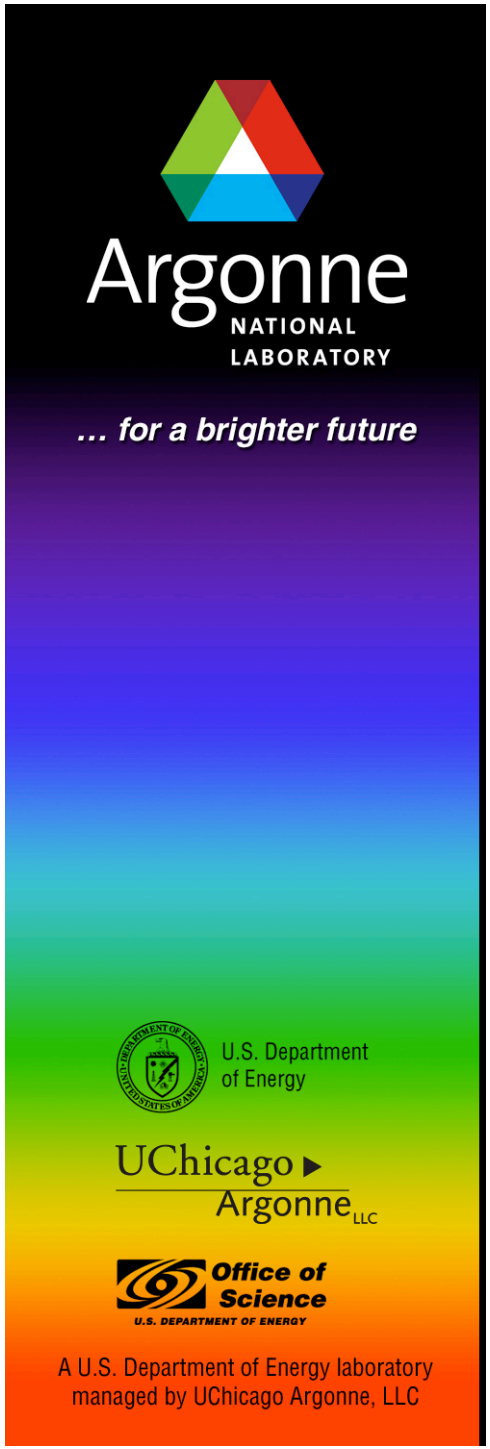


Test Suite for Evaluating Performance of MPI Implementations That Support MPI_THREAD_MULTIPLE

Rajeev Thakur and William Gropp

Mathematics and Computer Science Division

Argonne National Laboratory

Argonne, Illinois, USA

Introduction

- Thread-safe MPI implementations are becoming increasingly common
- Thread safety does not come for free, however
- Implementation must protect certain data structures or parts of code with mutexes or critical sections
- Implementations often focus on correctness first and performance later (if at all)
- Users need a way to determine how efficiently an implementation can support multiple threads
- Hence, a performance test suite is needed

Overview of MPI and Threads

- MPI-2 defines four levels of thread safety
  - **`MPI_THREAD_SINGLE`**: only one thread
  - **`MPI_THREAD_FUNNELED`**: only one thread that makes MPI calls
  - **`MPI_THREAD_SERIALIZED`**: only one thread at a time makes MPI calls
  - **`MPI_THREAD_MULTIPLE`**: any thread can make MPI calls at any time
- User calls **`MPI_Init_thread`** to indicate the level of thread support required; implementation returns the level supported
- Our test suite focuses on the **`MPI_THREAD_MULTIPLE`** case

Performance Expectations

- Users often have the following performance expectations
  - The cost of thread safety, compared with say MPI_THREAD_FUNNELED, is low
  - Multiple threads making MPI calls, such as MPI_Send or MPI_Bcast, can make progress simultaneously
  - A blocking MPI routine in one thread does not consume excessive CPU resources while waiting
- How true are they in practice?

Categories of Tests

- Cost of thread safety
  - One simple test to measure overhead of MPI_THREAD_MULTIPLE
- Concurrent progress
  - Tests to measure concurrent bandwidth by multiple threads of a process to multiple threads of another process, compared with multiple processes to processes
- Computation overlap
  - Tests to measure overlap of communication with computation
  - Tests to measure ability of an application to use a thread to provide a nonblocking version of a communication operation

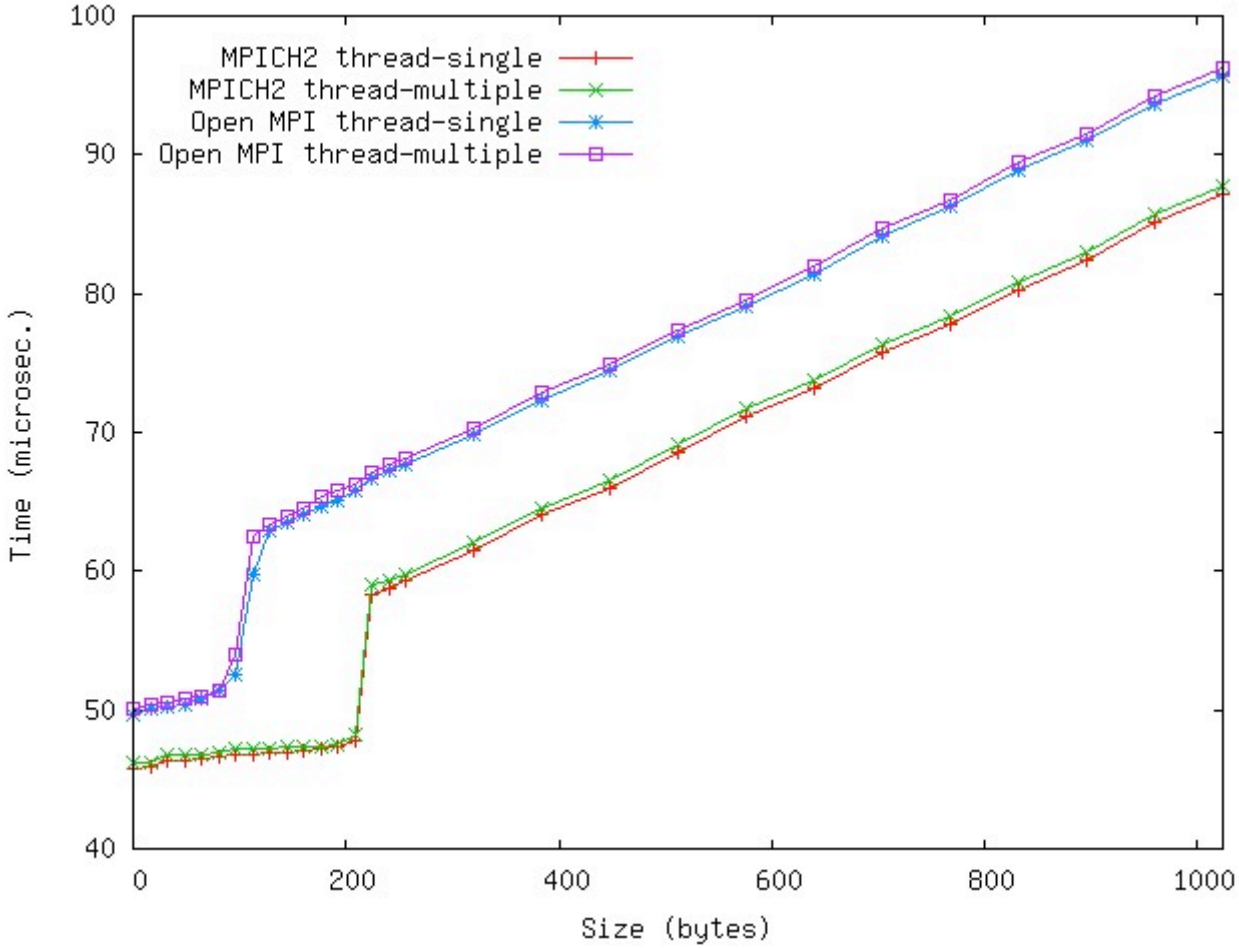
Platforms

- Linux Cluster
  - "Breadboard" cluster at Argonne with GigE
  - Each node has two dual-core 2.8 GHz AMD Opterons
  - MPICH2 1.0.5, Open MPI 1.2.1
- Sun Fire SMP
  - From the Sun cluster at Univ. of Aachen
  - Sun Fire E2900 with 8 dual-core UltraSPARC IV 1.2 GHz CPUs
  - Sun's MPI (ClusterTools 5)
- IBM SMP
  - IBM p655+ SMP from the DataStar cluster at SDSC
  - Eight 1.7 GHz POWER4+ CPUs
  - IBM's MPI

