

## Improved MPI with RDMA

Alan Gray, Joachim Hein and Stephen Booth  
*EPCC, The University of Edinburgh, James Clerk Maxwell Building,  
Mayfield Road, Edinburgh, EH9 3JZ, UK*

June 13, 2005

### **Abstract**

The effect of the newly available RDMA data transfer feature on HPCx is investigated using MPI benchmarks. Significant bandwidth increases are observed with the use of RDMA: with Ping Pong benchmarks, where 2 processes on separate frames communicate with one another, switch bandwidth is almost doubled and with Multi Ping Pong benchmarks, where 32 groups of 2 processors filling 2 frames communicate, around 25% improvement is seen.

**This is a Technical Report from the HPCx Consortium**

**© HPCx UoE Ltd 2005**

Neither HPCx UoE Ltd nor its members separately accept any responsibility for loss or damage from the use of information contained in any of their reports or in any communication about their tests or investigations.

## 1 Introduction

The Remote Direct Memory Access (RDMA) data transfer feature has recently been made available on HPCx due to a upgrade of the microcode operating the switch (Service Pack 12). In this paper we investigate the effects on bandwidths resulting from the use of RDMA.

## 2 Remote Direct Memory Access

RDMA allows the network adapter direct access to the user memory of the application. This allows the network adapter to move data between computational tasks with minimal CPU involvement, offering greater potential for overlapping calculation and communication when using non-blocking communication. RDMA enables large messages to utilise both network adapters attached to each of the compute nodes of the HPCx system without utilising multiple CPUs. This promises a substantially increased bandwidth for large enough messages [1]. In this report we concentrate on the bandwidth improvement. We intent to investigate the potential for overlapping computation and communication in the future.

RDMA requires the memory holding the data involved in the transfer to be pinned [2]. If an application sends the same large array(s) repeatedly during its execution, e.g. subsequent iterations, the set-up costs involved in pinning the memory will be amortised. On the other hand, if the location of your data in memory keeps changing during execution, the costs for pinning the changing memory locations might reduce performance. We recommend benchmarking your own application to determine whether it benefits from RDMA.

## 3 Hardware Review

The HPCx system features 50 IBM p690+ compute nodes (frames), each containing 32 IBM Power4 64-bit RISC processors. Each processor operates at 1.7GHz.

The 32 processors in a compute node are packaged into 4 Multi-Chip Modules (MCMs), each containing 8 processors. Within a node, MCMs communicate via a 4-way bus interconnect. Communication between nodes is provided by an IBM High Performance Switch (HPS). A total of 4 inter-node links are available: each node has 2 network adapters, each with 2 links.

## 4 How to enable RDMA

RDMA is disabled by default. Users can enable RDMA by simply adding the line

```
#@bulkxfer = yes
```

to the header of their loadleveler script. See the HPCx User's Guide for a full sample script [3].

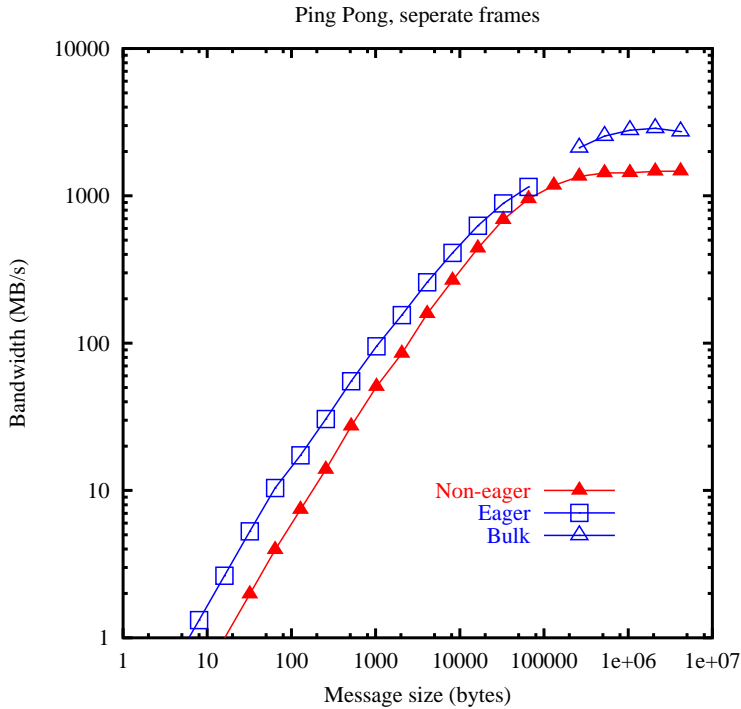


Figure 1: The dependence of the PingPong benchmark time on the message length. Closed triangles and open squares represent eager limits of 0KB and 64KB respectively. Open triangles represent RDMA being switched on.

