

Communications on SP7 and SP9

Adrian Jackson

October 7, 2004

Abstract

IBM's communications libraries on HPCx, LAPI and MPI, have recently received a software update, Service Pack 7. This report re-evaluates the performance of both communication libraries in light of this update, and also assesses the impact of a future update (Service Pack 9) on the LAPI library. We find that the Service Pack 7 update significantly improves the performance of both communication libraries.

1 Introduction

This technical report is a followup to the technical report "*LAPI on HPS: Evaluating Federation*"[1], which documents the performance of IBM's communications libraries on HPCx. More specifically it profiled the performance of the LAPI and MPI libraries, using a series of benchmarks.

Briefly, it documents the communication library hierarchy in which IBM's MPI is built on top of the LAPI library, and shows that even with this hierarchy the MPI library still outperforms the LAPI library for certain communication situations.

Since the publication of that report, IBM has developed a number of updates for their communication libraries, to address known performance issues. The most significant of these updates are contained within IBM's High Performance Switch (HPS) Service Pack 7 and Service Pack 9 upgrades. Currently HPCx is using SP7, meaning the LAPI and MPI libraries are benefiting from the improvements made available in the SP7 upgrade. However, IBM have also provided an updated, beta version, of the LAPI library, that contains the updates implemented in SP9, allowing us to evaluate the LAPI performance for both SP7 and SP9. It is intended that the full Service Pack 9 be installed on HPCx at the end of October 2004.

Since these updates affect the performance of the communications libraries on HPCx we have rerun the benchmarks from the previous report to establish the current performance levels. This report documents these benchmark results.

2 The HPCx system

The current HPCx hardware entered user service in late May 2004, and consists of fifty IBM p690+ compute frames plus two IBM p690 service frames. Each IBM p690+ contains 32 1.7 GHz IBM Power4+ processors and 32 GB of main memory. The frames are connected with IBM's High Performance Switch (HPS) network. There are two network adapters per frame, each adapter providing two links, giving a total of 4 links per frame. The system is operated as a cluster of 32-way SMP nodes.

There are two methods of communication on HPCx, off-node communication through the HPS, and on-node communication through shared-memory. Both methods have different performance characteristics, with on-node communication having a much lower latency, and higher bandwidth, than off-node communication. However, on-node communication can only be used to transfer data between, at most, 32 processor. This means, that for a large system such as HPCx, the off-node performance is most likely to be the limiting factor for the performance of users jobs, so this is the communication method we will be concentrating on.

For this investigation we used IBM's xlc C compiler. Unless stated, we used the following compiler arguments; -q64 when building the executable with the mpcc_r script. The standard unit of data used in the benchmarks was the integer, where an integer represented by 4 bytes of data.

3 LAPI

As previously mentioned, on HPCx, LAPI is available in both SP7 and SP9 flavours, with the default version of library containing only SP7 performance improvements. This means that the LAPI library can be evaluated to compare the performance of both the SP7 and SP9 updates. For this report we only document basic LAPI performance, using the LAPI_Put and LAPI_Get routines, that are the core of the Remote Memory Access (RMA), or single-sided, part of the LAPI library. LAPI does contain routines for more complex communication and data transfer functionality, such as non-contiguous data transfer, and active messages, but for this report we are only concerned in benchmarking the basic functionality of LAPI.

3.1 Put and Get

The Put and Get functions in the LAPI library perform basic single-sided operations, transferring data from the memory space of one process (the origin) to that of another process (the target), without the involvement of the target process.

Both of these functions were benchmarked by performing a Put (or Get) operation a number of times for a range of buffer sizes, and timing how long the transfer takes (details can be found in the previous technical report [1]). The benchmarks were repeated for both the LAPI libraries (SP7 and SP9 variants). The benchmarks were also repeat using different numbers of processors in a range of configurations (i.e. up to 64 processors, over 2 nodes).

Combined results from these benchmarks are shown in Figure 1 and Figure 2 for the Get routine, and Figures 3 and 4 for the Put routine. The legend 1N2P signifies that the benchmark was performed using 2 processors on 1 node. Likewise 2N1P shows that 2 nodes were used, each with 1 processor. This means that in both case 2 processor were used, although in the first case communications are performed on-node (i.e. using shared memory), and the second case communications are performed off-node (i.e. using the switch network).

We can see from the graphs that there is a significant performance improvement, for both Put and Get, when moving from Pre SP7 to the SP7 library, especially for off-node communications. Moving from SP7 to SP9 has less of an impact on communication performance. In fact, the off-node performance does not seem to change, although on-node performance does improve for small message sizes.

The performance of Put and Get in the SP9 library is compared in Figures 5 and 6, and shown in detail for both routines in Figures 7, 8, 9, and 10. The detailed graphs show performance of the benchmarks on up to 64 processors, across 2 nodes.

We can see from the comparison of Put and Get using the SP9 library there is little difference between the performance of the two routines. For small, on-node, messages Get appears to have slightly better performance, whereas the opposite is true for small, off-node, communications. For large messages, the performance is identical for both on-node, and off-node, communications.

