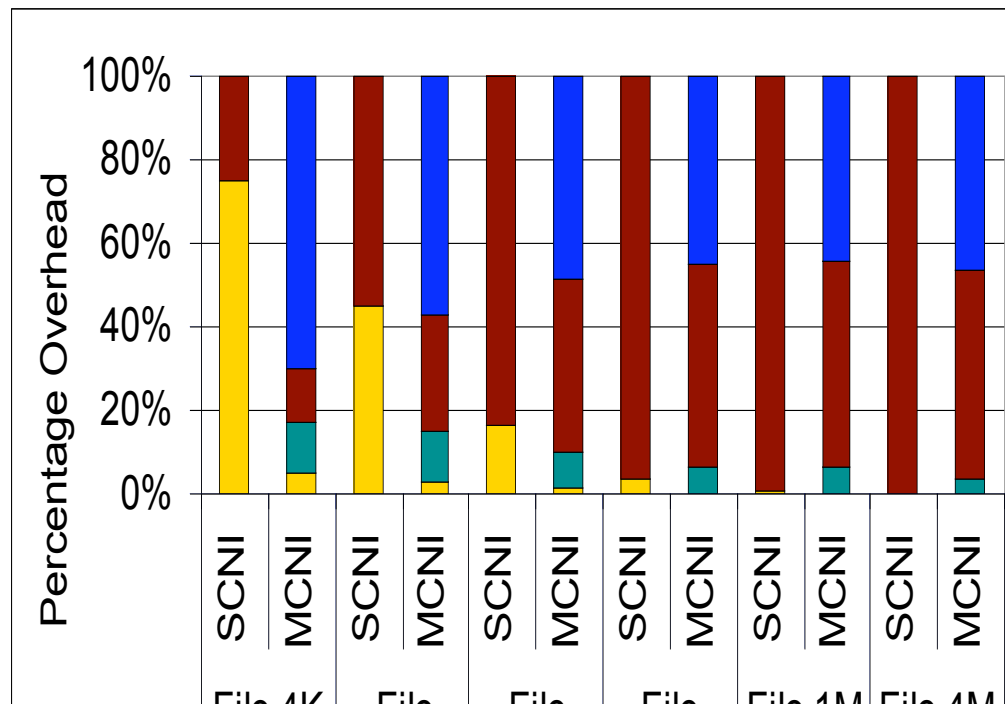
Multi-core Aware Soft Shared State



- Two approaches
 - Soft Shared State with memcpy based implementation (SCNI)
 - Soft Shared State with onloaded memcpy (MCNI)
- Potential benefits:
 - Save the intermediate copy
 - Overlap of memory copy with computation
 - Reduced Synchronization Cost (no need for IPCs)

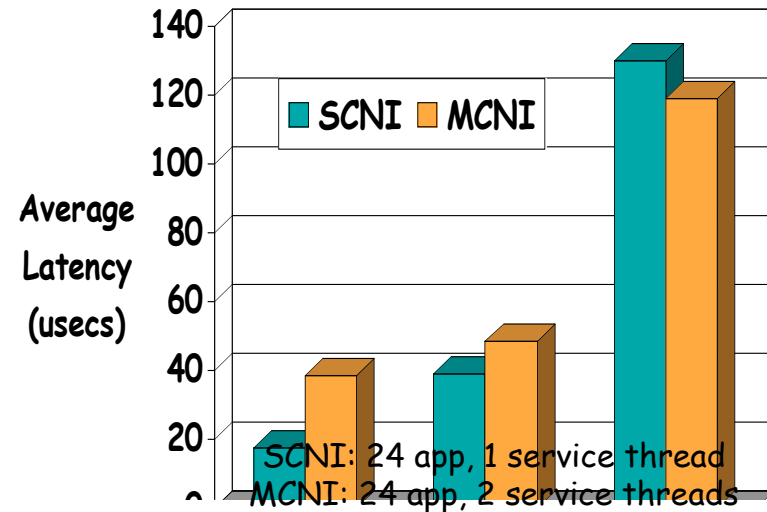
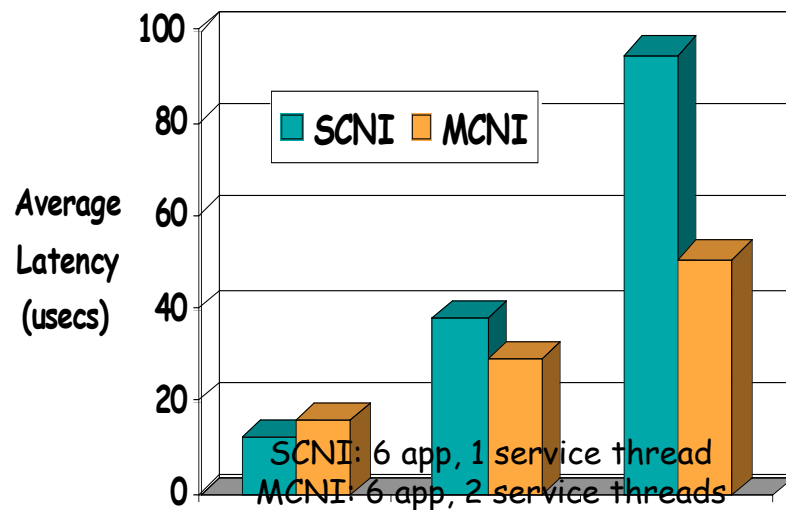