## 5 Results

The Intel MPI Benchmark Suite [4], formerly known as the Pallas MPI Benchmark Suite, was used to investigate the effects of RDMA. PingPong and Multi PingPong benchmarks were run, each providing bandwidth measurements for a variety of message lengths.

The PingPong benchmark involves direct communication between 2 processes. Runs were done with 2 processes each on separate frames, for eager limits of both 0KB and 64KB. This was repeated with RDMA switched on, with the minimum message size for bulk transfer (controlled by the `MP_BULK_MIN_MSG_SIZE` environment variable) set to the default of 150KB (which, with further testing, was seen to be a reasonable value). Figure 1 plots the resulting bandwidths against message size. As expected, switching on the eager limit of 64KB gives improvement up to this message length. RDMA is clearly seen to result in bandwidths almost doubling in its large message length regime. With RDMA enabled, bandwidths of just under 3GB/s are observed for message lengths of about 1MB.

The above PingPong communications do not saturate the inter-frame switch and a more rigorous and realistic test is provided by the Multi PingPong benchmark with 64 processors (32 groups of 2) on 2 frames. The code has been modified to ensure that

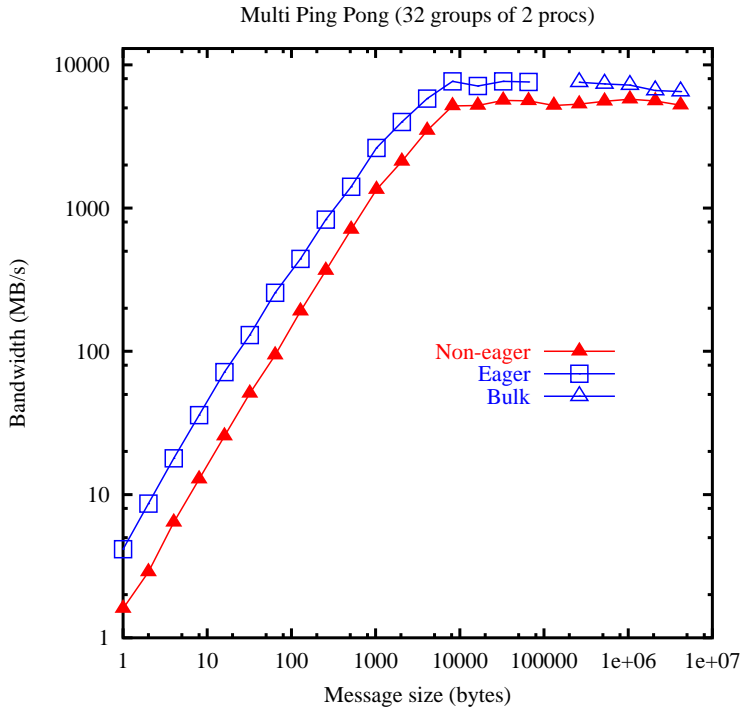


Figure 2: The dependence of the Multi PingPong benchmark time on the message length. Closed triangles and open squares represent eager limits of 0KB and 64KB respectively. Open triangles represent RDMA being switched on.

all communications utilise the switch. Again, from Figure 2, improvement is seen with RDMA in the large message length region. In this case the bandwidth improvement is around the 25% level, and a maximum bandwidth of 7.5GB/s is observed.

## 6 Conclusions

Enabling RDMA is seen to significantly improve bandwidths in both Ping Pong and more demanding Multi Ping Pong tests. Bandwidth increases are expected in applications which repeatedly send the same large array, but performance could be reduced for applications in which data regularly changes location in memory. Users are advised to benchmark their application to determine if its performance can be improved with the use of RDMA.

## 7 Acknowledgements

We wish to thank J. Follows and C. Grassl at IBM for their help with understanding and enabling RDMA.

## References

- [1] “HPS Service Pack 12”, README,  
<http://techsupport.services.ibm.com/server/hps/related/HPSservicePack12.html>
- [2] R. Treumann, presentation at ScicomP11, May 30 to June 3, 2005, Edinburgh, Scotland, UK and private communication
- [3] “HPCx User’s Guide”  
<http://www.hpcx.ac.uk/support/documentation/UserGuide/HPCxuser/HPCxuser.html>
- [4] Code and documentation available from  
[http://www.intel.com/software/products/cluster/downloads/mpi\\_benchmarks.htm](http://www.intel.com/software/products/cluster/downloads/mpi_benchmarks.htm)