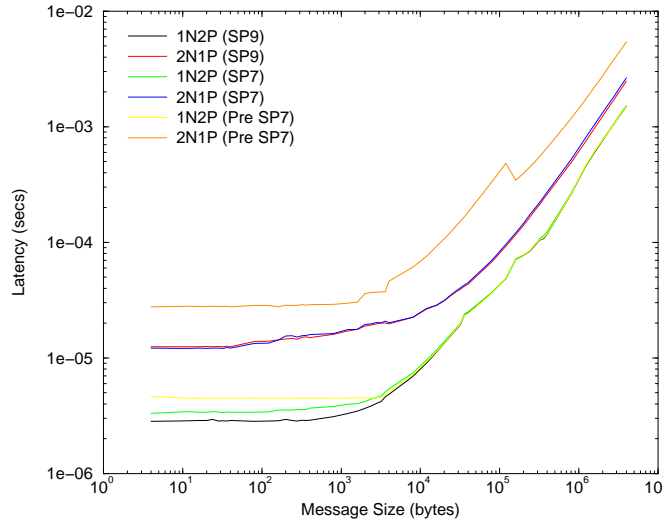


Figure 1: Comparison of Get Time using Pre SP7, SP7, and SP9 libraries with 2 Processors

The graphs showing the comparison of the various LAPI libraries (Pre SP7, SP7, and SP9) give us a LAPI latency of approximately $1.1e^{-05}$ seconds for off-node communications, and $2.8e^{-06}$ seconds for on-node communications. This compares with a Pre SP7 latency of $2.7e^{-05}$ seconds for off-node communications, $4.6e^{-06}$ seconds for on-node communications.

Figures 7, 8, 9, and 10 show the performance of the benchmarks when a range of processor numbers are used. Up to 64 processor are used, in both on-node and off-node configurations. As HPCx has nodes containing 32 processors, the largest number of processors that can be used for on-node communications benchmarks is 32. The largest off-node communication benchmark performed consisted of 2 fully occupied nodes (i.e. 2 lots of 32 processors).

These benchmarks we performed using the SP9 library. We can see that communication performance scales well as the number of processor used is increased, with linear scaling in off-node performance using up to 4 processors per node. This is to be expected, as each HPCx node has 4 communication links. When fully using these 4 links, Figure 8 shows that we can achieve a bandwidth of approximately 1.5 GB/s per processor, giving an aggregate bandwidth of 6 GB/s. However, when using 8 processors, each process can achieve up 1.0 GB/s of data transfer, giving an aggregate bandwidth of 8 GB/s. This is an improvement of 5 GB/s on the peak aggregate bandwidth of the Pre-SP7 library, which was approximately 3 GB/s, and shows that with the new LAPI library we need at least 8 processors per node to obtain the peak aggregate bandwidth from a node.

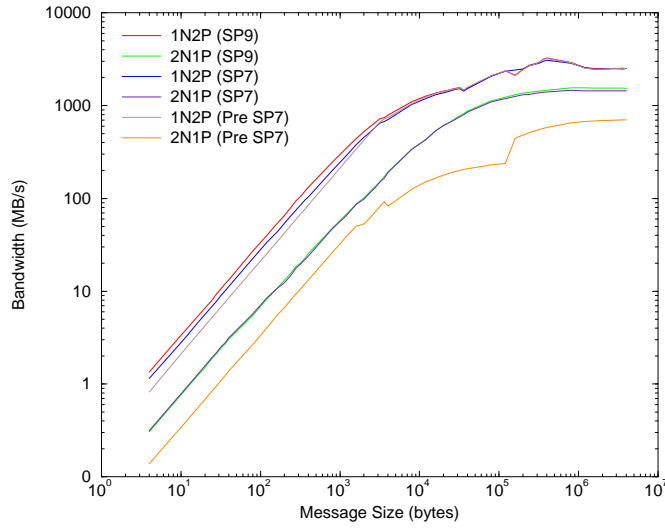


Figure 2: Comparison of Get Bandwidth using Pre SP7, SP7, and SP9 libraries with 2 Processors

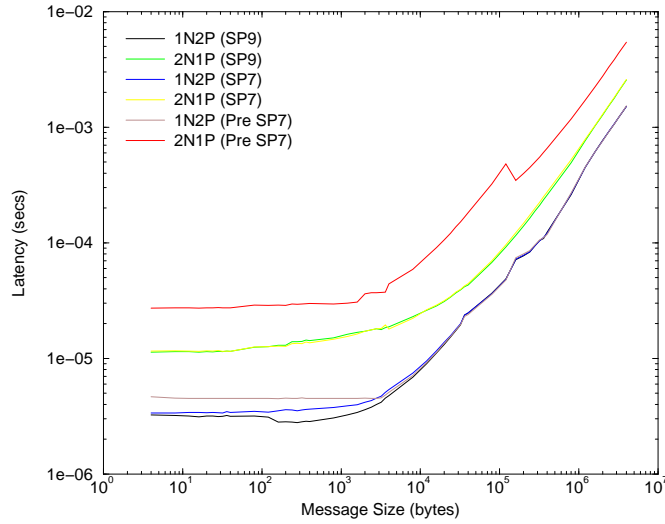


Figure 3: Comparison of Put Time using Pre SP7, SP7, and SP9 libraries with 2 Processors

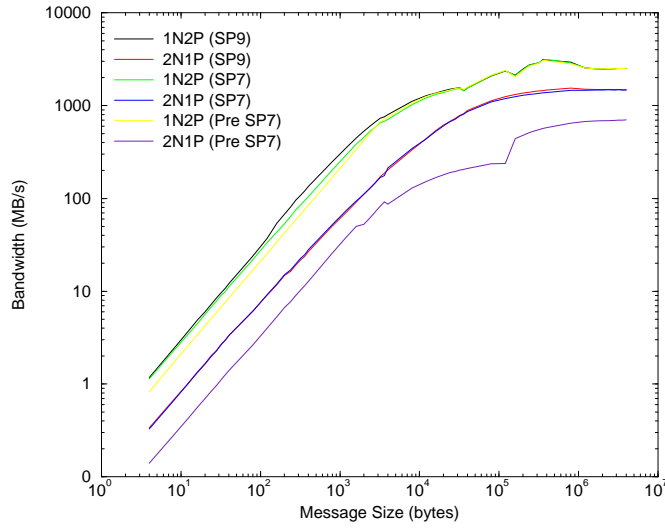


Figure 4: Comparison of Put Bandwidth using Pre SP7, SP7, and SP9 libraries with 2 Processors

