Non-data-communication Overheads in MPI: Analysis on Blue Gene/P

(P. Balaji, A. Chan, W. Gropp, R. Thakur, E. Lusk 2008)

Yi Wei

Department of Computer & Information Sciences
University of Delaware

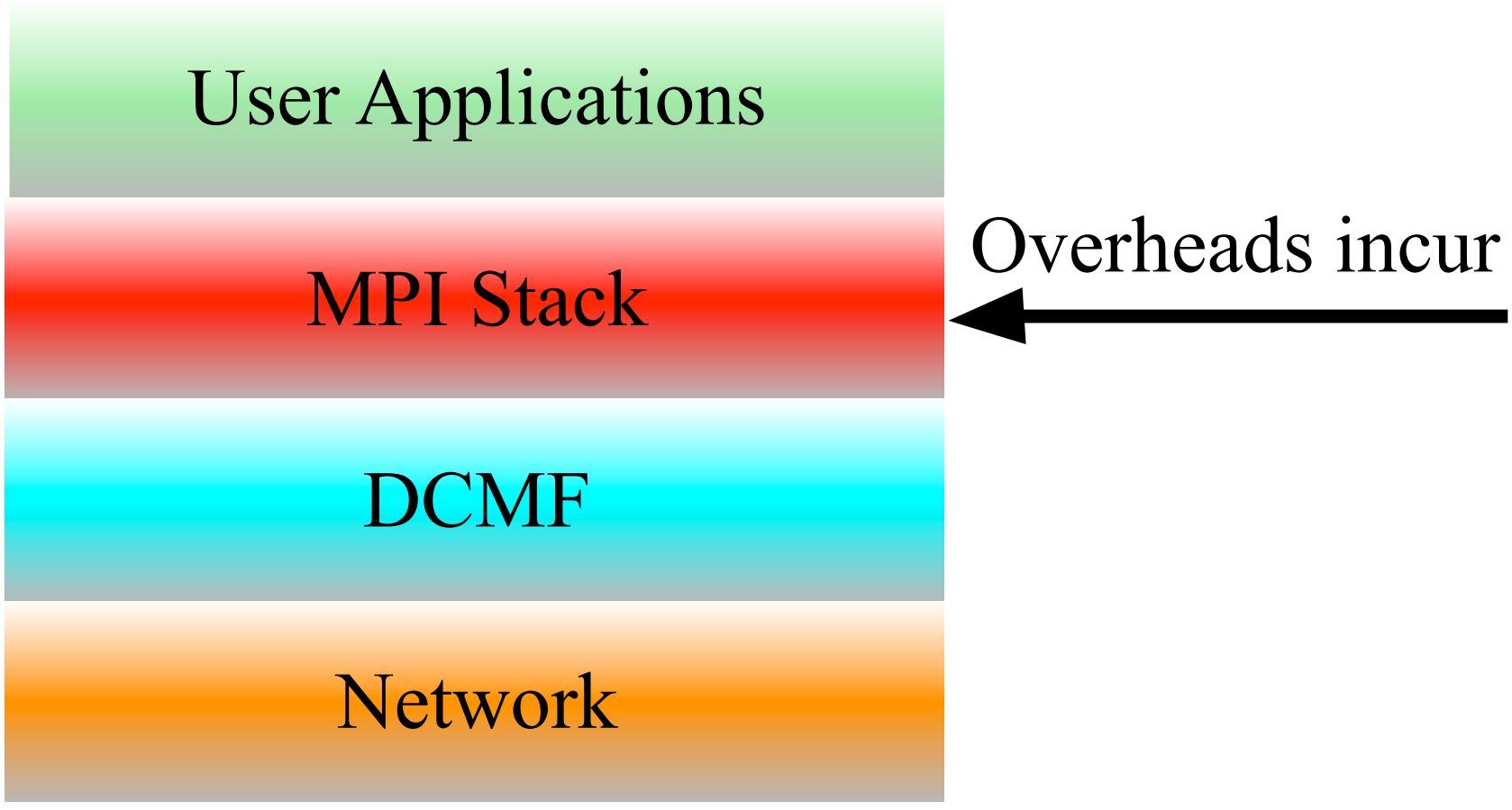
March 17, 2009

Outline

Motivations

- Challenges to the modern HPCs
  1. Lower processing power on each processor.
  2. Massive number of processors.
  3. Highly interleaving between computation and communication.
- Why we interested in MPI overheads ?
  Overheads on smaller systems may become significant on large scale systems.
- The IBM Blue Gene/P
  1. Five different networks. Three of them for MPI communications.
  2. The MPI implementation is based on MPICH2.
  3. Use Deep Computing Messaging Framework (DCMF) and Component Collective Messaging Interface (CCMI) as general underlying libraries.

Hardware and Software Layers on Blue Gene/P



What does Blue Gene/P look like?

Images from Google search

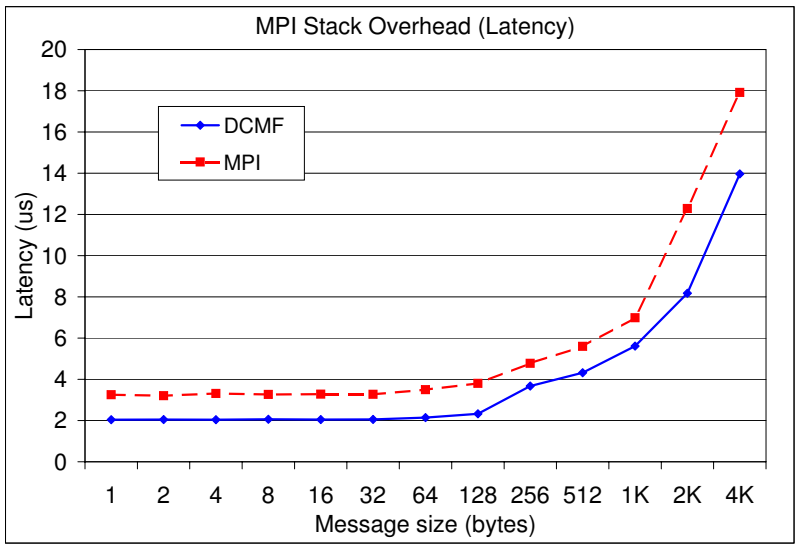
List of Test

- Basic overhead test
- Request processing test
- Tag and source matching test
- Multi-request operation test
- Derived datatype processing test
- Buffer alignment test
- Unexpected message processing test
- Thread communication test

Basic Overheads

- The first experiment compares the MPI communication performance with the underlying communication system performance. (DCMF)
- Blocking communication operations are used for latency tests and nonblocking communication operations for bandwidth tests.