Overhead of MCNI and SCNI in Data-Centers



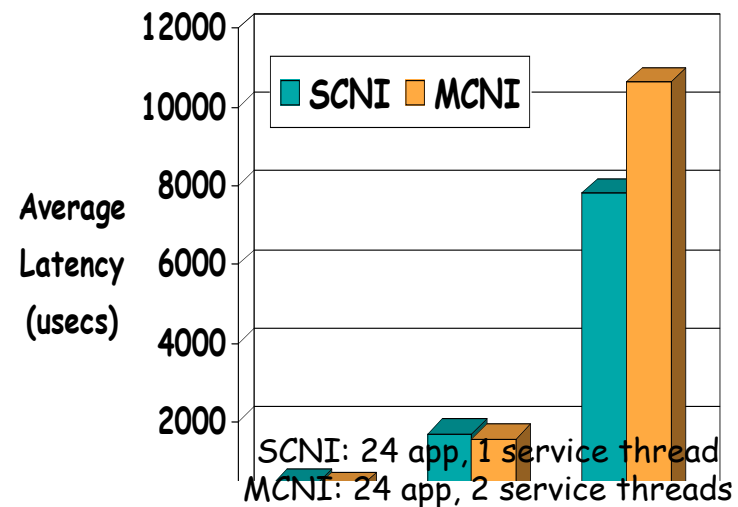
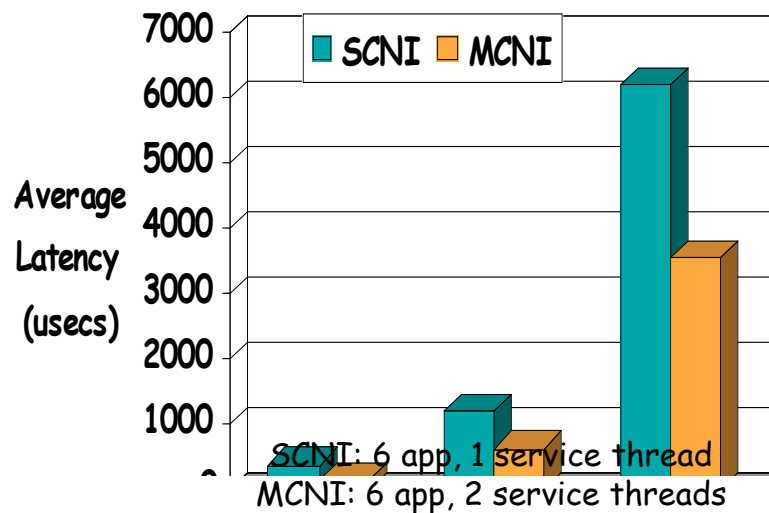
- SCNI requires synchronization overhead (IPC) to communicate between processes whereas MCNI can communicate directly
- MCNI's savings is close to 50% due to reduced memory copies