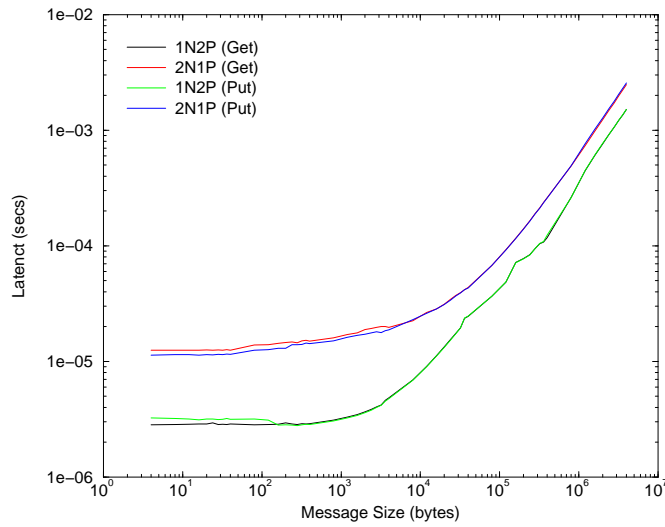


Figure 5: Comparison of Put and Get Time using 2 Processors

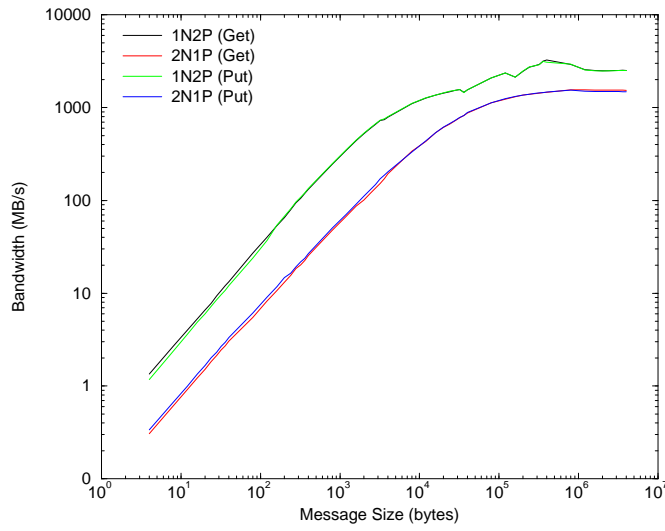


Figure 6: Comparison of Put and Get Bandwidth using 2 Processors

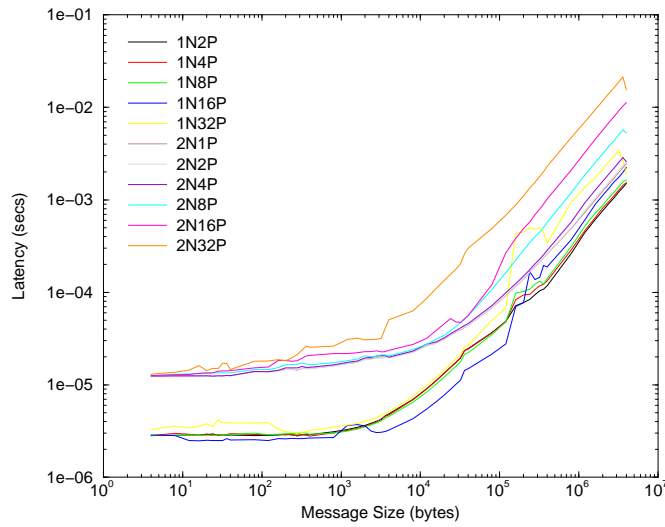


Figure 7: Get Time using up to 64 Processors

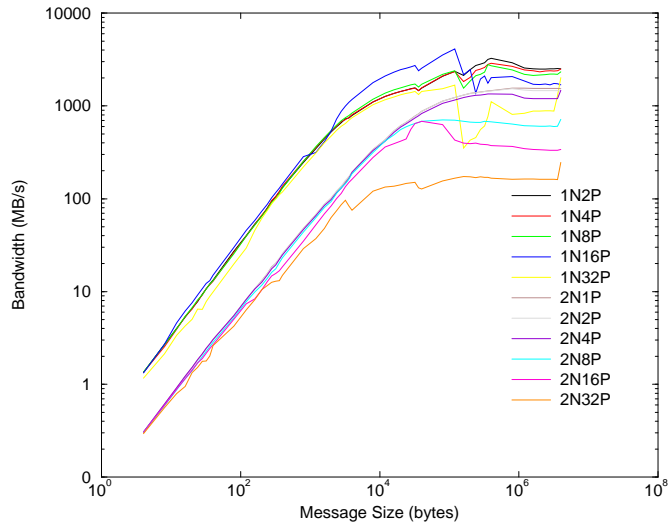


Figure 8: Get Bandwidth using up to 64 Processors

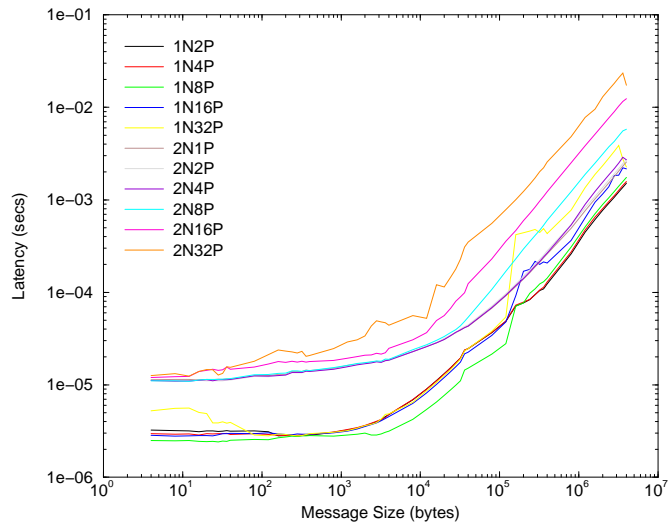


Figure 9: Put Time using up to 64 Processors

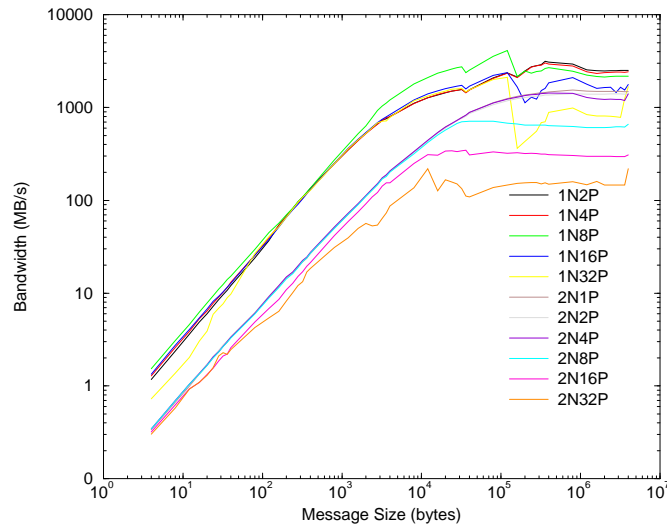


Figure 10: Put Bandwidth using up to 64 Processors

3.2 PingPong

The LAPI PingPong benchmark uses pairs of processes, each exchanging data using `LAPI_Put`. It works in a strict pingpong fashion with only one process communicating at any given time, and the other process waiting for that communication to complete before it starts its communication.

As with the Put and Get benchmarks, the benchmark is performed using a range of message sizes, and different combinations of processor and nodes numbers. This allows us to assess performance for a wide combination of communication patterns.