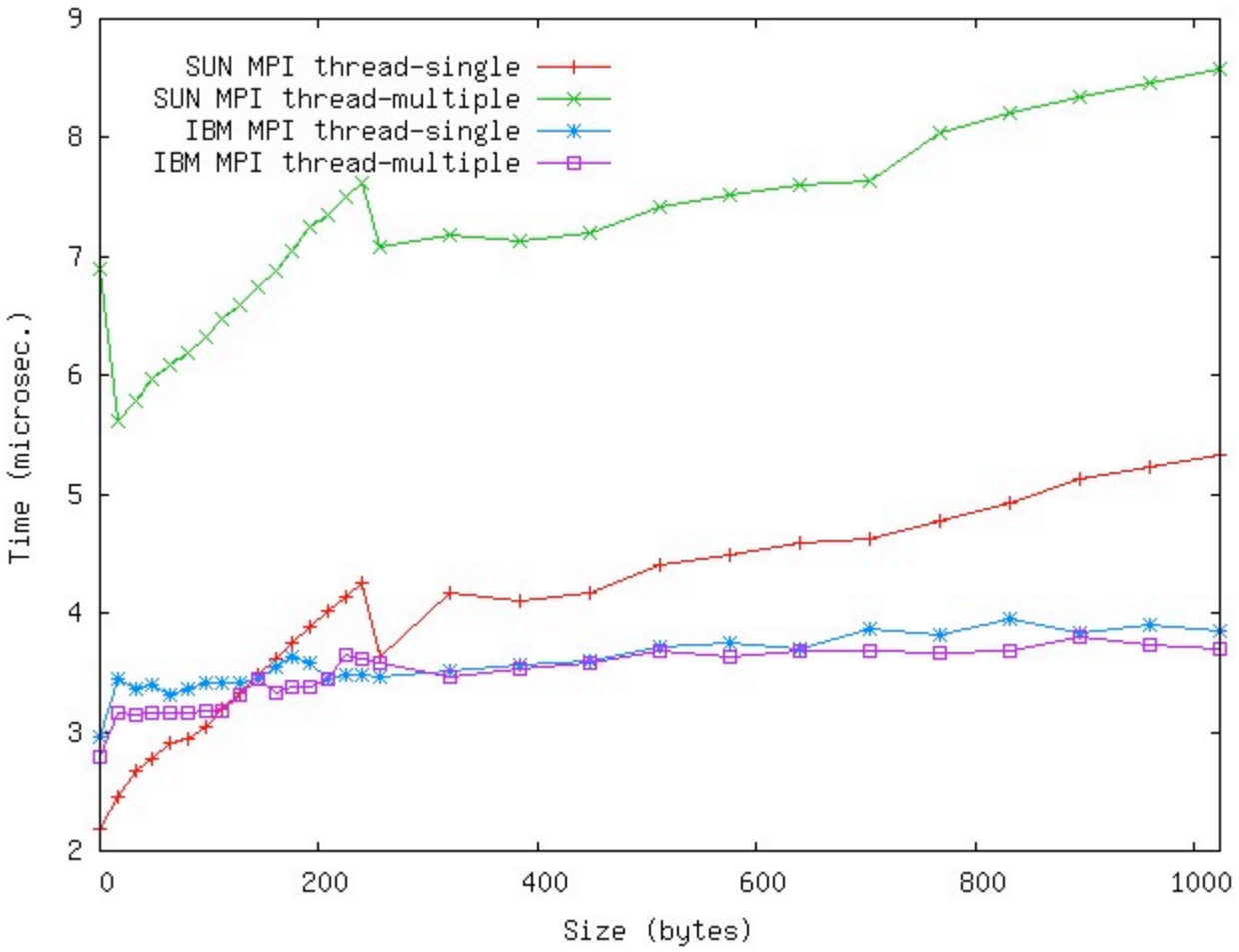
Test 1: MPI_THREAD_MULTIPLE Overhead

- Measures ping-pong latency for two cases of a *single*-threaded program
  - Initializing MPI with just MPI_Init
  - Initializing MPI with MPI_Init_thread for MPI_THREAD_MULTIPLE
- Demonstrates overhead of acquiring and releasing locks even when not needed

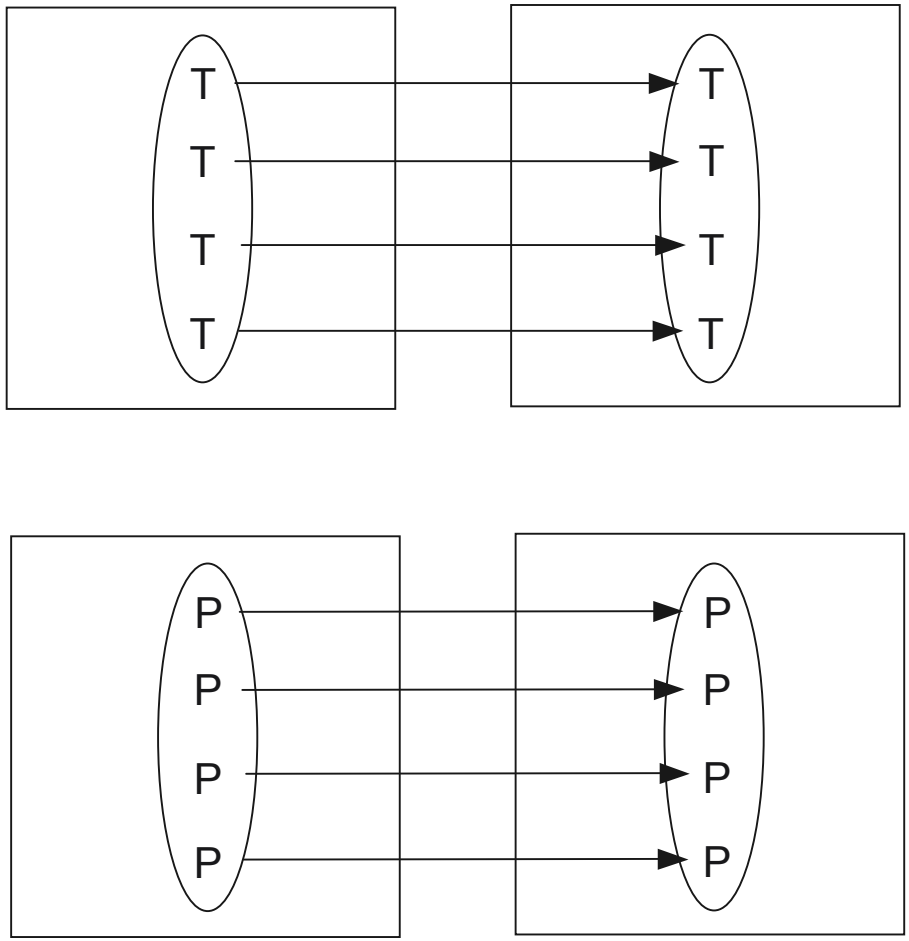
MPI_THREAD_MULTIPLE Overhead on Linux Cluster