Performance with Small File Workloads



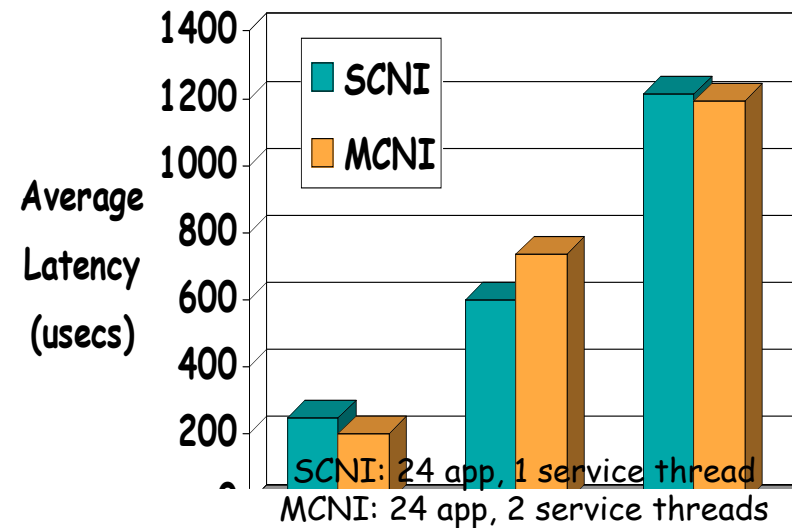
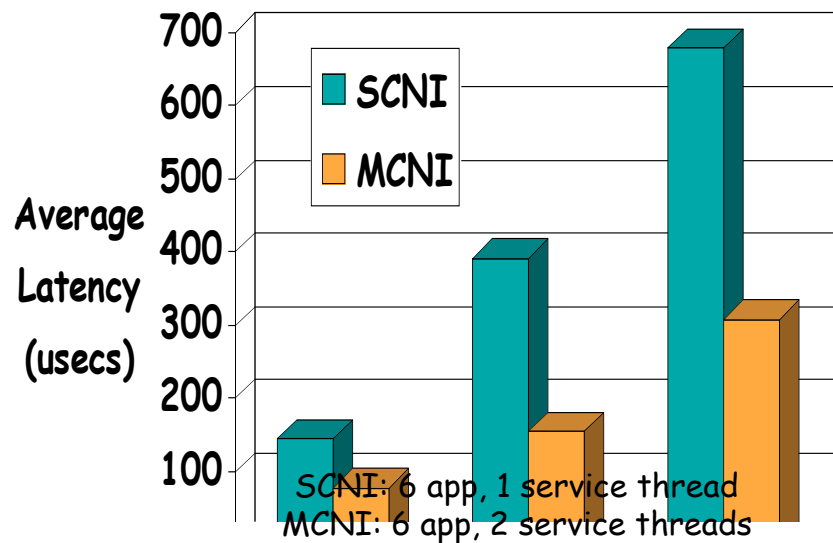
- MCNI improves the performance from file size 16KB giving up to 46% improvement for file size 64KB with lesser number of threads
- With large number of threads, performance degrades with MCNI

Performance with Large File Workloads



- MCNI improves the performance by 50% with lesser number of threads
- With large number of threads, MCNI performance degrades significantly

Performance with Zipf Workloads



- MCNI shows more than 60% improvement with lesser number of threads
- For large number of threads, MCNI shows no improvement



Presentation Outline



- Introduction
- Multi-core Aware Middleware
- Designs for MPI and Evaluation
- Designs for Data-Centers and Evaluation
- Conclusions and Future work





Conclusions and Future Work



- Multi-core Aware Designs for Middleware can benefit applications
 - Provided the dedicated cores do not become the bottleneck
 - May need heterogeneous cores with dedicated functionalities (different than the cores for processing)
 - Multiple network rails may be needed in large-scale multi-core clusters for balancing inter-node and intra-node communication performance
- More studies
 - Dedicated cores for MPI collectives, progress engine, etc.
 - Dedicated cores for other data-center services, network progress, I/O, etc.
 - Interaction with new technologies such as I/OAT
- Requires exploration of these designs and evaluations on large-scale HPC and Data-Centers



Acknowledgments

- Current Students
 - **Lei Chai (PhD)**
 - Rahul Kumar (MS)
 - Qi Gao (PhD)
 - Wei Huang (PhD)
 - **Matthew Koop (PhD)**
 - Ping Lai (PhD)
 - Amith Mamidala (PhD)
 - Sundeep Narravula (PhD)
 - Ranjit Noronha (PhD)
 - G. Santhanaraman (PhD)
 - Sayantan Sur (PhD)
 - **K. Vaidyanathan (PhD)**
 - **Abhinav Vishnu (PhD)**
- Current Programmers
 - Shaun Rowland
 - Jonathan Perkins
- Past Students
 - Pavan Balaji (PhD)
 - Sitha Bhagvat (MS)
 - D. Buntinas (PhD)
 - B. Chandrasekharan (MS)
 - Weihang Jiang (MS)
 - Sushmita Kini (MS)
 - S. Krishnamoorthy (MS)
 - Jiuxing Liu (PhD)
 - Jiesheng Wu (PhD)
 - Weikuan Yu (PhD)