The pingpong benchmark is the same as the one used for MPI benchmarking, except for the method of communication (i.e. the benchmark harness is the same). This allows us to compare the LAPI and MPI performance.

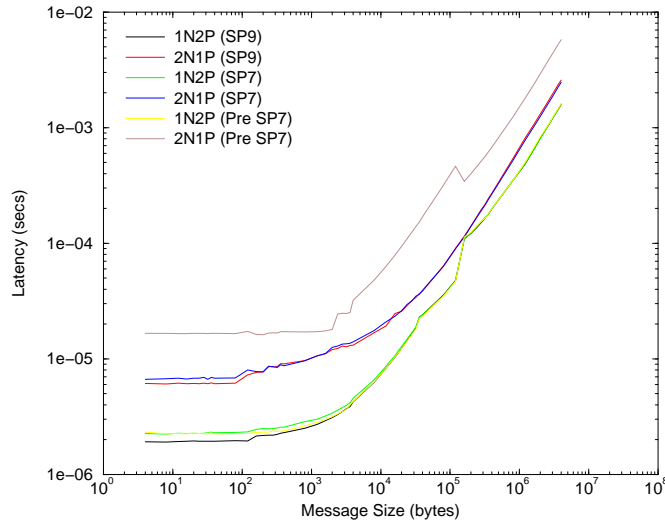


Figure 11: LAPI PingPong Large Message Time using Pre SP7, SP7, and SP9 Libraries on 2 Processors

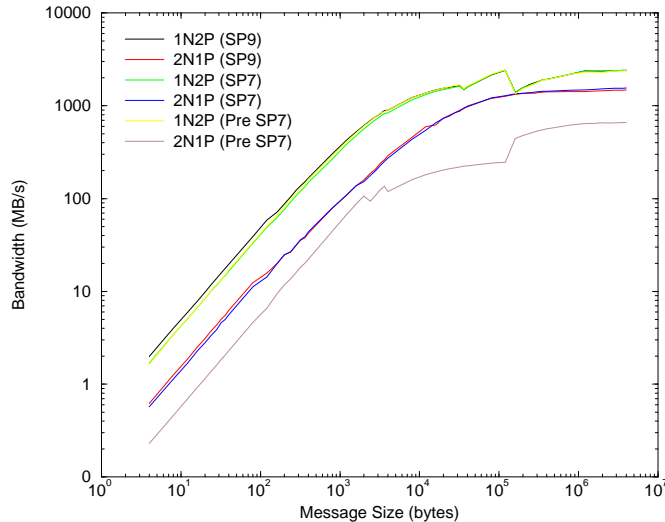


Figure 12: LAPI PingPong Large Message Bandwidth using Pre SP7, SP7, and SP9 Libraries on 2 Processors

Two message ranges are used for the performance analysis, a small message benchmark (from 1 to 2000 integers), and a large message benchmark (1 to 1 million integers). Again, each benchmark is

performed a number of times, for a range of processor and node numbers, using both of the new LAPI libraries.

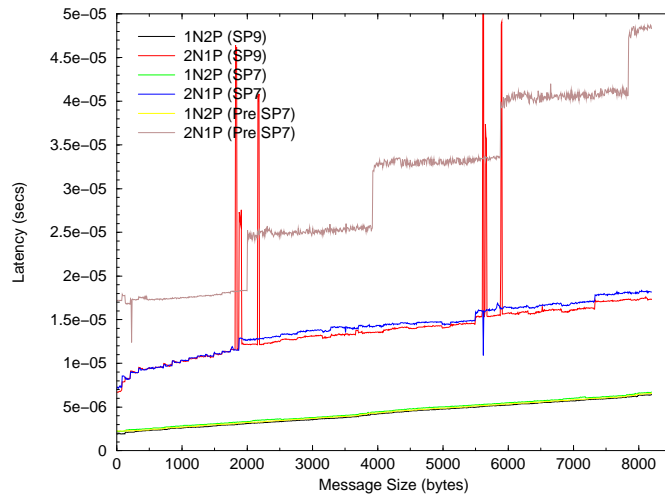


Figure 13: LAPI PingPong Small Message Time using Pre SP7, SP7, and SP9 Libraries on 2 Processors