Experiment Result for Basic MPI Overheads



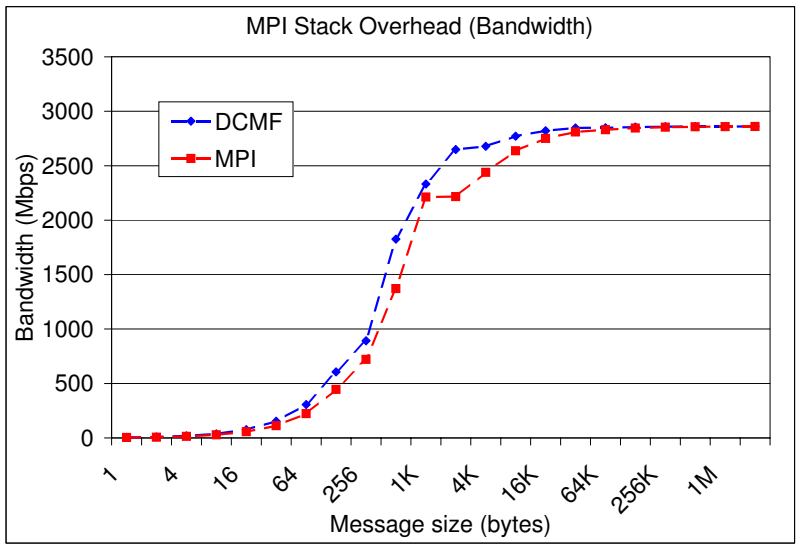


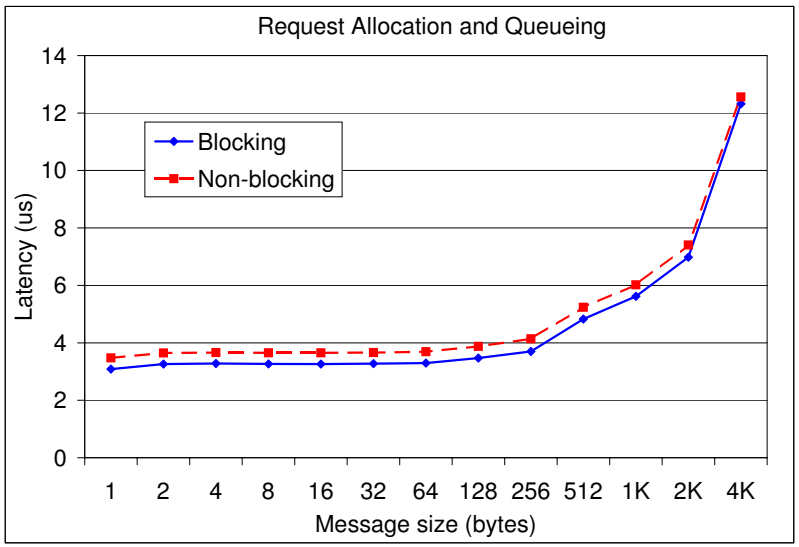
Image cited from paper

MPI on Blue Gene/P adds about $1.1\mu s$ overhead for small packages ($\leq 1KB$) and about $4\mu s$ for larger packages due to the switch of protocols from eager to rendezvous.

Request Allocation and Queueing Overheads

- Non-blocking communication routines provided by MPI have request objects associated to each message send/receive.
- These requests needs to be allocated, initialized, queue/dequeued, thus adding overheads.

