Performance Analysis of MPI Communications on the SGI Altix 3700

Nor Asilah Wati Abdul Hamid,
Paul Coddington, Francis Vaughan

Distributed & High Performance Computing Group
School of Computer Science, University of Adelaide
Adelaide SA 5005, Australia
Email: paulc@cs.adelaide.edu.au

September 2005

THE UNIVERSITY OF ADELAIDE AUSTRALIA

Motivation

- A 1672 CPU SGI Altix (APAC AC) is the new APAC peak facility, replacing an AlphaServer SC with Quadrics network (APAC SC).
- Most parallel programs run on APAC machines use MPI.
- Want to measure MPI performance on APAC machines.
- Several MPI benchmark software available to measure MPI communications performance, ie: Pallas MPI Benchmark (PMB), SKaMPI, MPBench, Mpptest, and recently developed MPIBench.
- The measurements reported here used our benchmark, MPIBench.
- So, we aim to :-
  - Measure the MPI performance of the SGI Altix
  - Compare the Altix results with Alphaserver SC with Quadrics network.
  - Compare the results of MPIBench with other MPI Benchmarks.

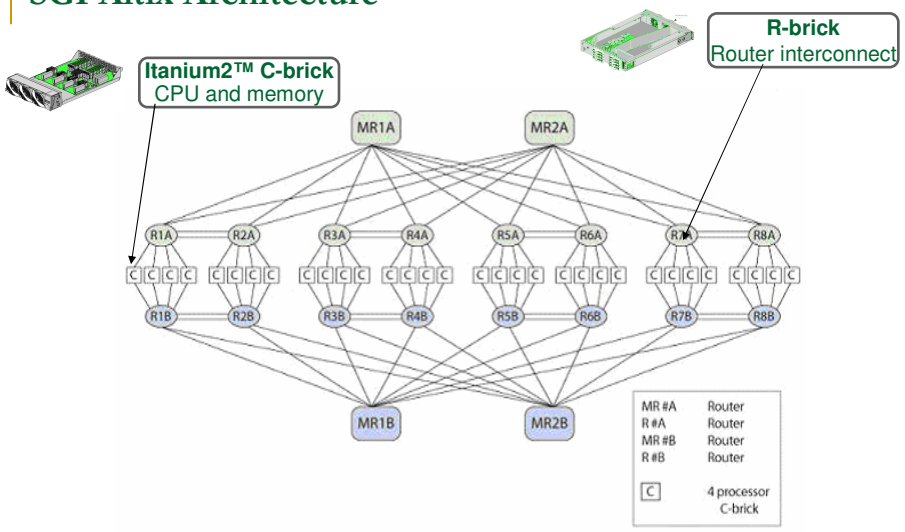
MPIBench

- Current MPI benchmarks (MPBench, MPPtest, SkaMPI, Pallas) have several limitations.
- MPIBench is a communications benchmark program for accurately measuring the performance of MPI routines.
- Uses a highly accurate, globally synchronized clock
  - so can measure individual message times, not just average over many ping-pongs.
- Gives probability distributions (histograms) of times
- Makes point-point measurements across many nodes
  - see effects of network contention, not just ping-pong for 2 nodes
  - handles SMP clusters - on-node and off-node communications
- Provides greater insight for performance analysis
- Useful for more accurate performance modeling

MPI Benchmark Experiments on the Altix

- Measurements were done on SGI Altix 3700 (Aquila) at SAPAC.
  - 160 Intel Itanium 1.3 GHz (APAC AC is 1.6 GHz)
  - 160 GB of RAM (APAC AC is 2 GB per CPU)
  - ccNUMA architecture
  - SGI NUMAlink for communication
  - SGI Linux (ProPac3)
  - SGI MPI library
  - Used MPI_DSM_CPULIST for process binding

SGI Altix Architecture



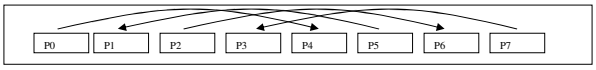
The Architecture of 128 Processor for SGI Altix