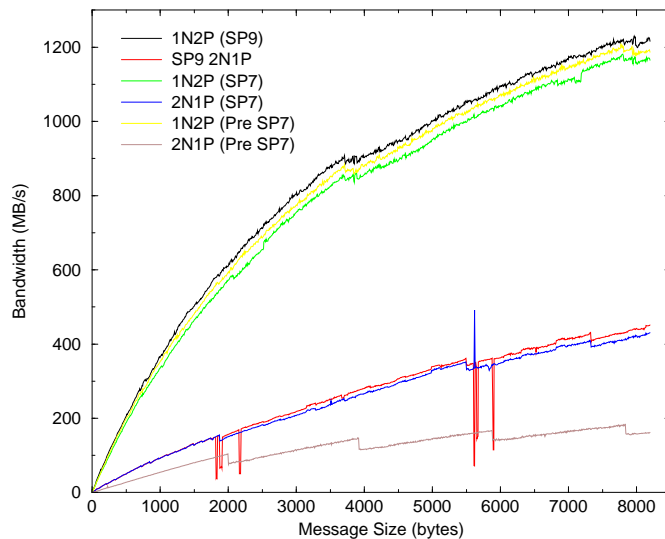


Figure 14: LAPI PingPong Small Message Bandwidth using Pre SP7, SP7, and SP9 Libraries on 2 Processors

Figures 11, and 12 compare the large message benchmark performance for all three libraries using 2 processor (in both on-node and off-node configurations). As with the Put and Get benchmarks, we can see there is a large performance improvement when moving from Pre SP7 to SP7, especially for off-node communications.

Figures 13, and 14 show the performance for the small message benchmark for the three libraries. From these graphs we can see that whilst performance is very similar for all three libraries when performing on-node communications, the performance of off-node communications changes significantly when using the different libraries. The Pre SP7 library shows marked performance degradation when messages sizes cross packet boundaries (i.e. around the 2k, 4k, 6k, and 8k message sizes). The SP7 and SP9 libraries show no such performance characteristics. This offers significant improvements for codes which send small messages.

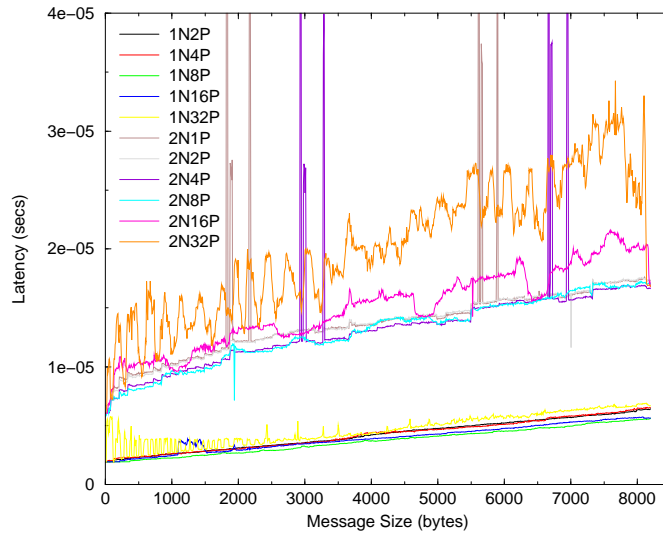


Figure 15: LAPI PingPong Small Message Time using the SP9 Library on up to 64 processors

Figure 15 graphs the time to perform the small message PingPong on a range of processor numbers. We can see that for this message range, for off-node communications, performance scales linearly up to 8 processors, and then falls off as the nodes become saturated.

4 MPI

The current version of the MPI library on HPCx is based on SP7, and we do not have an SP9 version of the library available at the moment. This means that we can only evaluate the performance of SP7 for MPI, and compare it with that of the Pre SP7 library on HPCx.

The main benchmark for the MPI performance is the PingPong benchmark, as used in the LAPI performance analysis. It can use any MPI point-to-point communication routine, but is mainly used with MPI_Send (standard send).

Prior to the SP7 upgrade, MPI had lower latencies than LAPI Put or Get, even though MPI uses LAPI primitives to perform communications.

4.1 PingPong

As with the LAPI pingpong benchmarks, data is sent between pairs of processes, with each process taking turns to send data, and only one of the pair communicating at any given time. The message pingpong is repeated many times, with an average time being calculated. It is also repeated for a range of message sizes.

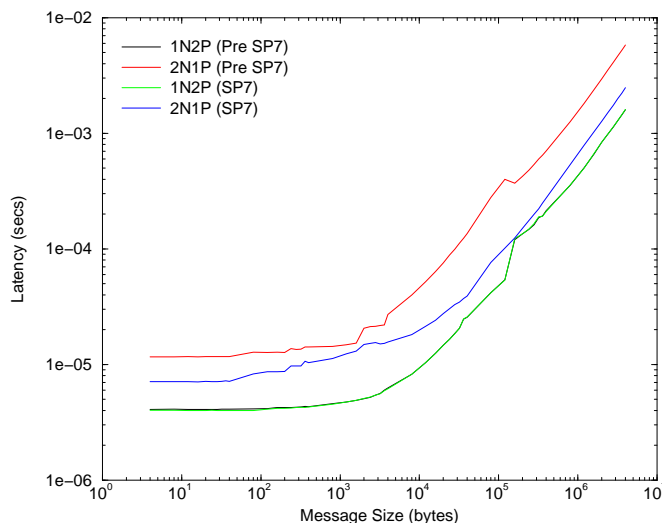


Figure 16: MPI PingPong Large Message Time using both the SP7 and Pre SP7 Libraries on 2 Processors

Again, there were two message ranges for these benchmarks, large (1 to 1 million integers), and small (1 to 2000 integers). This allowed us to study both general communication characteristics, and those of small messages (i.e. the size that is generally used by user codes).