Acknowledgements

Our research is supported by the following organizations

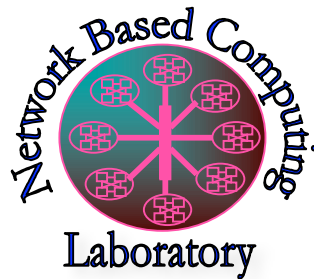
- Current Funding support by



- Current Equipment support by



Web Pointers



<http://www.cse.ohio-state.edu/~panda/>
<http://nowlab.cse.ohio-state.edu/>

MVAPICH Web Page

<http://nowlab.cse.ohio-state.edu/projects/mipi-iba/>

E-mail: panda@cse.ohio-state.edu

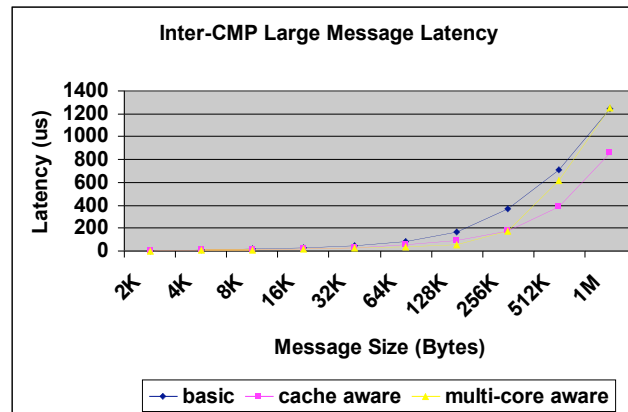
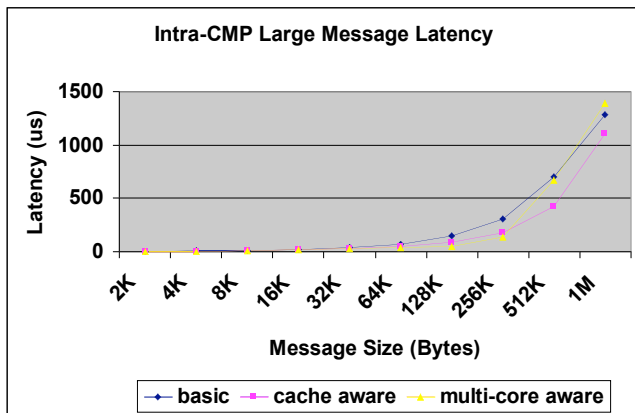
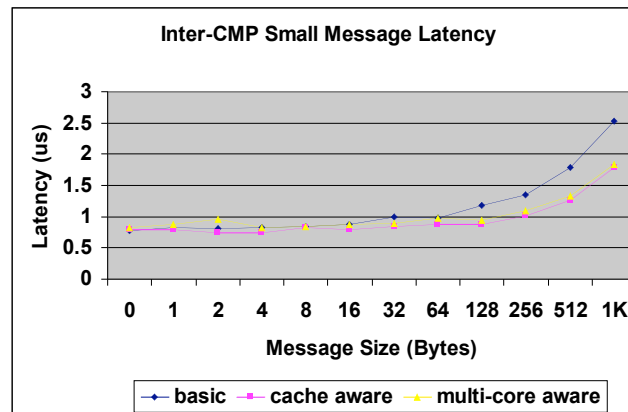
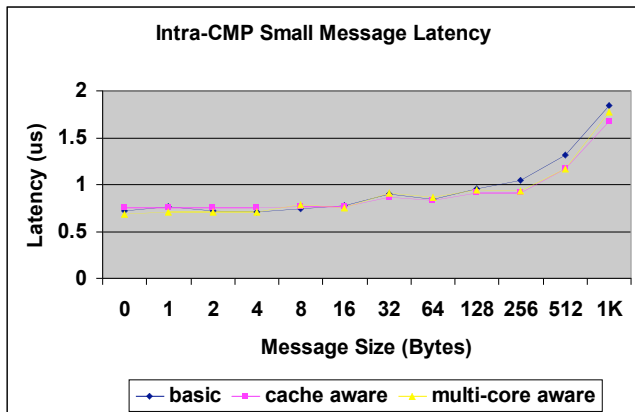