Avoid Message Buffering and Single Copy

- It is possible under certain conditions to avoid the need to buffer messages in SGI MPI.
- SGI MPI provides single copy options with much better performance.
- This is the default for MPI_Isend, MPI_Sendrecv, MPI_Alltoall, MPI_Bcast, MPI_Allreduce and MPI_Reduce for messages above a specified size (default is 2 Kbytes).
- However, it is not enable by default for MPI_Send.
- Can force single copy (non-buffered) using
  MPI_BUFFER_MAX <min message size>
- Note that our measurements for MPI_Send use the default settings (buffered).
- Using a single copy send can give significantly better communication times – a factor of 2 or 3 in some cases.

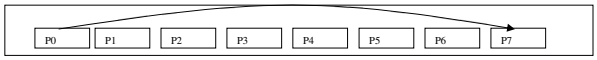
Point-to-Point

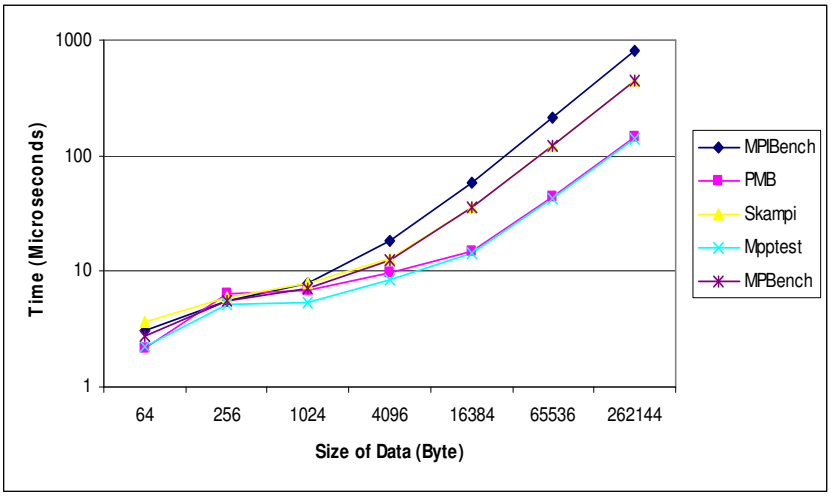
- MPIBench measures not just the time for a ping-pong communication between two processors, but can also measure the effects of contention when all processors take part in point-to-point communication.



- PMB and Mpptest which involve processors 0 and 1 only.

- SKaMPI and MPBench which use the first and last processor processor.





Comparison of results from different MPI benchmarks for Point-to-Point (send/receive) communications using 8 processors.

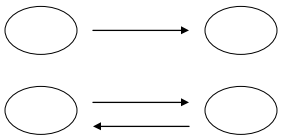
Summary of Latency and Bandwidth Results.

Number of Processors	2	4	8	16	32	64	128
Latency (MPIBench)	1.96 us	1.76 us	2.14 us	2.21 us	2.56 us	2.61 us	2.70 us
Latency (MPBench)	1.76 us	2.07 us	2.48 us	2.41 us	2.53 us	3.06 us	3.01 us
Bandwidth (MPIBench)	851 MB/s	671 MB/s	464 MB/s	462 MB/s	256 MB/s	256 MB/s	248 MB/s
Bandwidth (MPBench)	831 MB/s	925 MB/s	562 MB/s	562 MB/s	549 MB/s	532 MB/s	531 MB/s

Measured latency (for sending a zero byte message) and bandwidth (for a 4 MByte message) for different numbers of processes on the Altix. Results for MPIBench are for all processes communicating concurrently, so include contention effects. Results for MPBench (in blue font) are for only two communicating processes (processes 0 and N-1) with no network or memory contention.