The results from the large message benchmarks are compared in Figures 16, and 17. These graphs show the results for the benchmark using 2 processors with both the SP7 library and the Pre SP7 library. We can see from the graphs that, as with LAPI, there is a significant off-node communication performance improvement with the SP7 update. Off-node message latency drops by $4e^{-06}$ seconds from $11e^{-06}$ seconds to $7.1e^{-06}$ seconds. Off-node bandwidth has also increase, from about 650 MB/s for a single processor, to about 1.5 GB/s. The aggregate bandwidth available to a node has doubled from approximately 4 GB/s with the Pre SP7 library, to 8 GB/s with the SP7 release.

The equivalent results from the small message benchmark are shown in Figures 18 and 19. The time and bandwidth graphs are very similar to the equivalent LAPI results, showing greatly reduced sensitivity to the size of message packets for off-node communications. With the Pre SP7 library, off-node message time increased from $1.1e^{-05}$ seconds to $3.9e^{-05}$ seconds over the small message range.

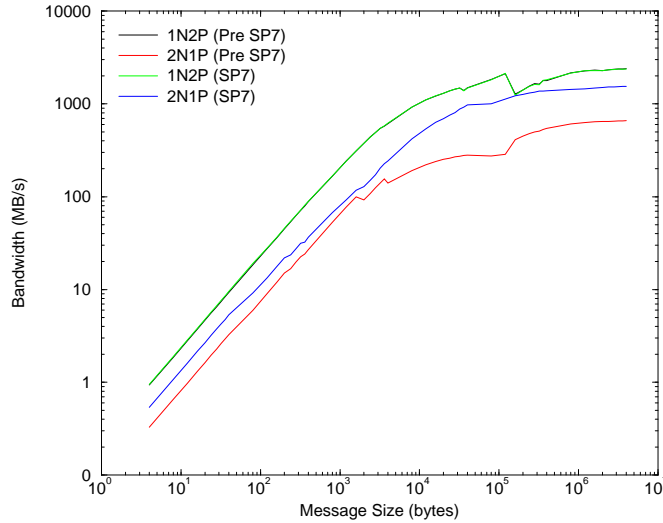


Figure 17: MPI PingPong Large Message Bandwidth using both the SP7 and Pre SP7 Libraries on 2 Processors

For the SP7 library the increase is from $7.1e^{-06}$ seconds to $18e^{-06}$ seconds. Again, on-node message performance hasn't changed.

The performance of both `MPI_Send` and `MPI_Ssend` is profiled in Figures 20, and 20. `Send` is MPI's standard send method which uses either a buffered or synchronous approach depending on the size of the message, whereas `Ssend` always sends messages synchronously. As previously mentioned, `MPI_Send` is the routine of choice for the majority of our MPI benchmarks, but some analysis of its relative performance is interesting. We can see from the graphs that `Send` significantly outperforms `Ssend`, using both on-node and off-node communications, for all but the largest messages. For the largest messages, at which point the *eager limit* is reached, all MPI messages are sent synchronously, so a call to `MPI_Send` is effectively a call to `MPI_Ssend`. We can see from the results that if a code replaced calls to `Ssend` with `Send`, then for small messages it would half its communication times. In fact, `Send` has a latency of approximately $7.1e^{-06}$ seconds compared with `Ssend`'s latency of $2.1e^{-05}$ seconds for off-node communication.

Figures 22, and 23 show the performance of the PingPong benchmark on up to 64 processors using the SP7 library. From these graphs we can see that the MPI SP7 library has similar scaling to the LAPI SP9 library, i.e. it scales linearly up to 4 processors per node for off-node communications, but losses linear scaling for large message sizes when using more processors per node. Likewise, the on-node communications scales linearly for all 32 processor for small and medium size messages, but breaks down when sending large messages.

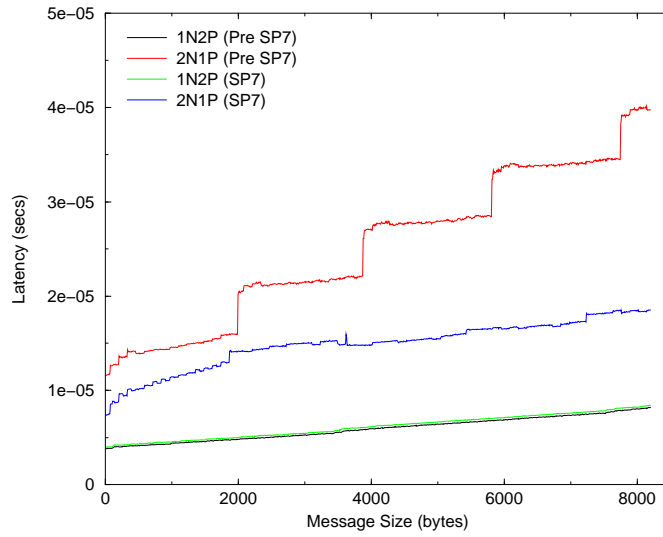


Figure 18: MPI PingPong Small Message Time using both the SP7 and Pre SP7 Libraries on 2 Processors

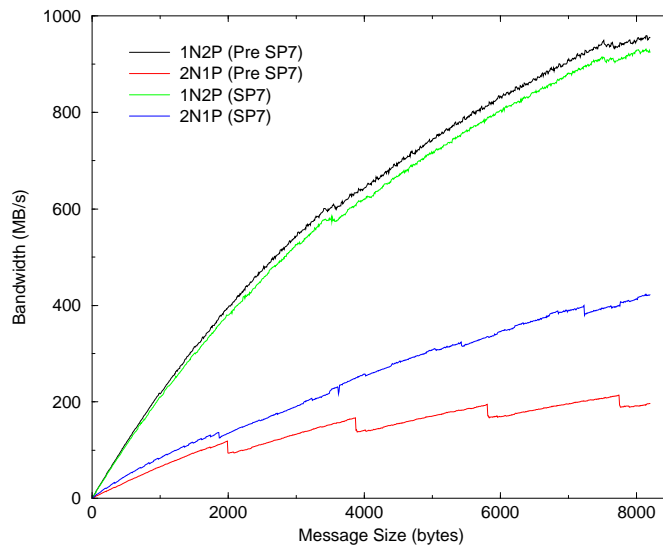


Figure 19: MPI PingPong Small Message Bandwidth using both the SP7 and Pre SP7 Libraries on 2 Processors