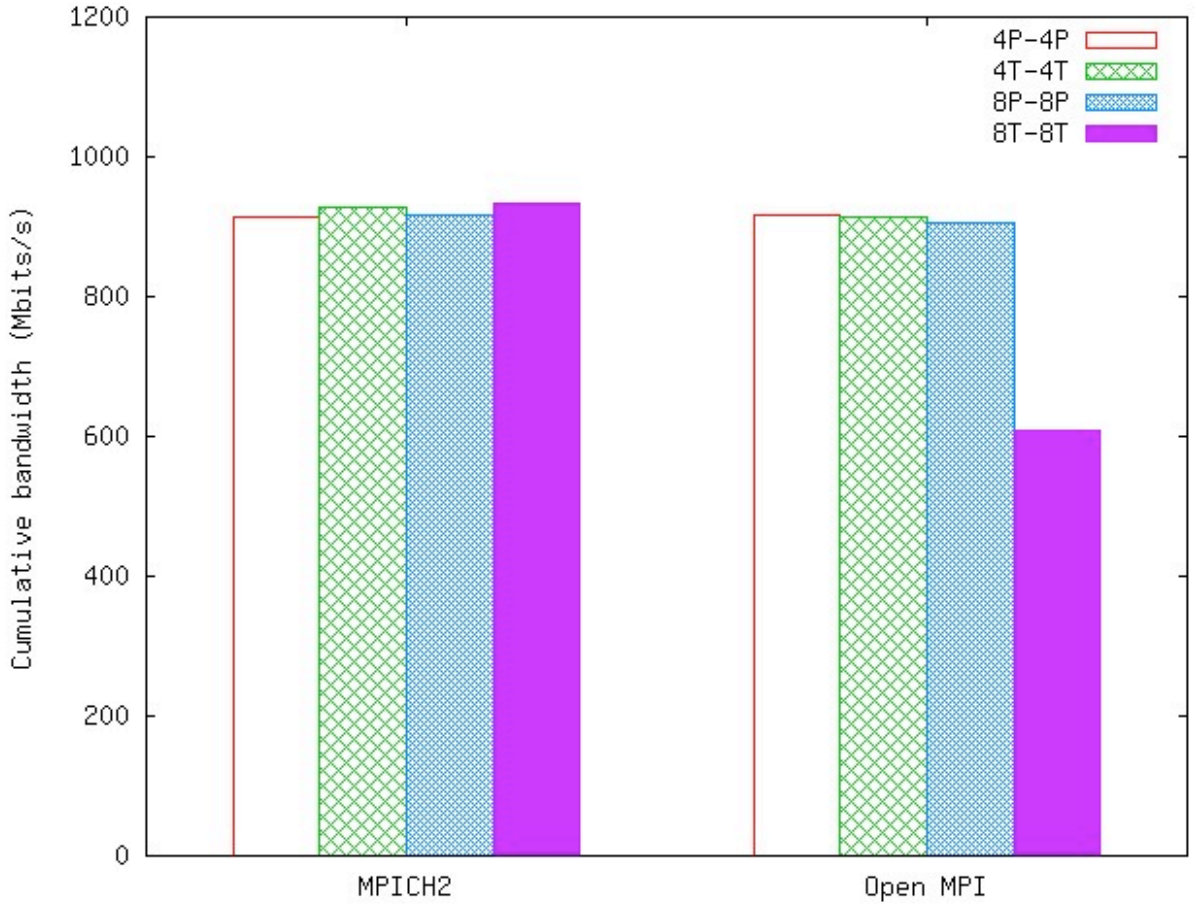
MPI_THREAD_MULTIPLE Overhead on Sun & IBM SMPs



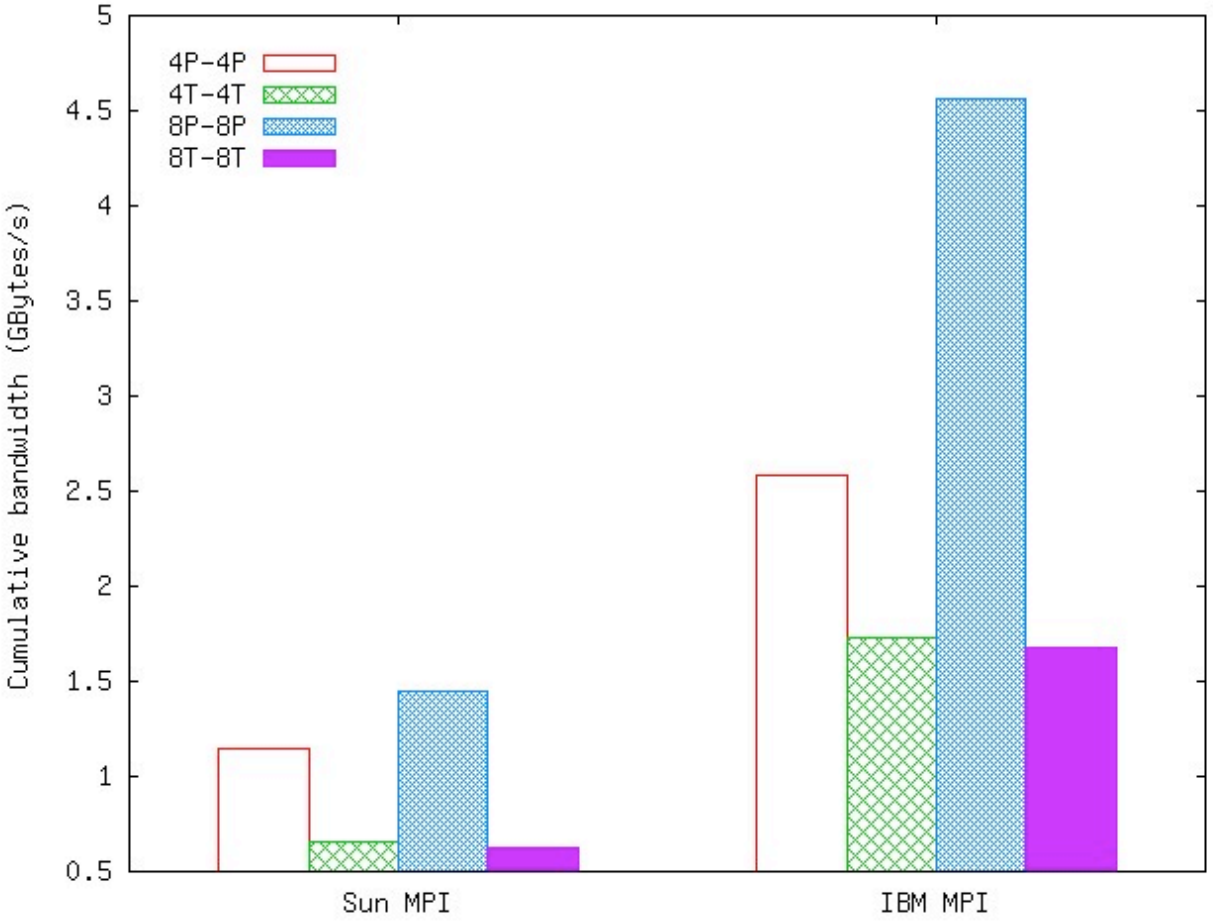
Tests with Multiple Threads versus Processes