Experiment Result for Request Overheads



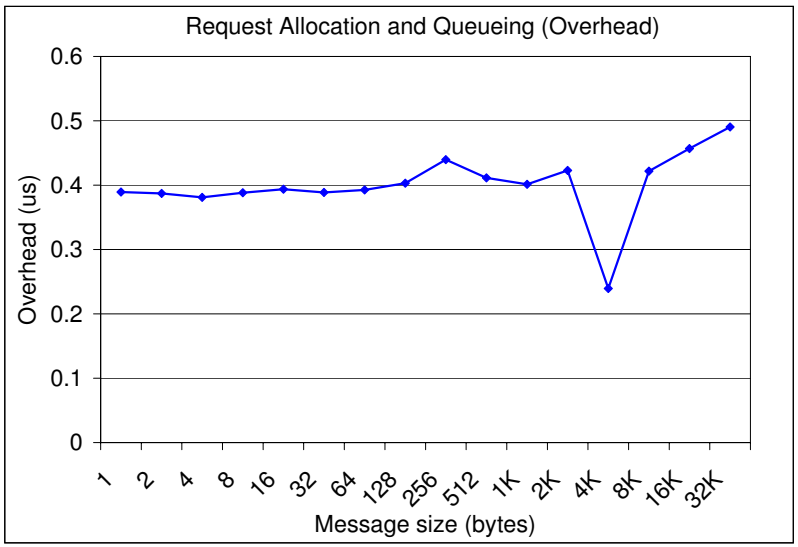


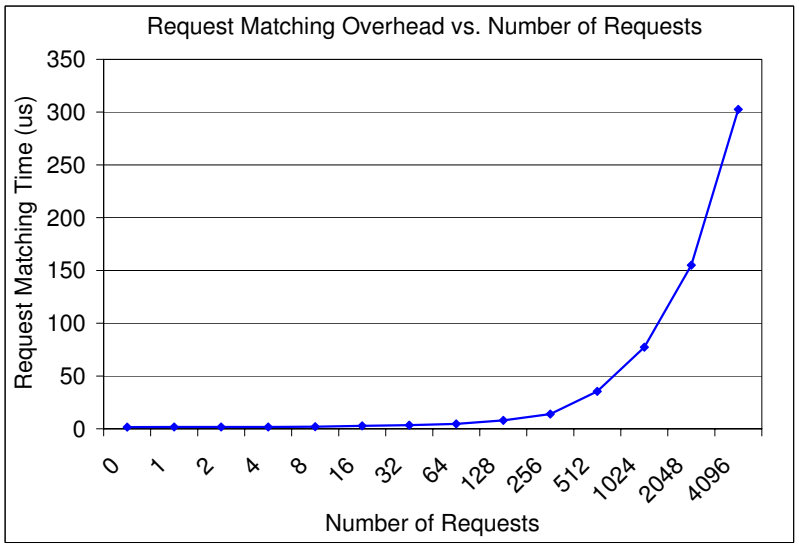
Image cited from paper

This experiment measures the overhead by comparing two versions of MPI programs, one uses MPI_Send and MPI_Recv and the other uses MPI_Isend and MPI_Irecv. The overhead is roughly 0.4μs.

Tag and Source Matching Overheads

- Each MPI sent message carries a tag, and each received message carries a tag and information about the source. So the receiver needs to search the queue of posted receive requests to find the one that matches the arrived message.
- Most current MPI implementations use a single queue for receive requests. This has a potential scalability problem when the length of queue becomes large.

Experiment result for Tag/Source Matching Overheads



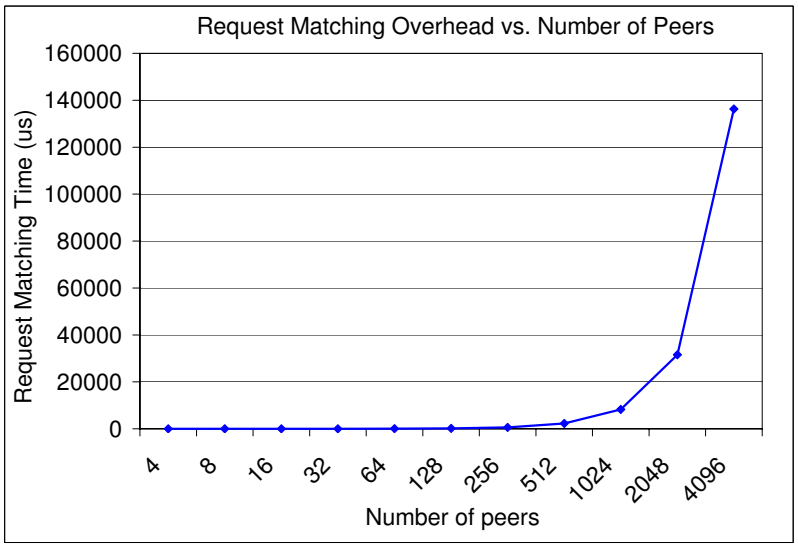


Image cited from paper

For 4096 peers, even just one request per peer can result in the queue parsing time of about 140000μs!

Experiment result for Tag/Source Matching Overheads, cont.

