

MPI Intra-node Communication & Multi-core Study

Ping Lai
02/07/2008

Presentation Outline

- MPI Intra-node communication
 - Shared memory: user space design (4h)
 - LiMIC: kernel space design (4j)
- Multi-core study (4i)

MPI Intra-node communication

- Motivation
 1. Large SMP system has been available
 2. Multi-core systems available

The diagram illustrates inter-node communication. A central 'Network' connects three node types: a 'Bus-based SMP Node' with multiple CPUs and shared memory, a 'Dual Core Chip' with two cores and shared memory, and a 'NUMA Node' with multiple CPUs and private memory. Arrows indicate 'Inter-node Communication' and 'SMP Extra-node Communication' between these nodes.

MPI Intra-node communication

- Generic approaches
 1. NIC based loop back

This diagram shows a 'NIC based loop back' approach. Node 0 and Node n are connected via a 'Switch'. Node 0 contains processors P0-P3, buffers for P0 and P3, and memory. A 'NIC' is connected to the switch. Red arrows labeled 'IO Bus Transactions' show data flow from Node 0's buffers through the NIC and switch to Node n's memory, and back to Node 0's buffers.

MPI Intra-node communication

- Generic approaches
 2. User space shared memory

This diagram illustrates 'User space shared memory'. Node 0 and Node n are connected via a 'Switch'. Node 0 has processors P0-P3, buffers for P0 and P3, and memory. A 'NIC' is connected to the switch. Red arrows labeled 'Two Memory Copy Operations' show data flow from Node 0's buffers through the NIC and switch to Node n's memory, and back to Node 0's buffers.

MPI Intra-node communication

- Generic approaches
 3. Kernel-based memory mapping

This diagram shows 'Kernel-based memory mapping'. Node 0 and Node n are connected via a 'Switch'. Node 0 has processors P0-P3, buffers for P0 and P3, and memory. A 'NIC' is connected to the switch. Red arrows labeled 'One Memory Copy' show data flow from Node 0's buffers through the NIC and switch to Node n's memory, and back to Node 0's buffers.

MPI Intra-node communication --User space design

Designing High Performance and Scalable MPI Intra-node Communication Support for Clusters

Lei Chai Albert Hartono Dhabaleswar K. Panda

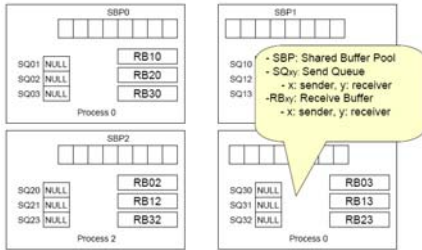
Original Design in MVAPICH



RBxy: Receive buffers
 -- buffer shared between each pair of sender and receiver; x: sender, y: receiver
 Disadvantage: large memory usage ($P*(P-1)*\text{buffersize}$); inefficient cache utilization

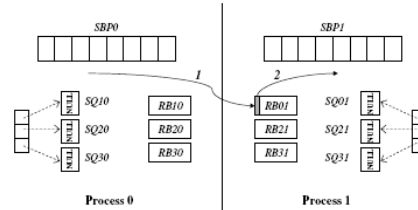
New Design

Data Structure



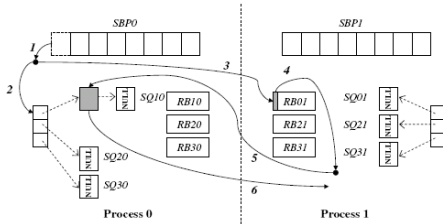
New Design

Algorithm: small message transfer



New Design

Algorithm: large message transfer



New Design

Advantages

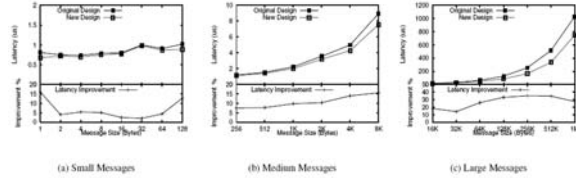
- Efficient memory usage
 - Receive buffers become smaller
 - Large message buffers are shared among all the connections
- Efficient cache utilization
 - Small message: small receive buffer
 - Large message: improved chances of buffer reuse

New Design

- Experiment setup
 1. NUMA cluster
 - Two nodes connected by InfiniBand
 - Each node has four AMD Opteron processors, 2.0 GHz
 - 1MB L2 cache
 2. Multi-core Cluster
 - Two nodes connected by InfiniBand
 - Each node has four dual-core AMD Opteron processors, 2.0 GHz
 - Each core has 1MB L2 cache