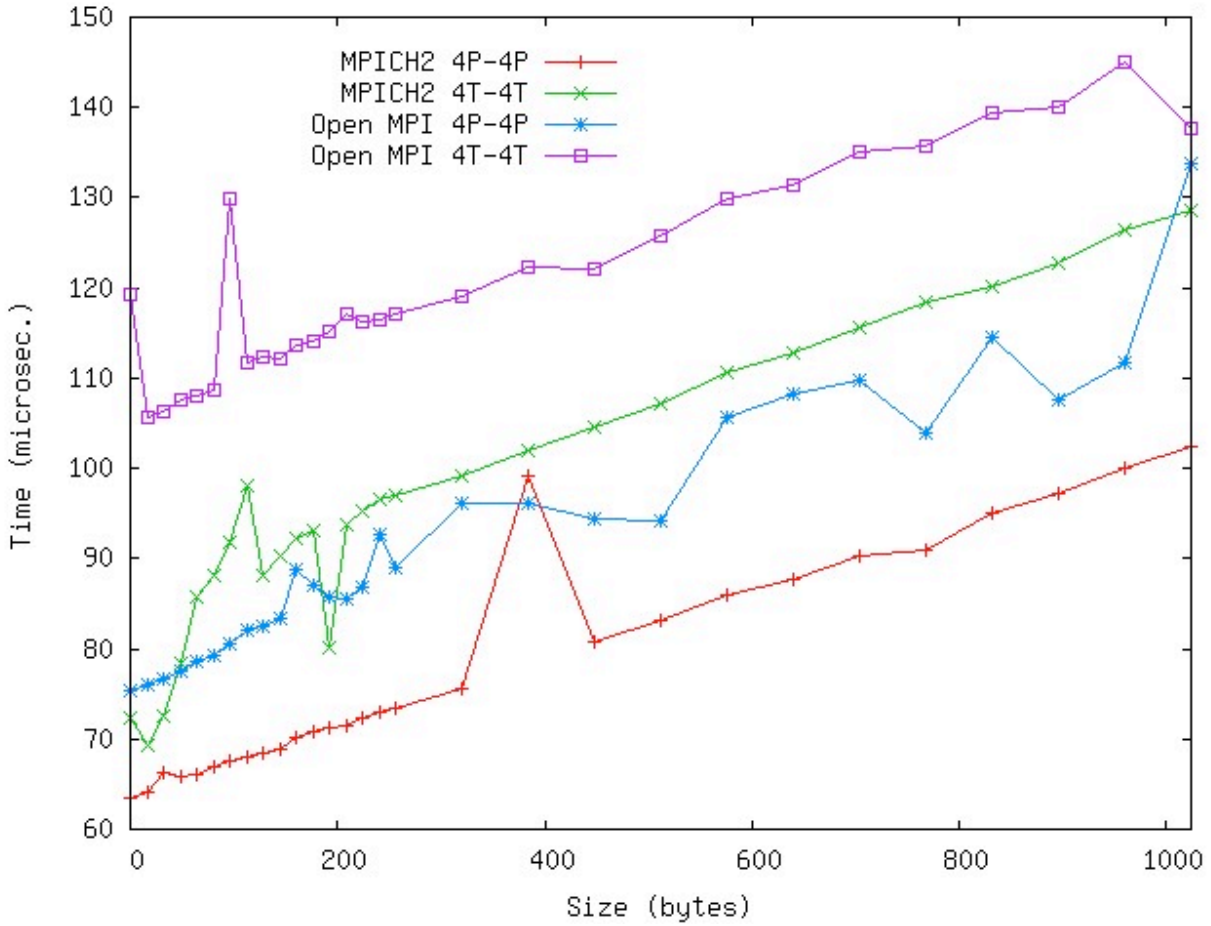
Concurrent Bandwidth Test on Linux Cluster



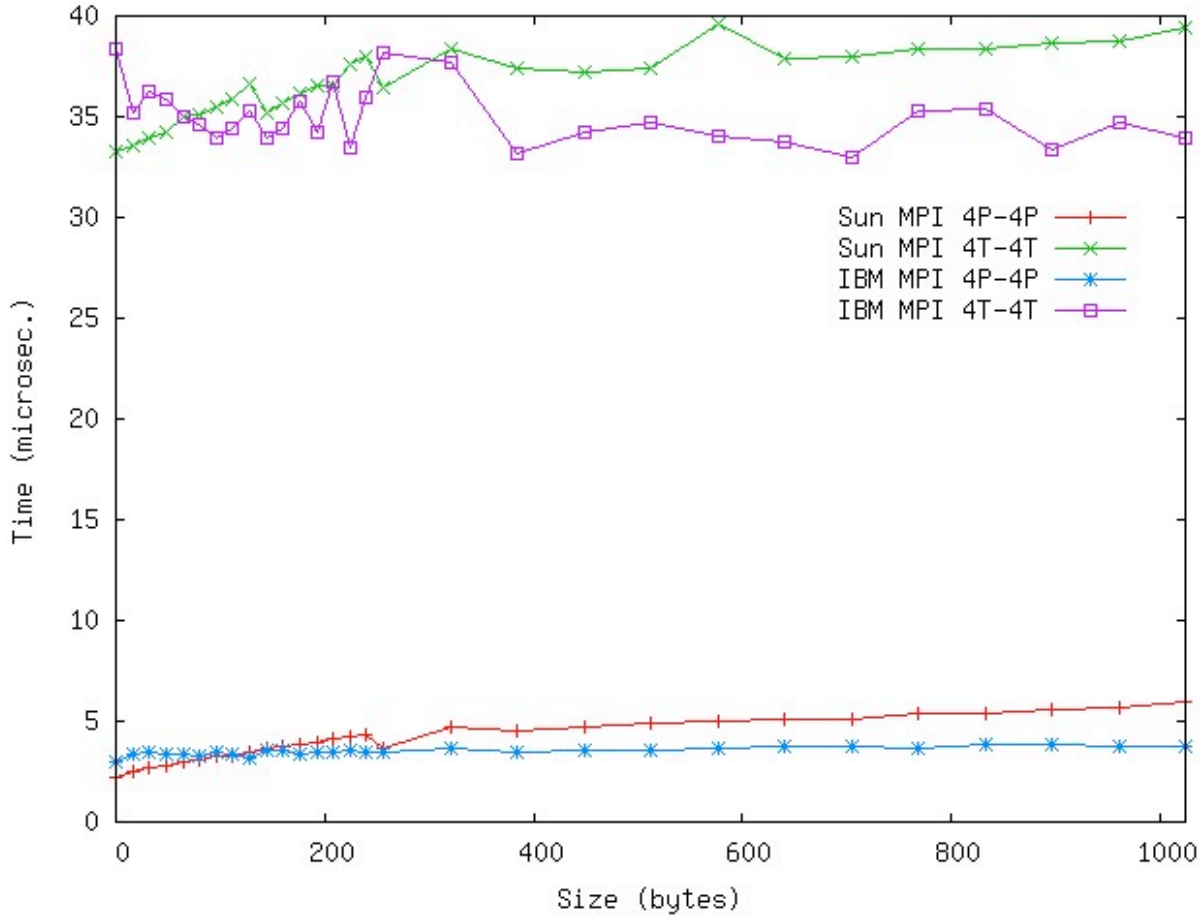
Concurrent Bandwidth Test on Sun and IBM SMPs



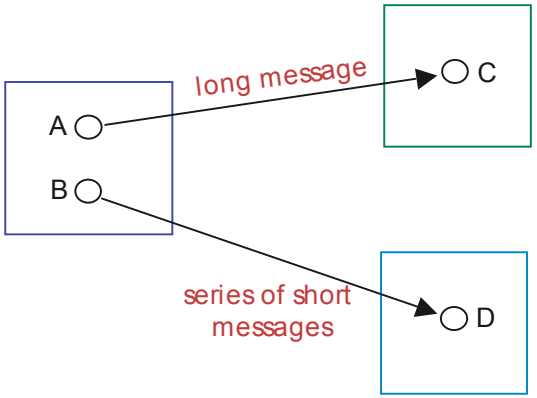
Concurrent Latency Test on Linux Cluster