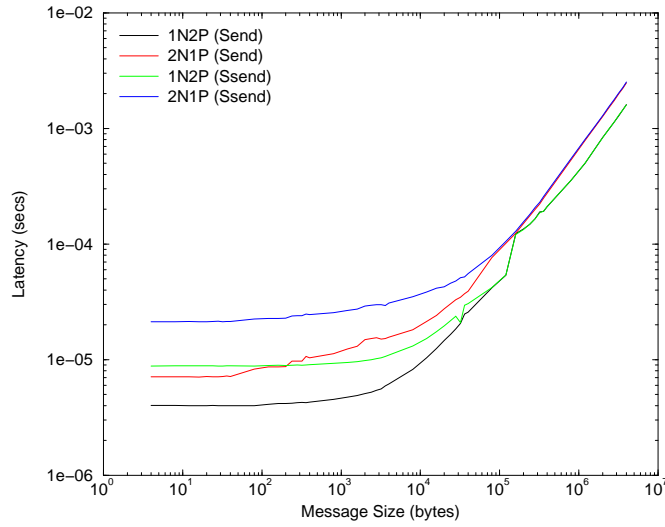


Figure 20: MPI Send and Ssend Time Comparison using the SP7 Library on 2 Processors

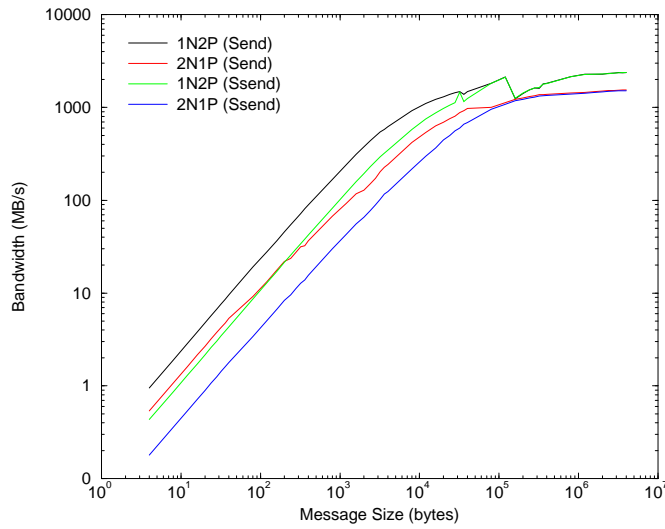


Figure 21: MPI Send and Ssend Bandwidth Comparison using the SP7 Library on 2 Processors

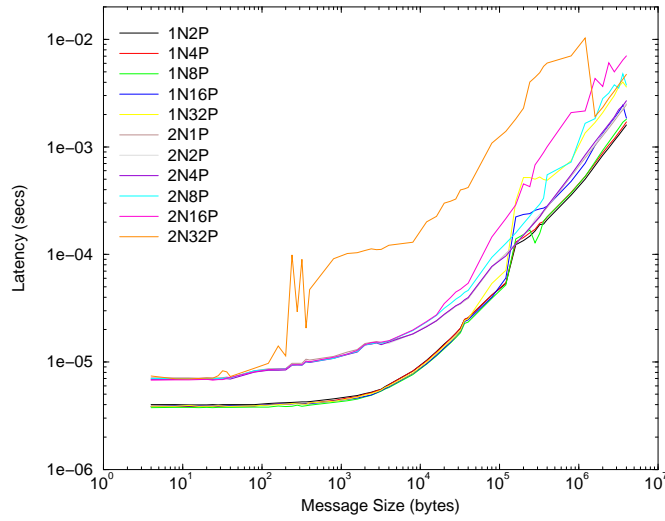


Figure 22: MPI Send PingPong Time using the SP7 Library on up to 64 Processors

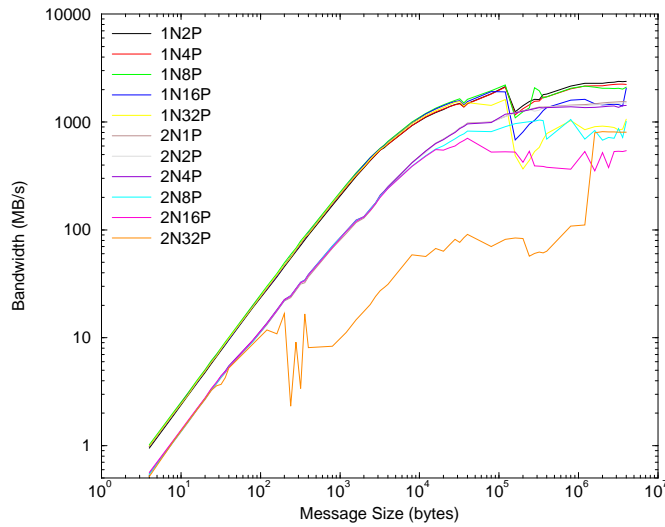


Figure 23: MPI Send PingPong Bandwidth using the SP7 Library on up to 64 Processors

4.2 AlltoAll

The poor performance of IBM's MPI AlltoAll function is one of the issues the SP7 and SP9 updates are intended to solve. Therefore we benchmarked the performance of the function before and after the SP7 update to monitor what performance improvement it gave.