		Alphaserver SC	SGI Altix 3700
Latency	Internode	5 Microseconds	1.96 Microseconds
	Intranode	5 Microseconds	1.76 Microseconds
Bandwidth	Internode	262 MBytes/sec	464 MB/s (buffered) 754MB/s (single copy)
	Intranode	740 MBytes/sec	851 MB/s (buffered) 1421 MB/s (single copy)

MPI_Sendrecv



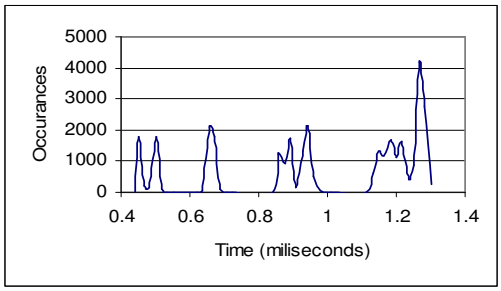
- SGI Altix with SGI MPI library
  - The time is similar with MPI_Send
  - So the bidirectional bandwidth works
- Alphaserver SC with Compaq MPI library
  - The time is 2 times slower.
  - So bidirectional bandwidth does NOT work

MPI_Bcast

- Broadcast usually implemented using tree structure of point-point communications, taking O(log N) time for N processes.
- Broadcast for **SGI Altix** take advantage of the Single Copy.
- The Quadrics network on the AlphaServer SC provides a very fast hardware broadcast.
- But only if the program is running on a contiguous set of processors.
- Otherwise, a standard software broadcast algorithm is used.

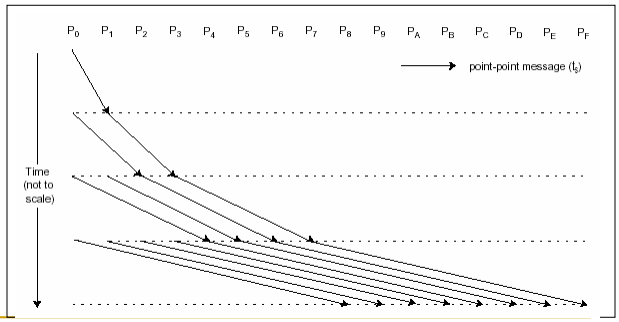
MPI_Bcast Results

- Comparison of broadcast performance with Alphaserver SC is difficult
  - Depends on whether hardware broadcast is used
- For smaller numbers of processors (< 32) the Altix does better due to its higher bandwidth.
- For larger numbers of processors the AlphaServer SC does better due to excellent scalability of hardware broadcast of the Quadrics network.
- Hardware broadcast of a 64 KByte message on the AlphaServer SC
  - 0.40 ms for 16 CPUs
  - 0.45 ms on 128 CPUs
- On the SGI Altix
  - 0.22 ms on 16 CPUs
  - 0.45 ms on 64 CPUs
  - 0.62 ms for 128 CPUs.
- If the processors for an MPI job on the AlphaServer SC are not contiguous, it uses software broadcast, which is much slower than Altix.

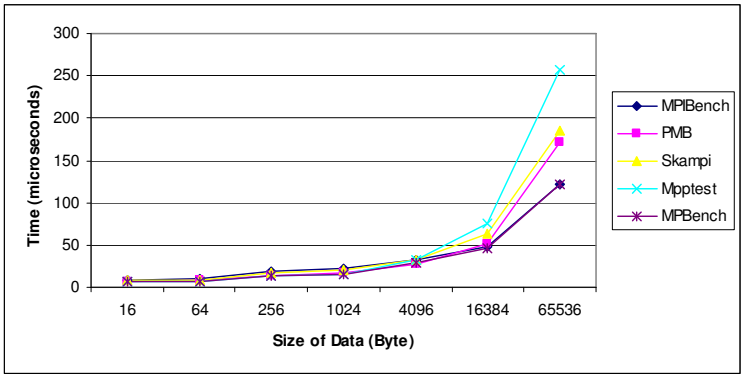


Distribution result for MPI_Bcast at 256KBytes on 32 processors.