Experimental Setup



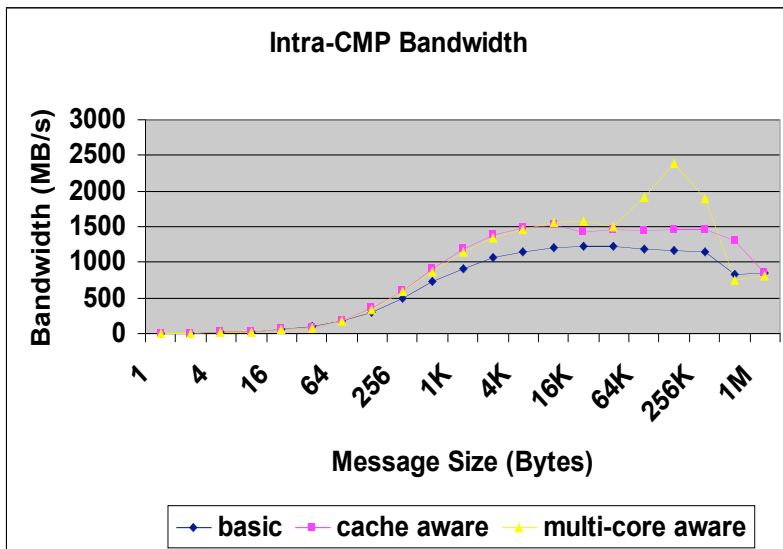
- Platform 1 - Clovertown:
 - Quad-core Clovertown systems
 - 8 cores/node
 - 2.33GHz
 - Shared 4MB L2 cache
- Platform 2 - Opteron:
 - Dual-core Opteron systems
 - 8 cores/node
 - 2.4GHz
 - Non-shared 1MB L2 cache

Latency on Dual-core Opteron

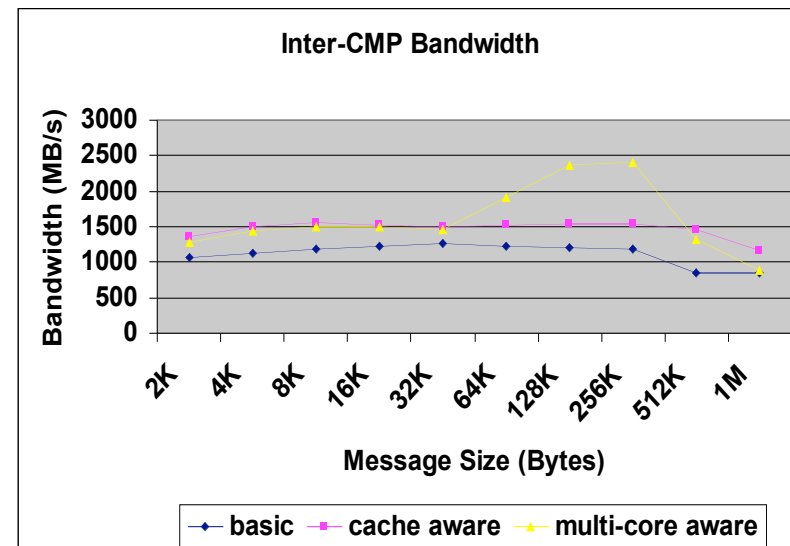


- Cache aware design improves intra-cmp latency by up to 40% and inter-cmp latency by up to 45%
- Multi-core aware design improves intra-cmp and inter-cmp latency by up to 45%

Bandwidth on Dual-core Opteron




2383
1521
1226

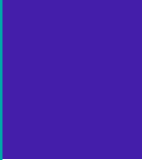


2413
1561
1256

- Cache aware design improves intra-cmp and inter-cmp peak bandwidth by 25%
- Multi-core aware design improves intra-cmp and inter-cmp bandwidth by around 90%



Limitations of Current Data-centers



- **Communication Requirements**
 - TCP/IP used even in the data-center: Sub-optimal performance
 - InfiniBand and other interconnects provide more features



High Performance Sockets (e.g., SDP)

- Superior performance with no modifications

- **Advanced Data-center Services**



Minimize the computation requirements

- Improved caching of documents
- Issues with caching Dynamic (or Active) Content



Maximize compute resource utilization

- Efficient resource monitoring and management
- Issues with heterogeneous load characteristics of data-centers



Our Proposed Architecture