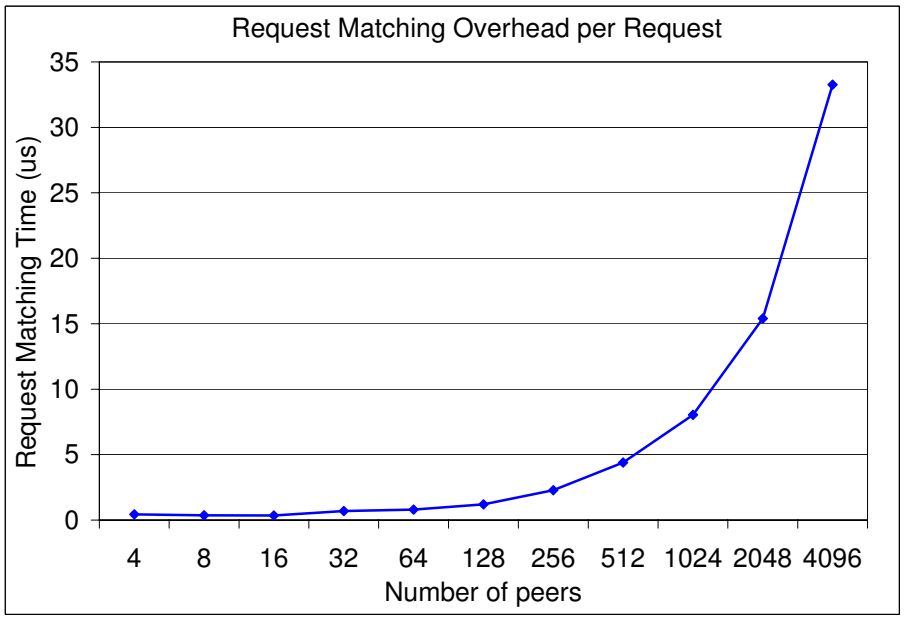


Image cited from paper

Another interesting observation is that the time increase with the number of peers is not linear. The average time per request increases as the number of requests increases.

Algorithmic Complexity of Multi-request Operations

- MPI provides operations such as MPI_Waitany, MPI_Waitsome and MPI_Waitall that allow the user to provide multiple requests at once and wait for the completion of one or more of them.
- The time taken by MPI_Waitany increases linearly with the number of request passed to it. This is because the MPI implementation performs the operation in 2 steps.