The binomial tree that constructs a 16 process software-based MPI Bcast from a collection of point-to-point messages.



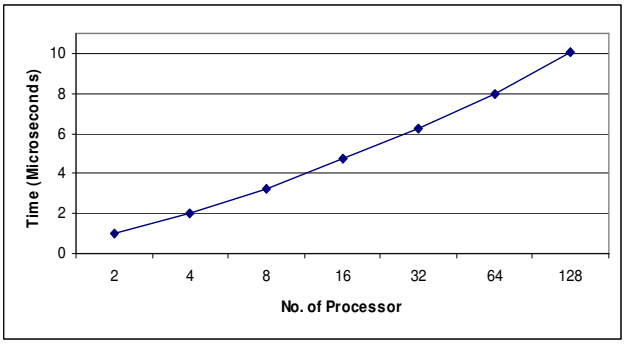
Benchmark Comparison for MPI_Bcast



Comparison between MPI benchmarks for MPI_Bcast on 8 processors.

- The main difference is :-
  - SKaMPI, Mpptest and PMB assume data is not held in cache memory.
  - MPIBench did preliminary “warm-up” repetitions to ensure data is in cache before measurements are taken.
    - The newer version already added an option to clear the cache.
- Another difference is:-
  - Measure collective communications time at the root node.
  - However for broadcast, the root node is the first to finish, and this may lead to biased results.
  - To solve the problem - Insert a barrier operation before each repetition.
  - But, Mpptest and PMB adopt a different approach – they assign a different root processor for each repetition.
  - On a distributed memory cluster this has little affect on the results.
  - However for shared memory moving the root to a different processor has a significant overhead.

MPI_Barrier

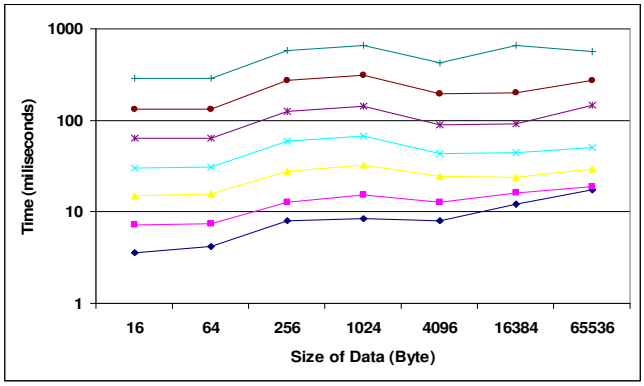


Average time for an MPI barrier operation for 2 to 128 processors

- **The times scale logarithmically with numbers of processors.**
- **Hardware broadcast on the Quadrics means that a barrier operation on the Alphaserver SC is very fast and takes almost constant time of around 5-8 microseconds for 2 – 128 processors.**
- **Which is similar to the Altix.**

MPI_Scatter

- Overall the time grows remarkably slowly with the data size.
- 1KByte per process - Altix is around 4 to 6 times faster than APAC SC.
- 4Kbytes per process - Altix is around 10 times faster than APAC SC.