Concurrent Latency Test on Sun and IBM SMPs

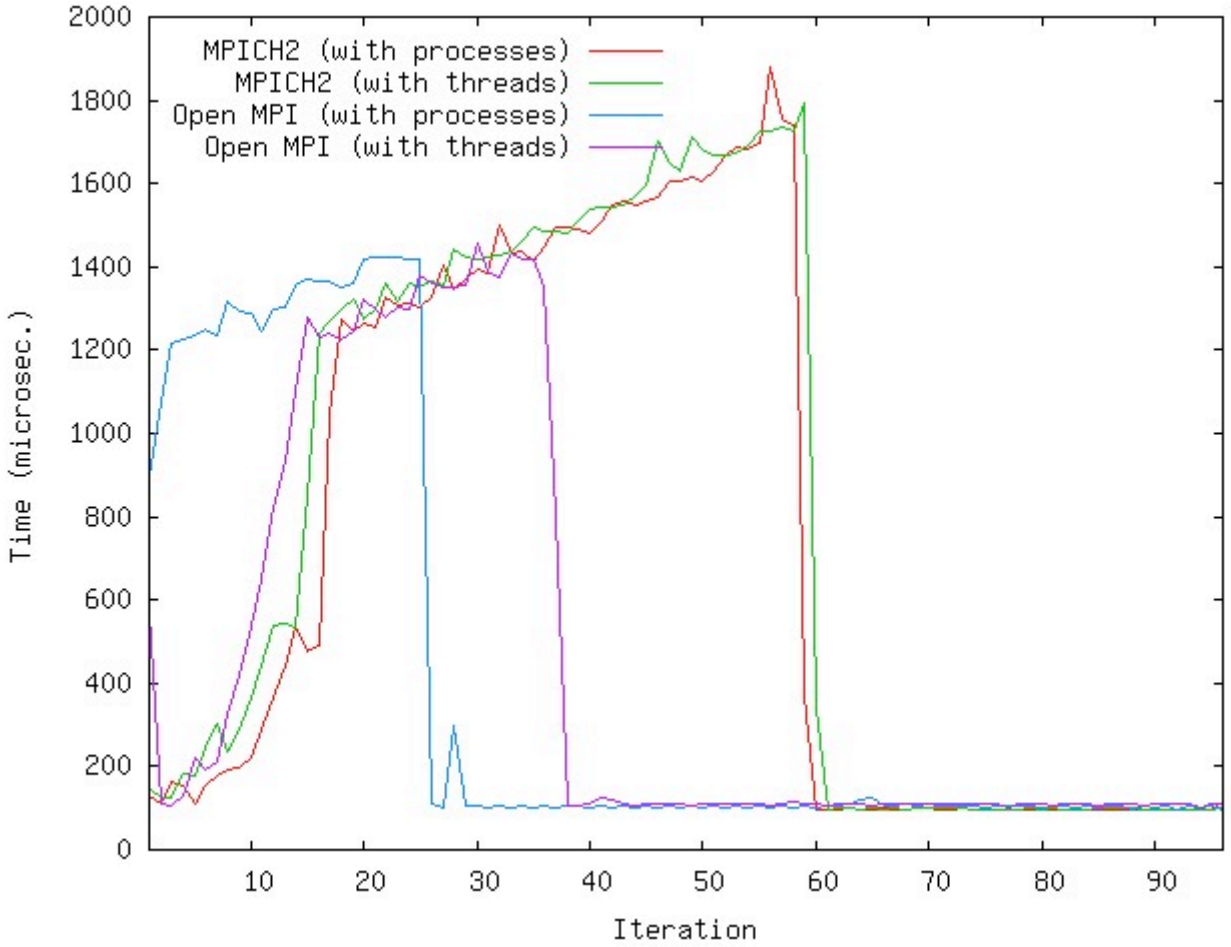


Test 4: Concurrent Short-Long Messages

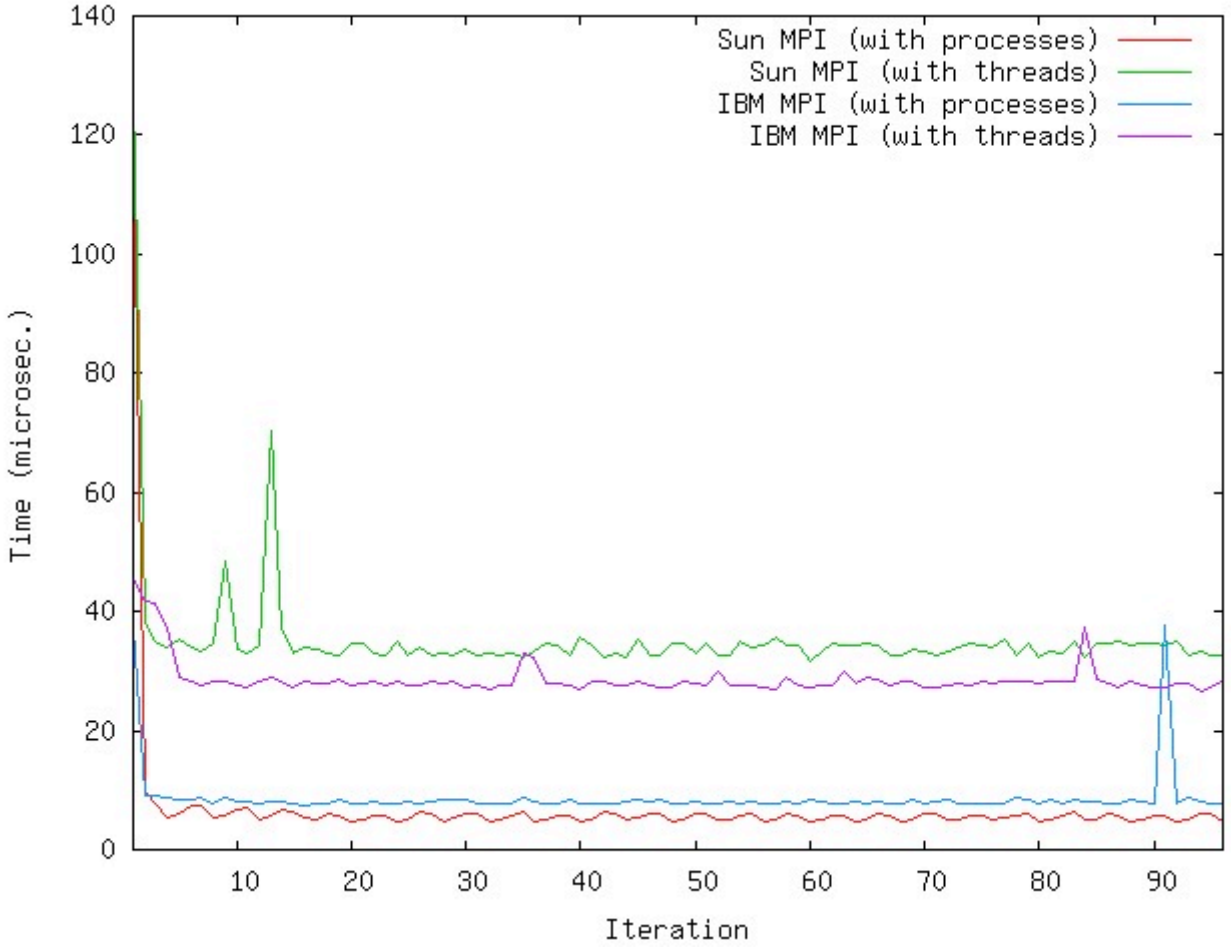


- "A" sends a long message to C
- "B" simultaneously sends a series of short messages to D
- Measure the variation in time taken by the short messages when
  - A and B are threads of one process
  - A and B are separate processes