Experiment result for Multi-request Operations Overheads

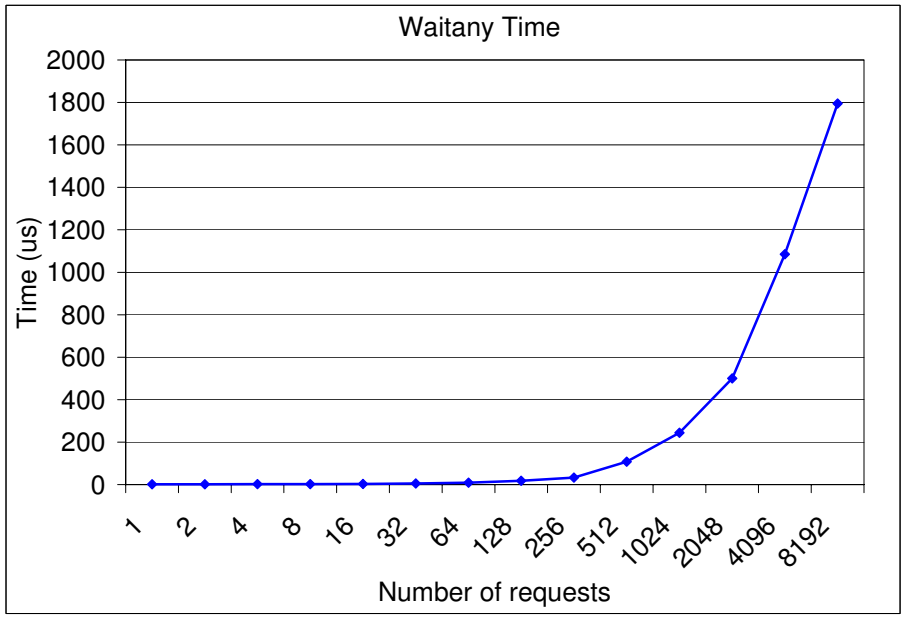


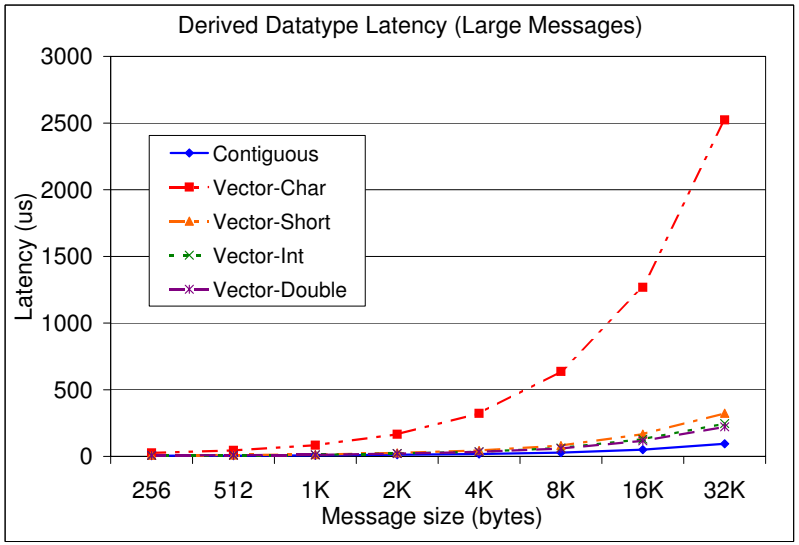
Image cited from paper

Even in the best case, acquiring of internal request handlers can increase the time taken linearly with the number of request.

Derived Datatype Processing Overheads

- MPI allows non-contiguous messages to be sent and received using derived datatypes to describe the message. Implementing these efficiently can be challenging.
- Most MPI implementations pack sparse datatypes into contiguous temporary buffers before performing the actual communication. This stresses both the processing power and the memory/cache bandwidth of the system.