New Design

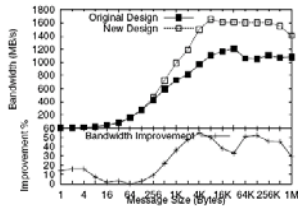
- Latency on NUMA cluster



- small and medium messages: improved by 15%
- large messages: improved by 35%

New Design

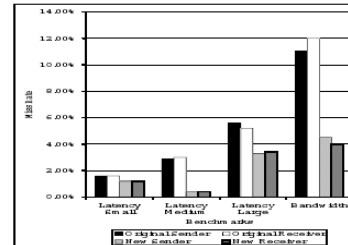
- Bandwidth on NUMA cluster



- improved by up to 50%

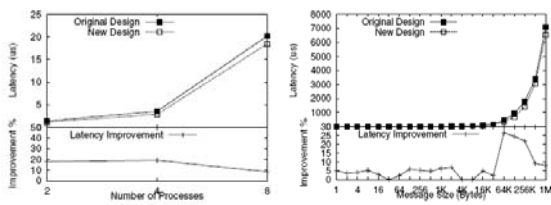
New Design

- L2 Cache Miss on NUMA cluster



New Design

- Collectives on NUMA cluster

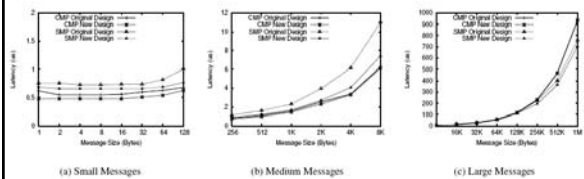


MPI_Barrier

MPI_Alltoall

New Design

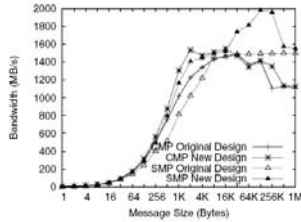
- Latency on Multi-core cluster



- CMP latency is lower than SMP latency for small messages, but not for large messages
- SMP latency is improved for all the messages
- CMP latency is improved for small messages

New Design

Bandwidth on multi-core cluster



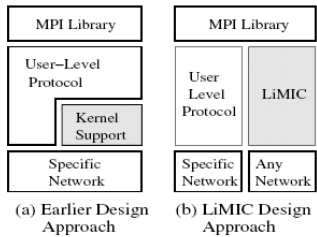
- SMP is improved significantly
- CMP is also improved for small and medium messages

MPI Intra-node communication --Kernel space design

LiMIC: Support for High-Performance MPI Intra-Node Communication on Linux Cluster

Hyun-wook Jin Sayantan Sur Lei Chai
Dhaleswar K. Panda

Design Comparison



No portability to other user-level protocol
No optimization-space for MPI library developer

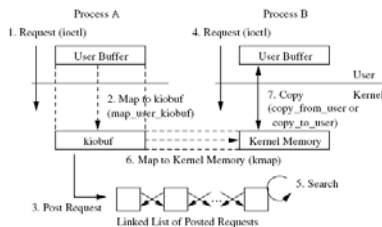
Has portability and optimization space

Implementation

- Kernel module based design
 - No kernel modification
 - Portable across major Linux versions
- Portable and MPI friendly interface
 - Independent interface on communication library and MPI implementation
 - Interfaces: LiMIC_Isend(dest, tag, context_id, buf, len, req); LiMIC_Irecv(dest, tag, context_id, buf, len, req); LiMic_Wait(src/dest, req)
 - Interfaces trap into the kernel internally by calling ioctl() system call

Implementation

Memory mapping mechanism



Implementation

- Copy mechanism
 - Copy on send and receive calls
 - better progress
 - less resource usage
- MPI Message matching based on source
 - source on the same node
 - source on a different node
 - source on the same node and MPI_ANY_TAG
 - MPI_ANY_SOURCE and MPI_ANY_TAG

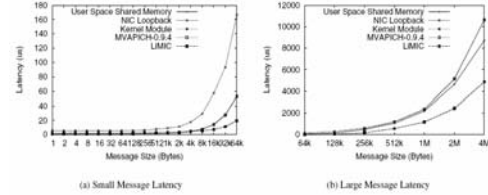
Performance

■ Experiment setup

- Cluster A
 - 8 nodes with dual Intel xeon 3.0GHz processor and 512KB L2 cache
- Cluster B
 - 8 nodes with dual Intel Xeon 2.4GHz processor and 512KB L2 cache

Performance

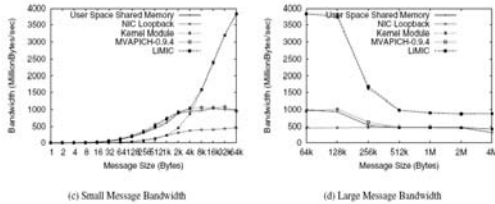
■ Microbenchmark Latency



- For small message, user space shared memory is better
- For large messages, LiMIC is better