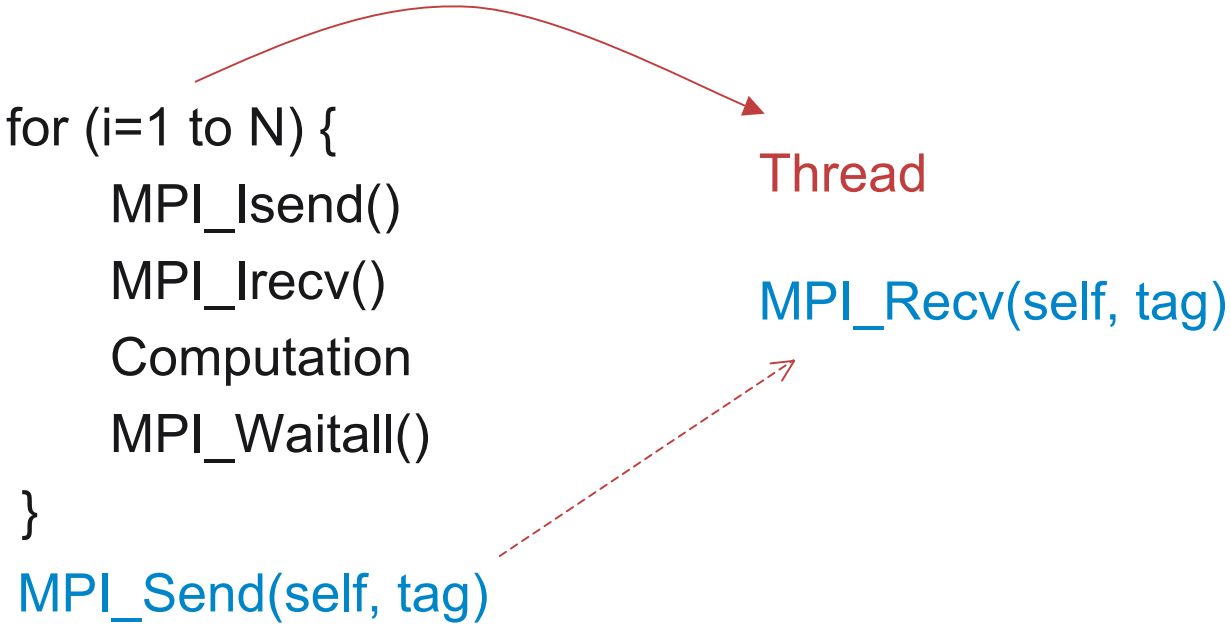
Concurrent Short-Long Messages Test on Linux Cluster