- The experiment reveals a significant gap in performance between sending a contiguous message and a non-contiguous message. The situation is particularly serious for a vector of individual bytes. (MPI_Char)

Experiment result for Derived Data Type Overheads



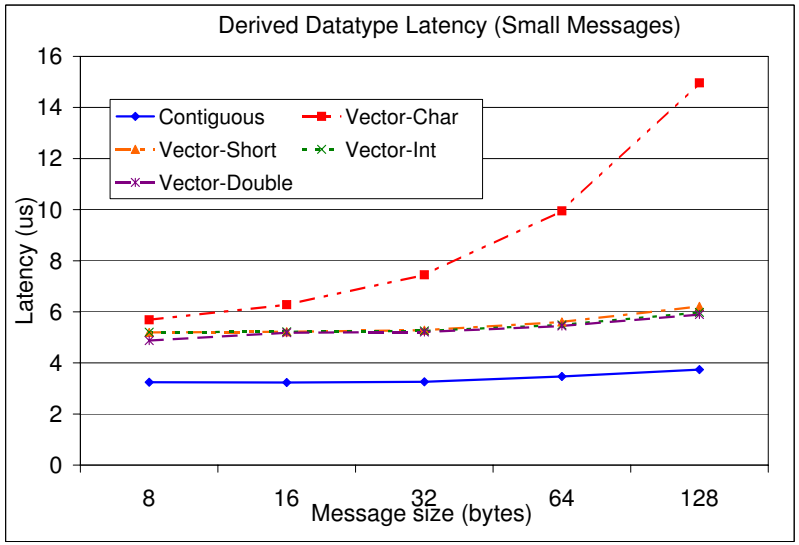


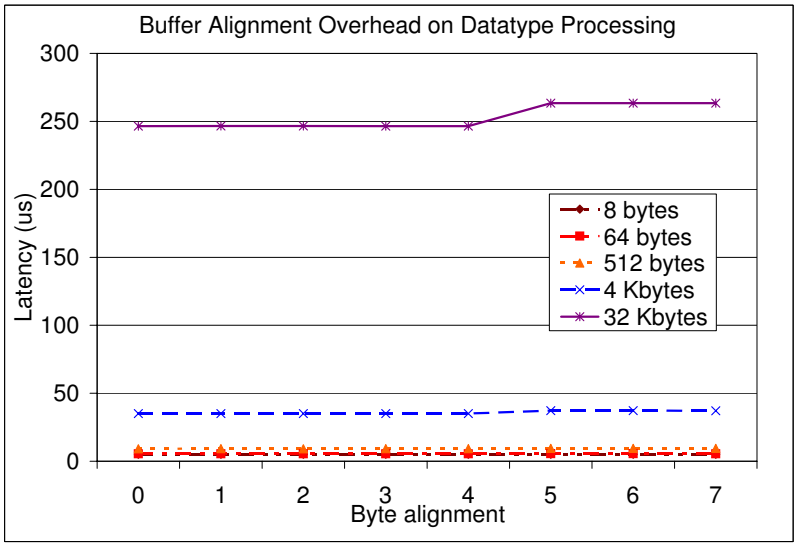
Image cited from paper

The experiment shows a roughly $2\mu s$ gap in performance between contiguous send and a send of short, integer or double precision data with a stride of two.