Figure 24 shows the time to perform an AlltoAll operation using 512 processors, over a range of data sizes. The message size shown is that sent by each process to every other process, meaning that for the 512KB message size there is actually 256Mb of data being sent by each process. Each AlltoAll was repeat a large number of times, with the time shown the average time for one operation.

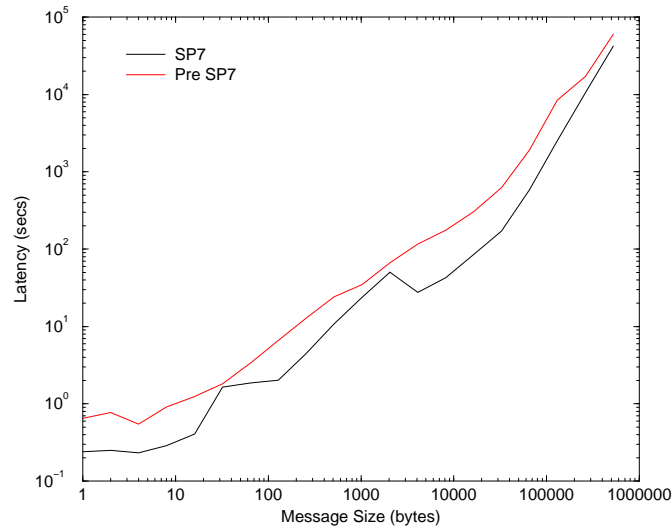


Figure 24: MPI AlltoAll using SP7, and Pre SP7 Libraries on 512 Processors

We can see from the graph that the performance has significantly improved after the SP7 update, especially for small buffer sizes. For a buffer size of 1 byte the time for an AlltoAll operation for the SP7 library is 0.24 seconds, which is nearly 3 times faster than that for the Pre SP7 library, which is 0.65 seconds. For 8 bytes, the equivalent times are 0.29 seconds to 0.9 seconds, which is over 3 times quicker.

5 LAPI vs MPI

The MPI library is constructed on top of the LAPI library, so it is of interest to compare the relative performance of both libraries. The results for the large PingPong benchmark using the SP7 library on 2 processors for both MPI and LAPI are shown in Figure 25. From this graph we can see that for off-node communication MPI and LAPI have very similar performance, but for on-node communication (using only 2 processors), LAPI significantly outperforms MPI for very small messages.

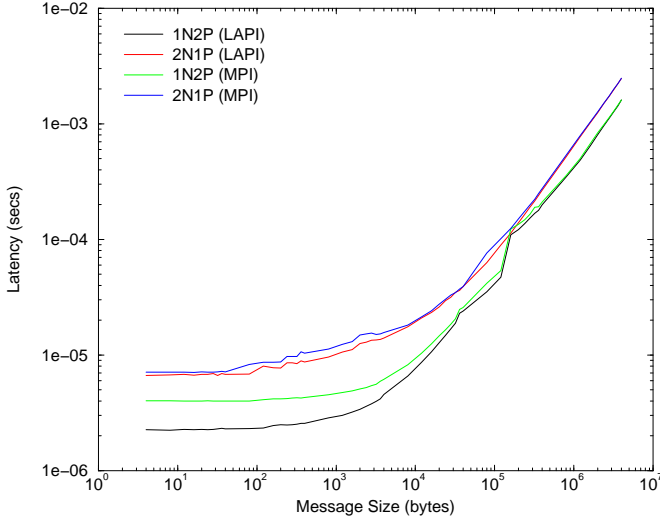


Figure 25: A Comparison of PingPong Time for LAPI and MPI using 2 Processors using SP7 libraries

This is a change from the performance of the Pre SP7 library, where MPI outperformed LAPI in the same benchmark for off-node communications, but LAPI still had better on-node communication (see Figure 26

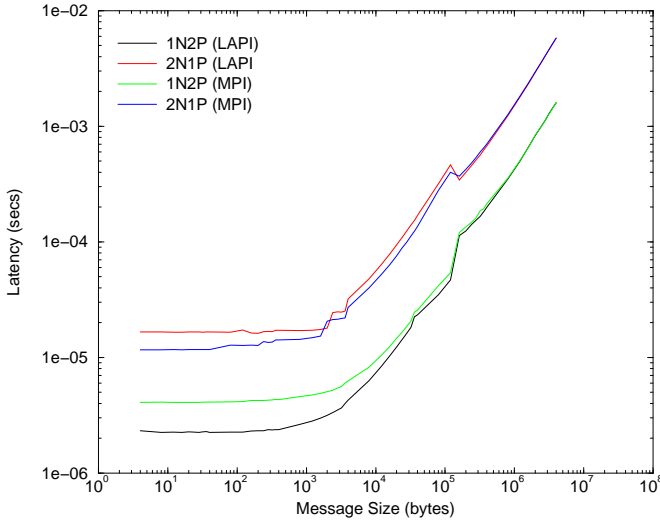


Figure 26: A Comparison of PingPong Time for LAPI and MPI using 2 Processors using Pre SP7 libraries

6 Summary

It is evident that the SP7 update has significantly improved the performance of both LAPI and MPI, halving latency for off-node communications and increasing bandwidth for small and large messages alike. We can also see that the SP9 update will not adversely affect performance, although it doesn't appear to significantly improve LAPI performance. However, the SP9 LAPI library used in these benchmark is a pre-release version, therefore when the library is officially installed on HPCx it may have different performance characteristics. The performance of communications on HPCx after the SP7 update is now in line with the stated performance of IBM's HPS.

We can see that LAPI and MPI performance is similar, and that the `MPI_Send` routine offers significant performance benefits for small messages when compared with the `MPI_Ssend` routine. The performance of MPI's AlltoAll function has also improved with the SP7 update, and it may improve further when the SP9 update is installed.

References

- [1] **LAPI on HPS: Evaluating Federation**, A. Jackson, EPCC, August 23rd, 2004
http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0407.pdf