Optimising the MPI Library for the T3E

Stephen Booth

EPCC, University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ
s.booth@epcc.ed.ac.uk

Abstract. This paper describes an optimised MPI library for the T3E.[1] Previous versions of MPI for the T3E were built on top of the SHMEM interface. This paper describes an optimised version that also uses additional capabilities of the low-level communication hardware.

The MPI library[1] is a fundamental part of many parallel applications. Almost any improvement to the performance of the MPI library will benefit a large number of application codes. This paper describes optimisation work for the T3E MPI library. We were able to utilise some special features in the T3E communication hardware to reduce message latency and improve the performance of global barriers.

1 Implementation of the MPI Library

The original implementation of MPI for the T3D was developed at EPCC[4]. This formed the basis for the MPI library shipped as part of the Cray Message passing Toolkit (MPT) for the T3E. The T3D communication hardware provided remote memory access between processors and some additional features such as atomic swap and barrier. The T3D MPI library was built on top of the Cray SHMEM library that provided direct access to all of these capabilities.

The fundamental building blocks of the library are "protocol messages". The sending and receiving processors need to exchange protocol messages to allow matching pairs of send and receive calls to be coupled together. Once the calls are coupled data is copied between processors using SHMEM put or get calls. Each processor needs to maintain a queue of incoming messages and be able to insert messages into remote queues. In the T3D library the queue was implemented using the SHMEM library. Protocol messages were 4-word data packets (corresponding to the packet size on the T3D network) and the queue was a symmetrically allocated (same address on all processors) buffer and lock variable. The lock was always manipulated using the SHMEM atomic swap, so in addition to a `shmem_put` to transfer the data at least 2 atomic swaps were needed to send a protocol message, one to lock and one to unlock the queue.

Several different protocols were used to implement the point to point communications depending on the size of the message.

[1] An earlier version of this paper was presented at the 6th European SGI/Cray MPP workshop Manchester, UK 7-8 September 2000.

- `T` Very small messages (< 3 words) were packed into the protocol message.
- `Tn` Medium sized messages were packed into several contiguous queue slots.
- `RTA` In the Request Transfer Acknowledge protocol the sending processor sends a request to the receiver. When the receiver matches this send to a receive it uses `shmem_get` to fetch the data and sends an acknowledge back to the sender. `RTA` had a higher latency than Tn but better bandwidth due to fewer memory copies so it was used for larger messages.
- `RAT` The Request Acknowledge Transfer protocol introduced an additional protocol message but allowed data to be sent using `shmem_put` which had better performance than `shmem_get` on the T3D. However unless the send was synchronous (e.g. `MPI_Ssend`) extra memory copies were needed so this mode was only used in special cases.

2 T3E Optimisations

The T3D MPI library was ported to the T3E almost without change. The hardware differences of the two machines were hidden behind the SHMEM interface. The MPI communication bandwidth is determined by the SHMEM put/get bandwidth so there was little scope for improving this. However the message latency was improved by making protocol messaging more efficient. The T3E communication hardware is quite different to that of the T3D, in addition to capabilities needed to implement SHMEM there is direct hardware support for message queues. These queues are *very* similar to the protocol queue used by MPI. The only major difference is that the packet size is 8 words (64 bytes) instead of 4. Messages are atomically inserted into the remote queue so no locks are required. No routines were provided by Cray to access these functions so it was necessary to write two additional custom routines to send and receive using the hardware message queues[2]. These were then integrated into the MPI library by modifying two macros in the code:

1. `PUTP` This sent a protocol message to a remote processor.
2. `BC_ABSORB` This processed any incoming protocol messages in the queue.

The `RAT` protocol was introduced to take advantage of the better `shmem_put` performance on the T3D. This was only used for large messages where the impact of the additional protocol messages is small so it was retained where convenient to do so. The `Tn` protocol cannot be implemented with the hardware messaging and was removed completely.

The following results were generated using the MPICH `mpptest` program[2]. All runs were performed on 8 processors of the 344 PE T3E-900 at EPCC using MPI standard sends. Message buffering was suppressed by setting the environment variable `MPP_BUFFER_MAX=0`.

For large messages (16 Kb and above) performance can be fit to a straight line and the asymptotic latency from these fits has been reduced from 51 to 12 microseconds.

[2] Thanks to David Tanqueray of Cray-UK for assistance with this.

The behaviour for short messages is shown in Figure 1; again message latency has been significantly reduced. This is especially true for very short messages using the `T` protocol where the latency is almost the same as for a SHMEM operation. As the protocol messages are larger up to 7 words of data can now be sent using the `T` protocol. This graph also shows an improved message bandwidth because for the larger message sizes on this graph the new version is using the `RTA` protocol instead of `Tn` which required an additional memory copy.