Buffer Alignment Overheads

- The buffer alignment could add another factor of overhead to the communication.
- In this experiment, we measure the communication latency of a vector of integers (4 bytes) with a stride of 2.
- We perform the test for different alignment of these integers, "0" for perfect alignment to a double-word boundary, "1" refers to an misalignment of 1-byte, etc.

Experiment result for Buffer Alignment Overheads



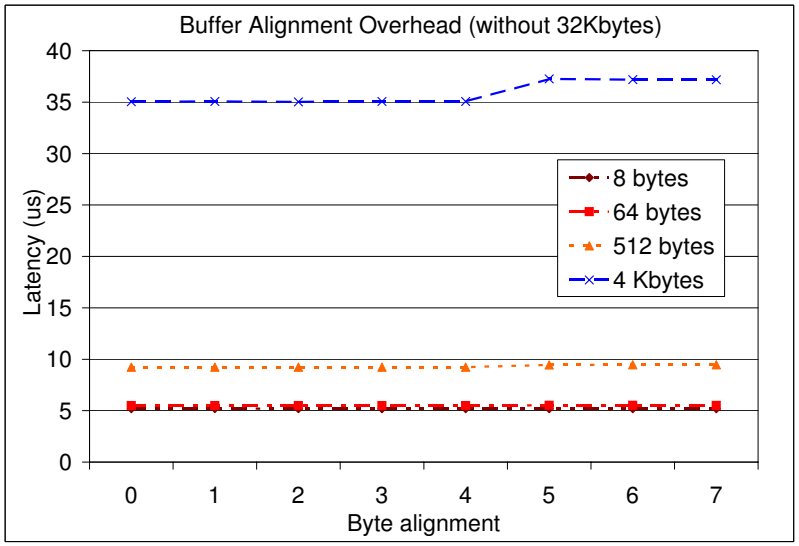


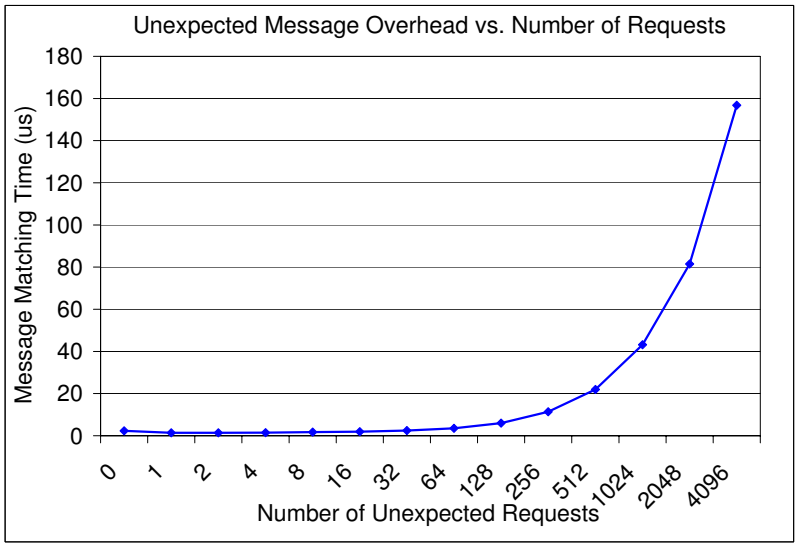
Image cited from paper

As long as the integers are within the same double-word (0-4), the performance is better as compared to when the integers span different double-words. The difference is about 10%.