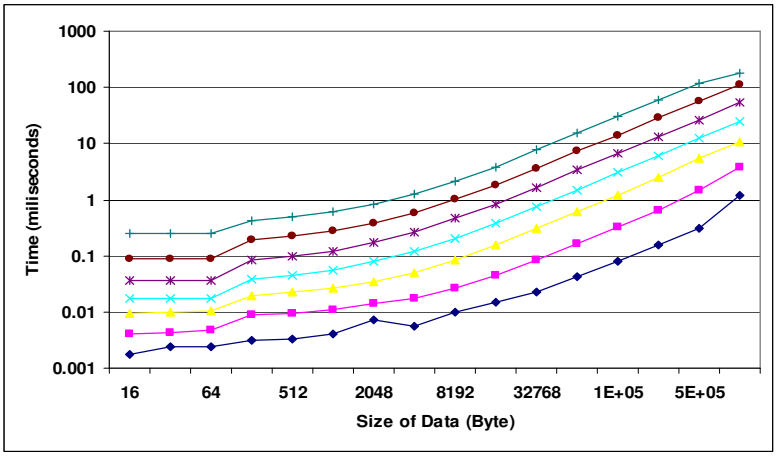


Performance for MPI_Scatter for 2 to 128 processors

MPI_Gather

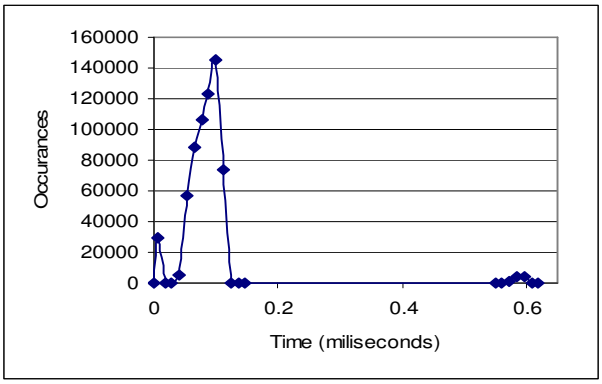
- For MPI_Gather, the time is proportional to the data size.
- The Altix gives significantly better results than APAC SC.
- 1 KByte per process – Altix is around 2 to 4 times faster than APAC SC.
- 2 KByte per process – Altix is around 10 times faster than APAC SC.
- > 2 Kbytes per process
  - MPI on Alphaserver SC became unstable and crashed,
  - Altix continues to give good performance.

- Figures shows average times to complete for MPI_Gather operation.
- Times is roughly proportional to data size for large data sizes.



Performance for MPI_Gather for 2 to 128 processors

- Process 0 is by far the slowest process to complete,
  - It has to gather and merge results from all other processors.
- Process 1 is the first to complete,
  - It is on the same node as the root process.



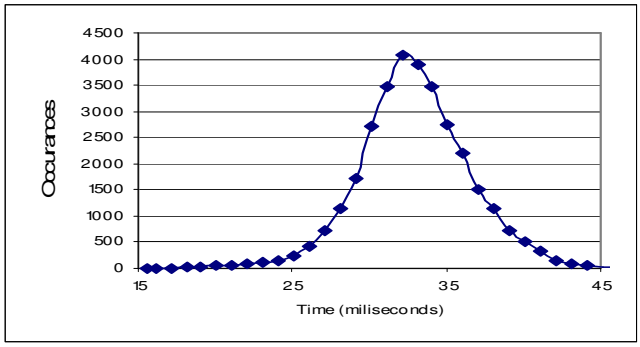
Probability distribution for 64 processors and at 4KBytes per process.

MPI_Alltoall

- The times for MPI_Alltoall are significantly better on the Altix than the Alphaserver SC.
- 1 KByte per processor - Altix is 2 to 4 times faster than APAC SC.
- 4 KByte per processor - Altix is 20 times faster than APAC SC.
- 8 KByte per processor - Altix is 30 times faster than APAC SC.
- This is partly because the MPI implementation on the APAC SC did not appear to be optimized for SMP nodes.

MPI_Alltoall Time Distribution

- Distribution figures shows that for large messages, there is wide range of completion times.
- This is indicate the contention effects.



Distribution for MPI_Alltoall for 32 processor at 256KBytes

Summary

- The SGI Altix shows very good MPI communications performance.
  - Using process binding improves performance
  - Enabling single copy for MPI_Send greatly improves performance
- Overall performance was significantly better than the APAC SC.
- Altix provides higher bandwidth and lower latency than the Quadrics network on the APAC SC.
- Altix has significantly better collective communications performance
  - except where SC can use Quadrics hardware for broadcast and barrier
- Different MPI benchmarks can give very different results for some operations
  - much greater variation on the Altix than for clusters.
  - mainly due to cache effects, which are important on ccNUMA

END