Concurrent Short-Long Messages Test on Sun & IBM SMPs

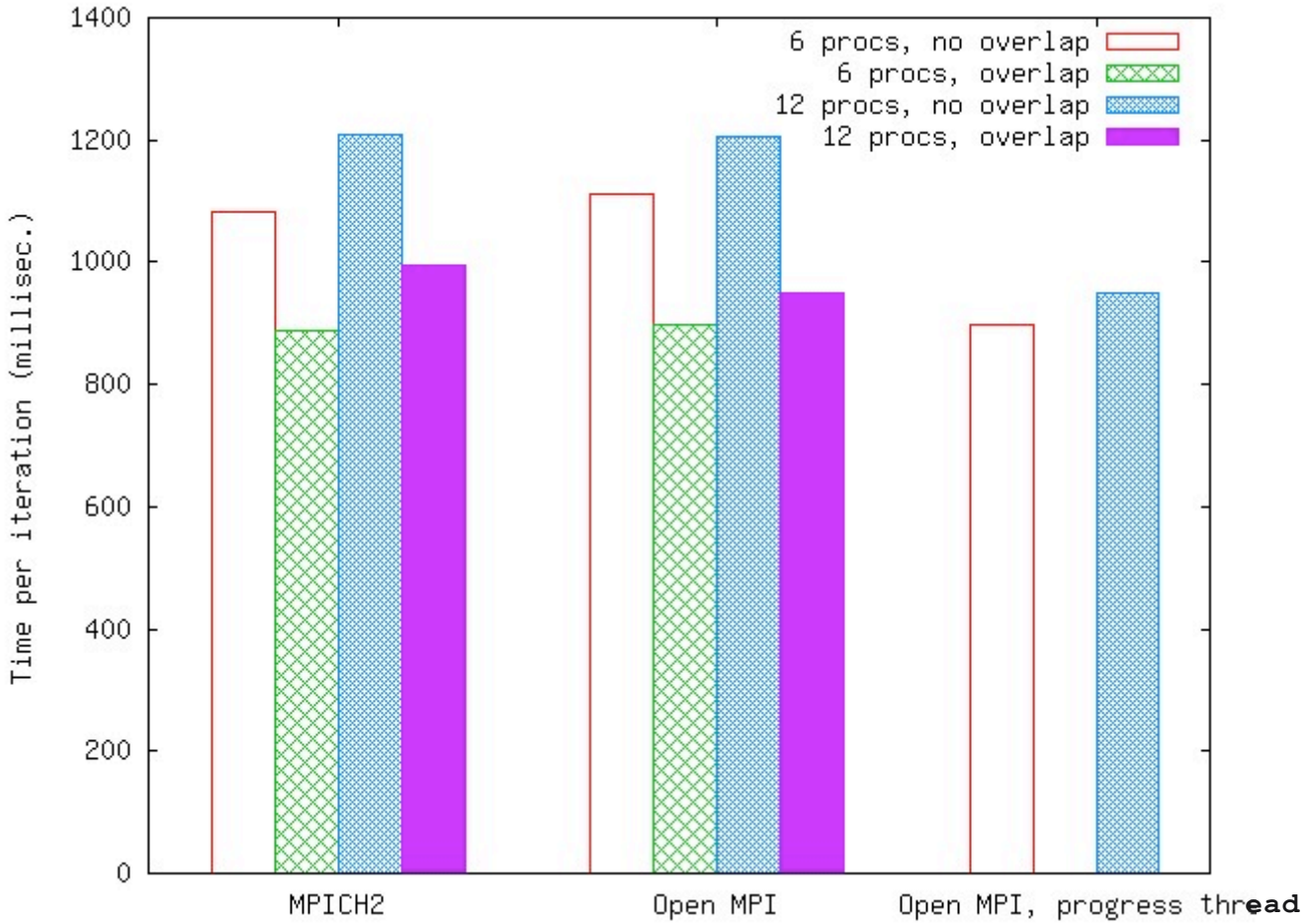


Test 5: Computation/Communication Overlap

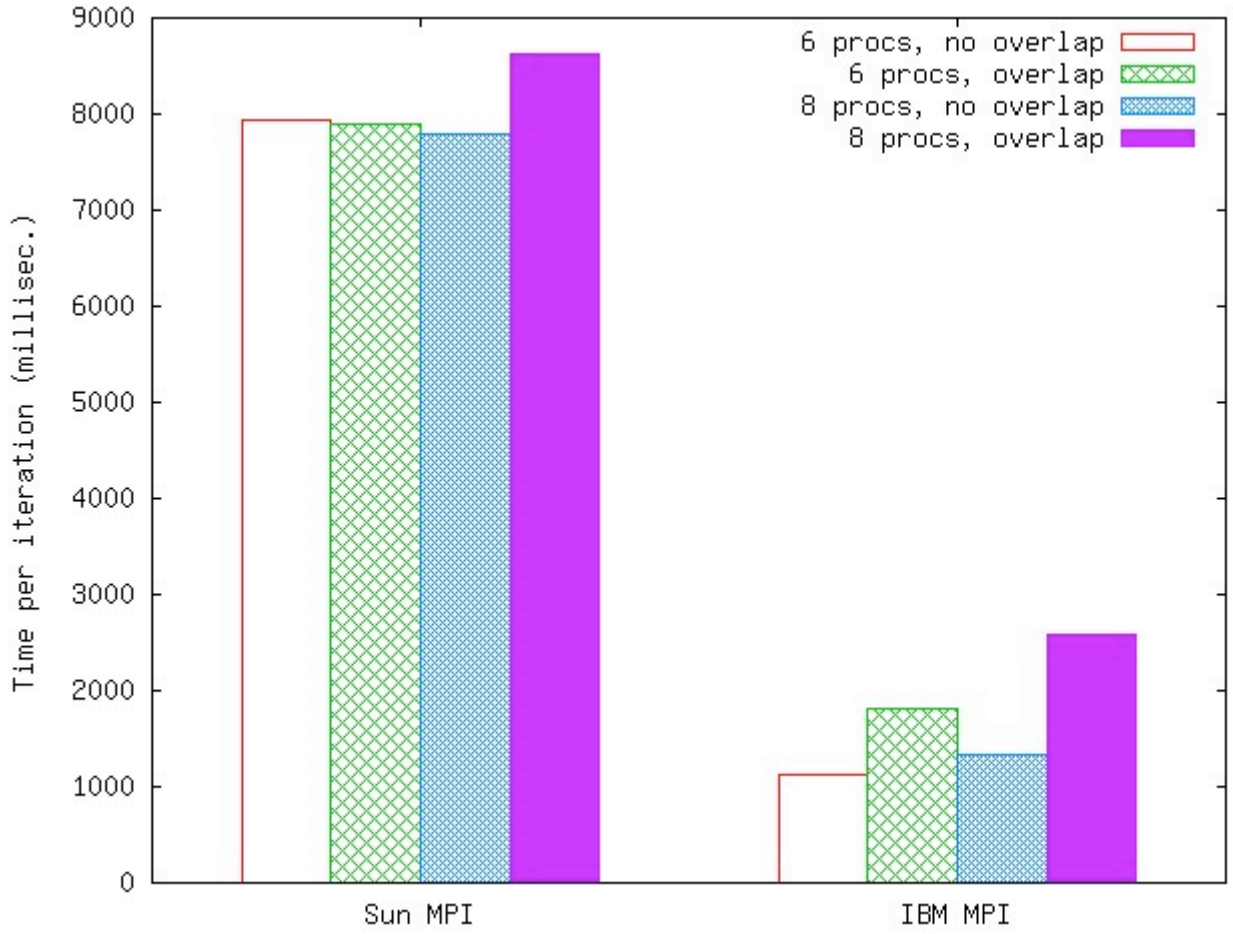


- Measure time taken by the communication-computation loop with and without the thread

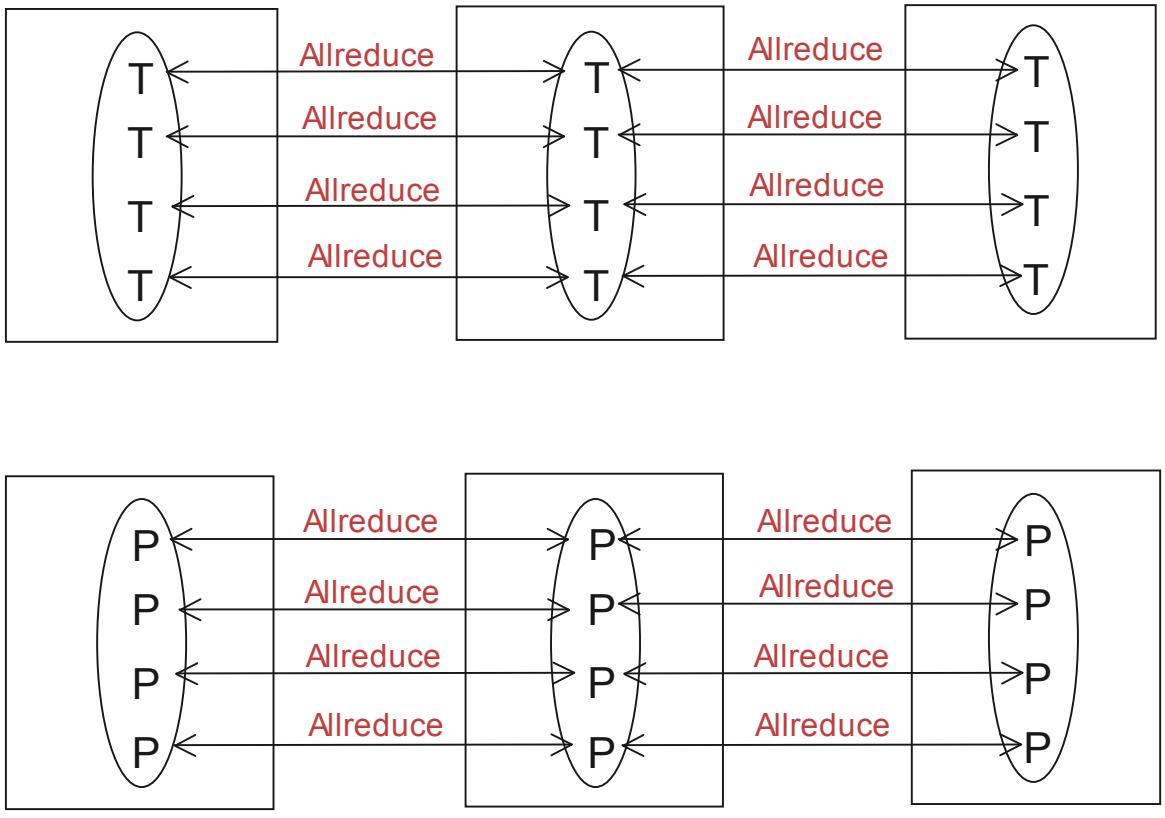
Comp/Comm Overlap Test on Linux Cluster



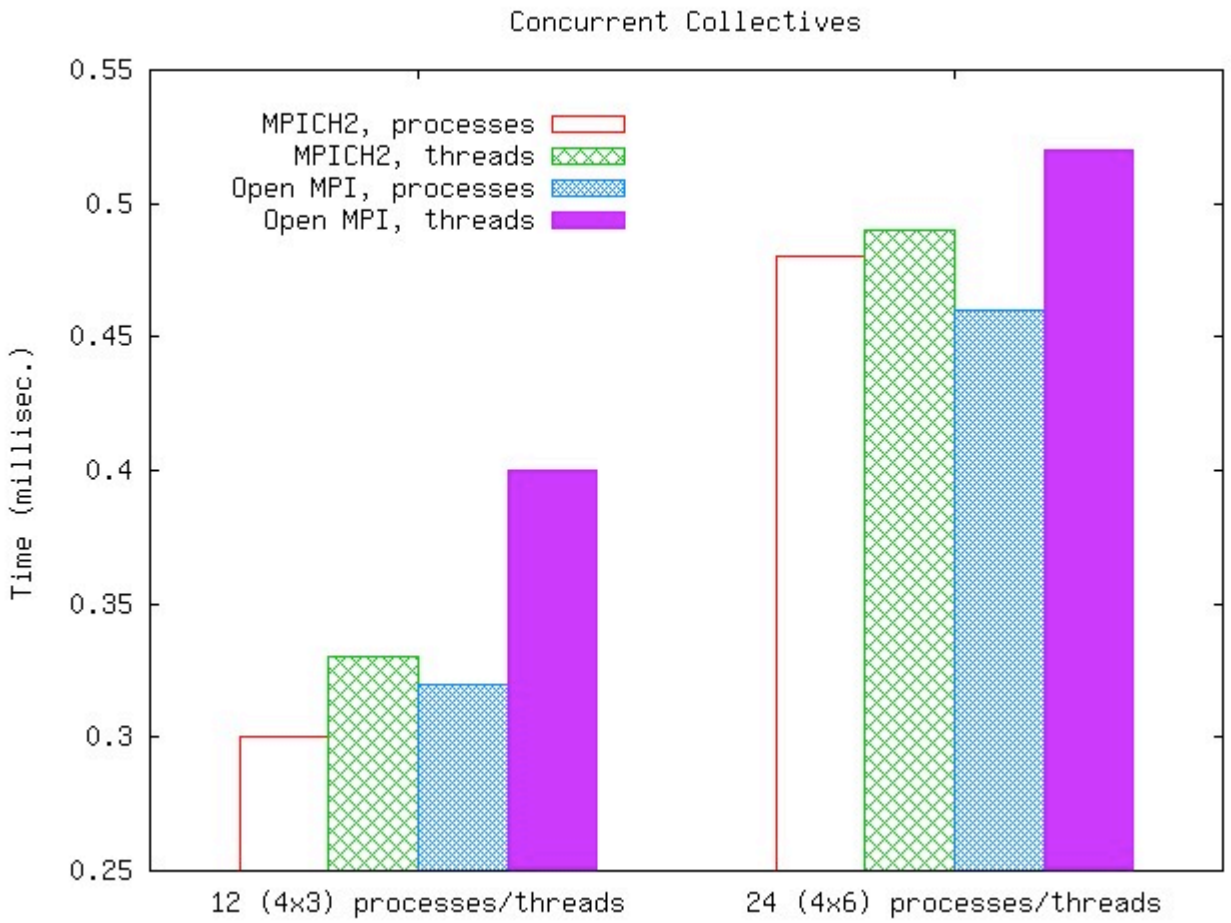
Comp/Comm Overlap Test on Sun & IBM SMPs