Performance

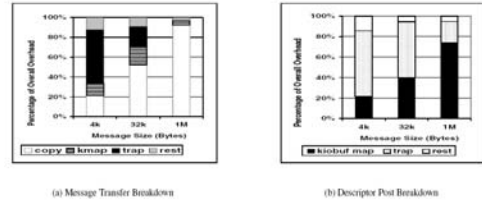
■ Microbenchmark Bandwidth



Same observation as latency performance

Performance

■ Cost breakdown



1. For small message, kernel trap overhead is dominant
2. For large message, copy overhead is dominant

Performance

■ NAS Integer Sort

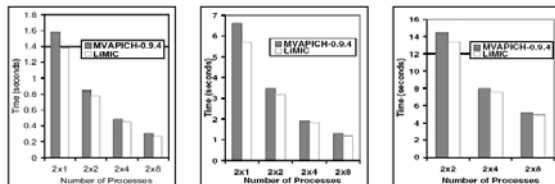


Figure 6. IS Total Execution Time Comparisons: (a) Class A, (b) Class B, and (c) Class C

The improvement for 2*8 configuration can be 10%, 8% and 5% respectively.

Multi-core Study

Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System

Lei Chai Qi Gao Dhableswar K. Panda

Introduction

- Multi-core processor
 - Two or more processing cores on the same chip
 - Dividing workload to different cores
 - Chip-level Multi-Processor (CMP)
 - Independent or shared L2 cache
 - e.g.: Intel Clovertown processor has shared L2 cache
 - AMD dual-core Opteron has separate L2 cache
- Multi-core cluster
 - Intra-cmp communication, Inter-cmp communication, inter-node communication

Motivation & Methodology

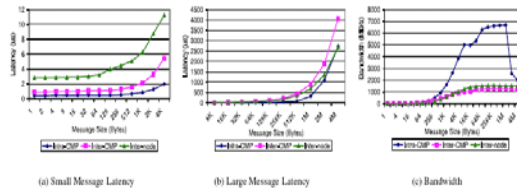
- Addressing three problems
 - What are the application communication characteristics in multi-core clusters?
 - What are the potential bottlenecks in multi-core clusters and how to possibly avoid them?
 - Can multi-core clusters scale well?
- Programming model
 - MVAPICH2 using memory copy for intra-node communication

Evaluation Platform

- Intel Bensley cluster
 - Dual-core Woodcrest processors, 2.6GHz
 - 4 processors per node, 4 nodes connected by InfiniBand
 - 2 cores on the same chip share a 4MB L2 cache
- Single-core Cluster
 - Intel Xeon processors, 3.6GHz
 - 2 processors per node, 32 nodes connected by InfiniBand
 - Each processor has 2MB L2 cache

Results & Analysis

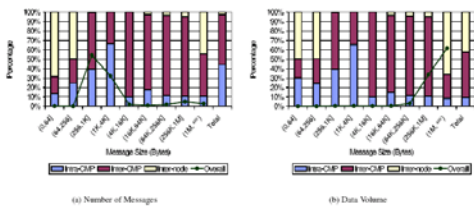
- Microbenchmark latency and bandwidth



- Intra-CMP performance is far better than inter-CMP and inter-node
- Inter-CMP performance is not as good as inter-node performance with large messages

Results & Analysis

- Message distribution in HPL



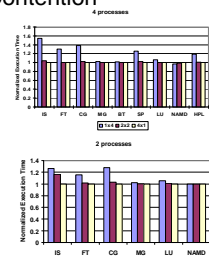
If communication is evenly distributed, then
 intra-cmp = 1/15 ~ 6.7%
 inter-cmp = 2/15 ~ 13.3%
 Inter-node = 12/15 = 80%

- Intra-node traffic is well above that in even distribution
- Optimizing MPI intra-node communication in multi-core clusters is important!

Results & Analysis

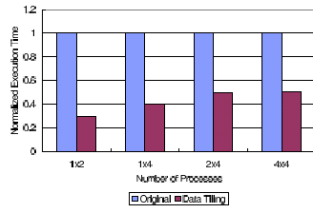
- Potential Cache/Memory Contention

- Configurations
 - 4 processes
 - 1x4, 2x2, 4x1
 - 2 processes
 - Intra-cmp, inter-cmp, inter-node
- Performance
 - 1x4 <= 2x2 ~ 4x1
 - Intra-cmp <= inter-cmp ~ inter-node
- Potential bottleneck
 - Shared L2 cache and memory controller
- Applications need to be multi-core aware



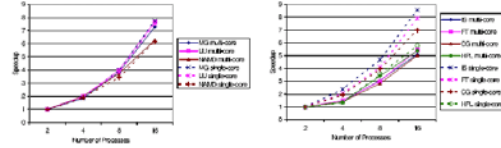
Results & Analysis

■ Benefit of data tiling



Results & Analysis

■ Scalability



- Some applications achieve almost ideal scalability on the multi-core cluster
--MG, LU, NAMD
- Some applications do not scale as well as on the single-core cluster
--IS, FT, CG, HPL
- Multi-core cluster scalability is promising
--Multi-core aware applications

Thank you !!