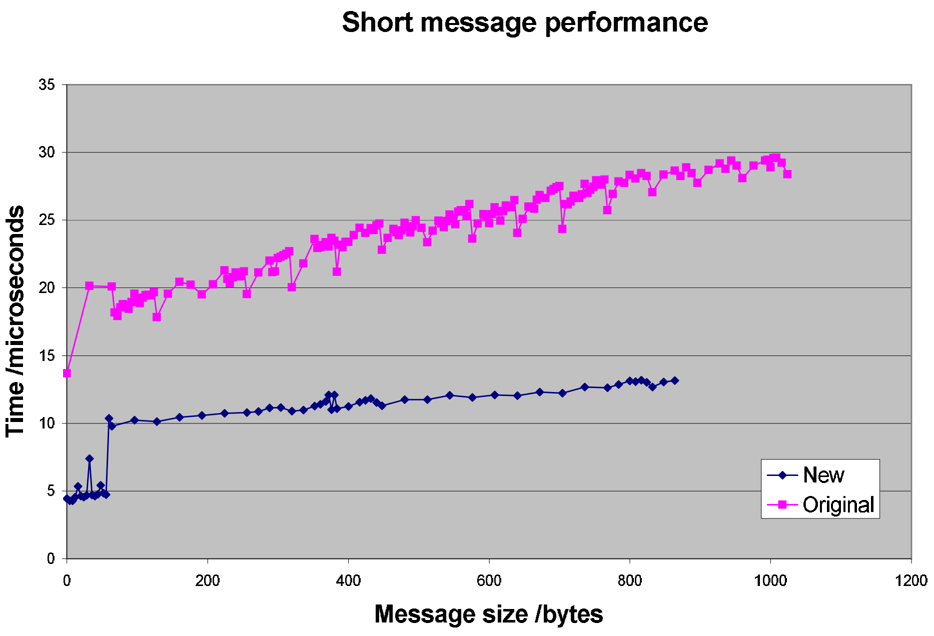

Fig. 1. Short message performance

This figure shows the performance of short messages for the two versions of the library.

The MPI standard mandates that point to point messages must continue to progress while a processor is blocked in a `MPI_Barrier` call. On the T3D this prevented the use of the hardware barrier. On the T3E the Barrier hardware can be polled. This allows global MPI barriers to utilise the barrier hardware for barriers across `MPI_COMM_WORLD` or other global communicators. This increased performance of global barriers is shown in Figure 2. This benefit increases as a larger number of processors are used.

All of the above optimisations are incorporated in the Cray MPT release 1.3.0.5 or above.

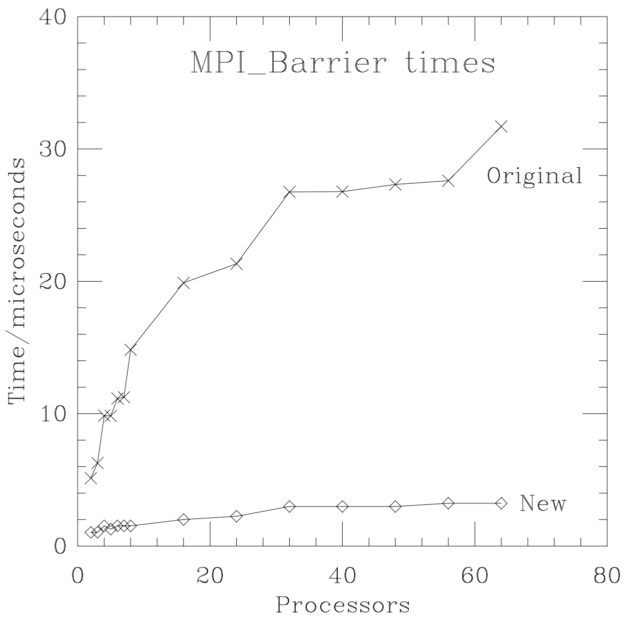

Fig. 2. Global barrier performance

This figure shows the time taken to complete a global barrier as a function of the number of processors being synchronised.

3 Conclusion

By dropping below the SHMEM interface, significant performance improvements have been made to the MPI library. This should improve the performance of any application code that makes extensive use of global MPI barriers or is sensitive to MPI message latency.

References

1. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard June 1995 `http://www.mpi-forum.org/` 80
2. W. Gropp, E. Lusk, N. Doss, A. Skjellum: A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing 22 789-828 September 1996. 81
3. R. Barriuso, A. Knies: SHMEM User's Guide for C., Cray Research Inc., August 1994.
4. K. Cameron, L. Clarke, A. Smith: "CRI/EPCC MPI for CRAY T3D", 1st European Cray T3D Workshop, EPFL, 6 September 1995. 80