Test 6: Concurrent Collectives



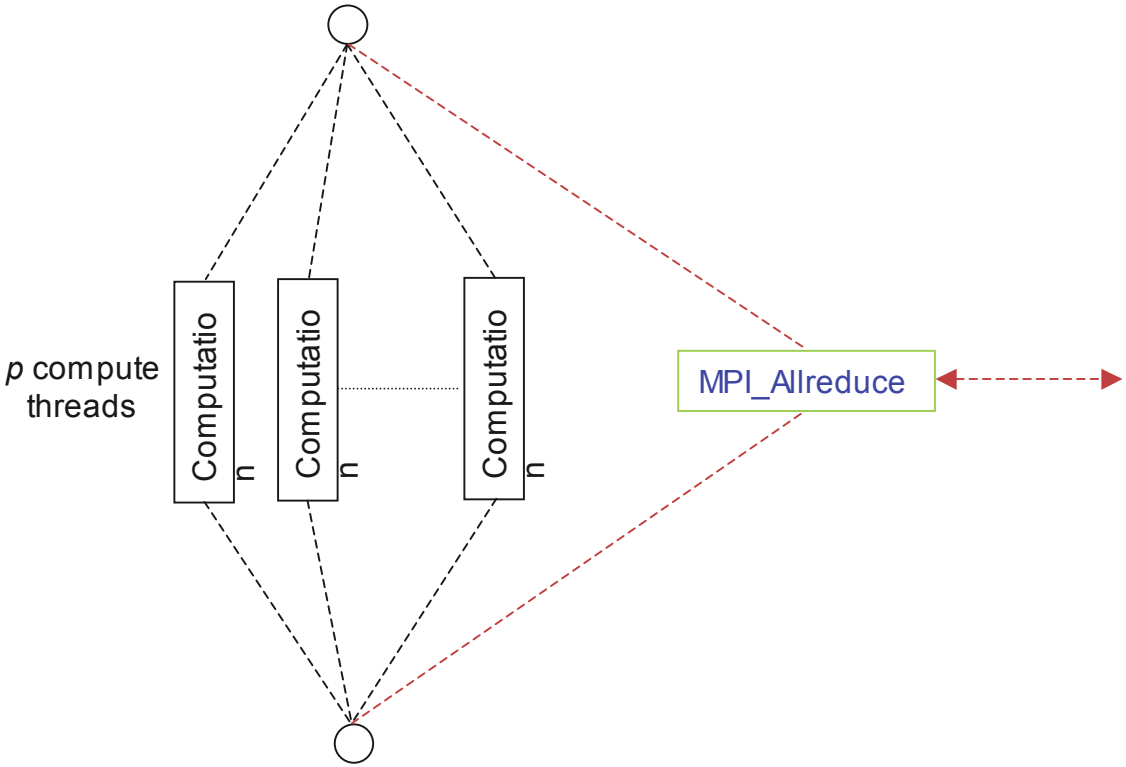
Concurrent Collectives Test on Linux Cluster



Test 7: Concurrent Collectives and Computation

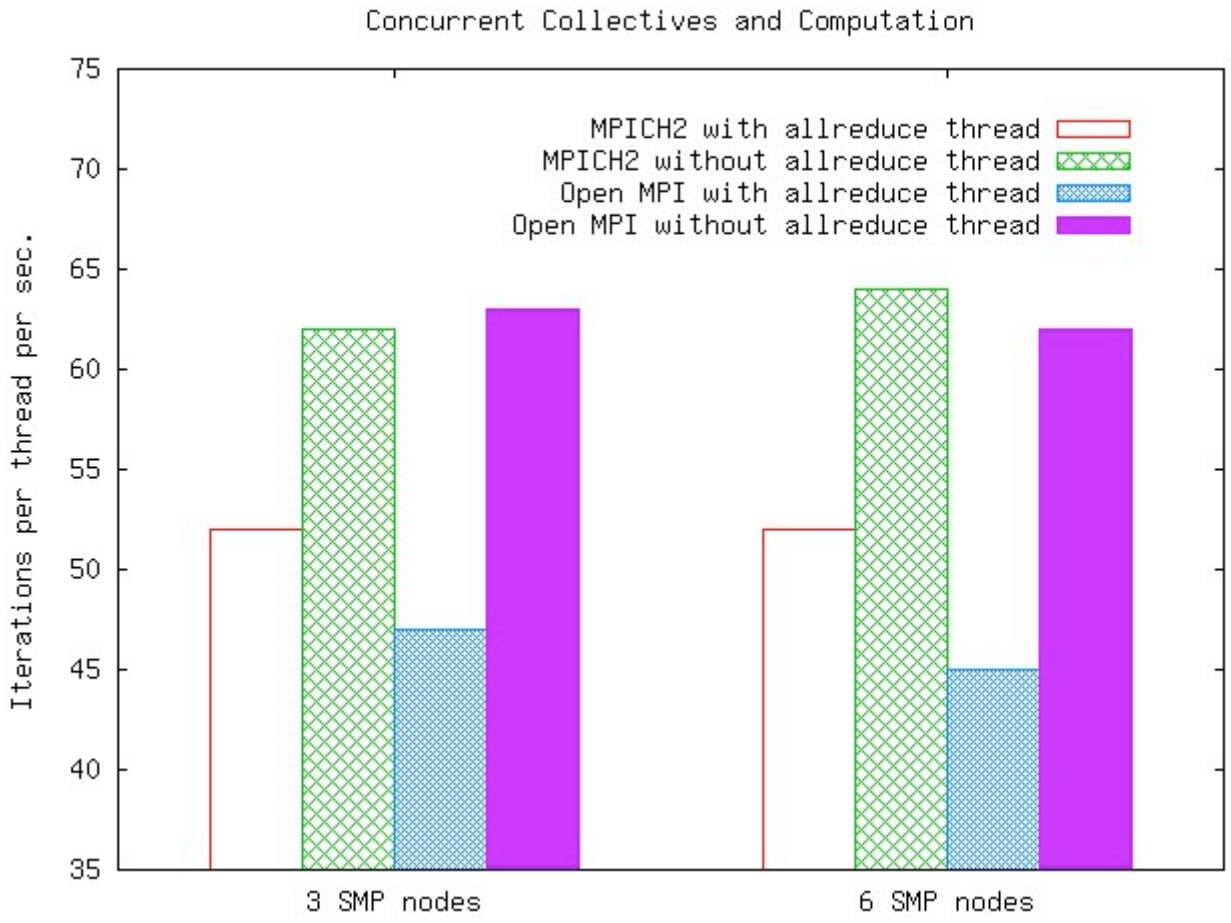
- Uses p+1 threads on a node with p processors
- Threads 0 to p-1 perform some computation iteratively
- Thread p does an MPI_Allreduce with corresponding thread on other nodes
- After the Allreduce completes, thread p sets a flag
- This flag stops computation in other threads
- The average number of compute iterations completed on the threads is reported
- This number is compared with the case where there is no allreduce thread

Test 7: Concurrent Collectives and Computation



Concurrent Collectives and Computation Test on Linux Cluster

(the higher the better)



Concluding Remarks

- There is a need for tests that shed light on the performance of MPI implementations in the presence of multiple threads
- The results indicate relatively good performance with MPICH2 and Open MPI on Linux clusters, but poor performance with IBM and Sun MPI on IBM and Sun SMPs
- We plan to add more tests, such as to measure overlap of comp/comm with MPI-2 file I/O and connect-accept features
- We welcome contributions from others to the test suite
- Available for download from http://www.mcs.anl.gov/~thakur/thread-tests