Unexpected Message Overhead

- In MPI, when the receiver tries to receive the message it needs, all the previously sent messages are considered *unexpected* and are queued within the MPI stack for later use.
- Such queueing and dequeueing of request, along with possible data copying, can add overhead.

Experiment result for Unexpected Message Overheads



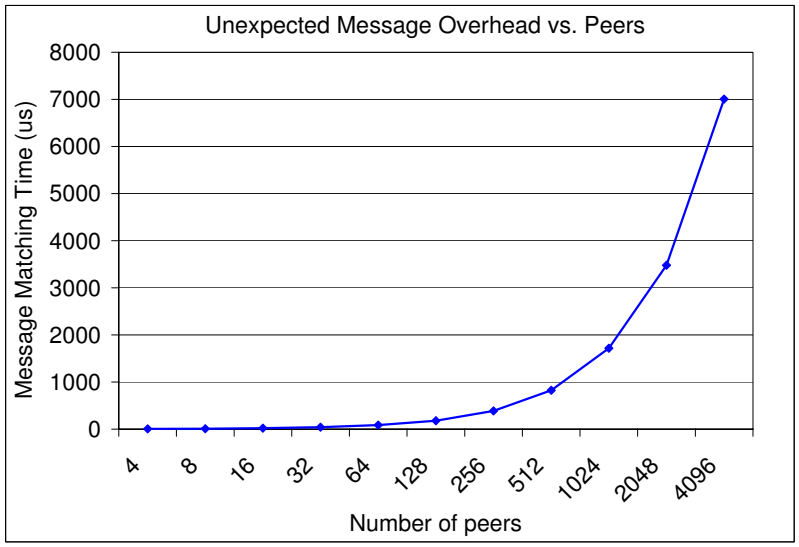


Image cited from paper

In the experiment, the time taken to receive the desired message increases linearly with the number of unexpected messages.

Thread Communication Overhead

- To support flexible hybrid programming model such as OpenMP plus MPI, MPI allows applications to perform independent communication calls from each thread by requesting for MPI_THREAD_MULTIPLE.
- The cost of this flexibility is that MPI has to perform appropriate locks within shared regions to avoid conflict or race conditons.
- Such locking will either add overhead or serialize communication.

Experiment result for Thread Communication Overheads

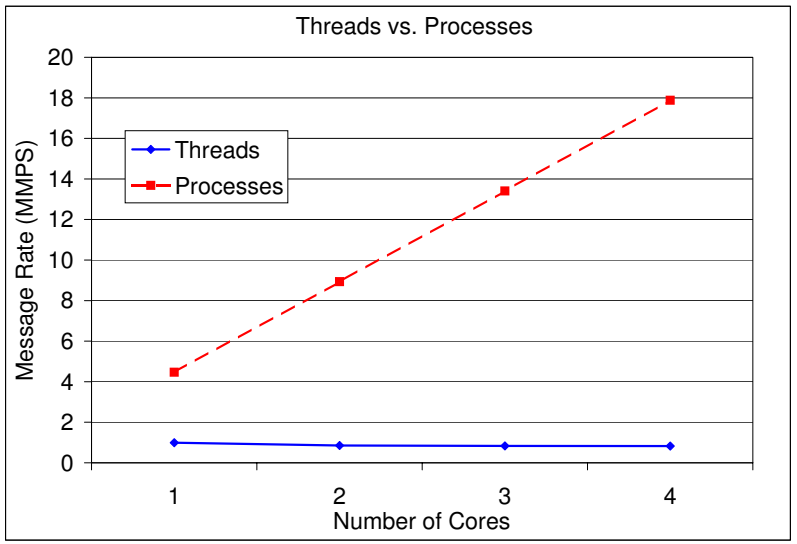


Image cited from paper

The experiment result shows that using multithread mechanism inside pure MPI implementation will not help speed up the execution.

Conclusion

- We have identified several bottlenecks in the MPI stack.
- Some of them comes from MPI itself, while others can be avoided by careful design.
- We also see that flexibility is not always a better choice.

END

Thank you!