LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-221775

Infiniband Performance Comparisons of SDR, DDR and Infinipath

M. Minich

June 2, 2006

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Abstract

This technical report will be comparing the performance between the most common infiniband-related technologies currently available. Included will be TCP-based, MPI-based and low-level performance tests to see what performance can be expected from Mellanox's SDR and DDR as well as PathScale's Infinipath. Also, we will be performing comparisons of the Infinipath on both OpenIB as well as PathScale's ipath stack.

Contents

List of Figures

1 Overview and Goals

Infiniband promises to bring high performance interconnects for I/O (filesystem and networking) to a new cost performance level. Thus, LLNL has been evaluating Infiniband for use as a cluster interconnect. Various issues impact the decision of which interconnect to use in a cluster. This technical report will be looking more closely at the actual performance of the major infiniband technologies present today. Performance testing will focus on latency, and bandwidth (both uni and bi-directional) using both TCP and MPI. In addition, we will be looking at an even lower-level (removing most of the upper-level protocols) and seeing what the connection can really do if the TCP or MPI layers were perfectly written.

1.1 Test Suite

All of the testing done in accordance with these results used the following benchmarks. For the Ohio State University (OSU), more information for these tests (as well as the source code) can be found on the Ohio State University's MPI over Infiniband Project website[1]. We are using these tests because they have become somewhat of a standard when used to benchmark mpi-based infiniband testing.

The compiling for each of these tests made use of a standard user environment with available tools. Running the resulting binary was performed through whatever standard mean was available for the system in question. More specfically, where available mpicc was used for compilation and srun or mpirun were used for runing the jobs. No additional flags were passed to the compilers, so the resulting binaries are non-optimized for the system they are running on. The reason for this was that we wanted to test what the generic results would be for an uneducated user who would simply want to use all of the default options to build an executable.

1.1.1 OSU Bandwidth Test

This test sends out a fixed number of back-to-back messages and waits for a reply from the receiver. When the receiver has collected all of the associated messages, the receiver sends a reply message back to the sender. Time is measured from the moment the sender sends it's first message to the time it receives a reply back. The test utilizes non-blocking MPI functions (MPI_Isend and MPI_Irecv).

1.1.2 OSU Latency Test

This test performs a ping-pong between a sender and receiver. A message of a given size is sent out, when a receiver receives the message it sends back a reply with the same data size. Averaging a number of iterations, one-way latency numbers are obtained. The test utilizes blocking MPI functions (MPI_Send and MPI_Recv).

1.1.3 OSU Bi-Directional Bandwidth Test

This test is similar to the bandwidth test except that both nodes send out back-to-back messages and wait for a reply. This is a measurement of the maximum sustainable aggregrate bandwidth by two nodes.

[1] See URL http://nowlab.cis.ohio-state.edu/projects/mpi-iba/index.html

1.1.4 Message Injection Rate Test

Based heavily on the OSU Bandwidth Test, this test was created by PathScale to show the rate at which the interconnect can inject messages into the fabric. For clusters that utilize lots of small packets, this message injection rate can become a major factor.

1.1.5 NetPIPE Test

NetPIPE[2] is another MPI based bandwidth and latency measurement tool. While it does standardly use MPI over Infiniband (or any other high performance interconnect), NetPIPE can also utilize tcp-based connections, which will allow us to also test IP-over-IB (IPOIB) connections. NetPIPE performs a ping-pong transfer style, measuring the transmission rates.

1.1.6 Mellanox RDMA Tests

These tests come standard with the OpenIB subversion repository. They measure performance at the verbs layer rather than using higher-level protocols. In theory, this should show you what you can really expect from Infiniband, which makes a good comparison to see how well the MPI and TCP layers are performing.

1.2 Discussion on the Infiniband Technologies

Before we get too in-depth, we should probably take a moment to talk about the infiniband technologies we are looking at. As infiniband has been increasing in the market-place, most installations have been using the Mellanox-based single data-rate (SDR) option because Mellanox successfully aligned themselves as the primary source for infiniband silicon. Running mostly unoppossed, the major infiniband vendors (e.g. Voltaire, Cisco, Silverstorm) all use Mellanox chips. Recently, Mellanox released a new chip design that could perform at double data-rate (DDR) which boasts twice the performance. By design, a DDR connection (20Gb/s) will get twice the performance of an SDR connection (10Gb/s) by allowing two bits of data to be transferred per clock cycle. PathScale, on the otherhand, is attempting to move in on the market using their own newly developed SDR-based layout. Combined with the hypertransport bus, they are able to off-load as much as possible to the host system. By using the host processor (which is faster than any available embedded processor that could be installed on the infiniband card) they claim that they are able to acheive much lower latency and much higher message injection rates. For this to be effective, though, they are going to need direct access between the card and the host CPU (which is provided by the hypertransport bus) as well as well written drivers to make good use of the system.

1.3 Hardware Layout

To properly compare the different infiniband technologies, we utilized a standard cluster layout and attempted to minimize the impact that actual hardware can have on the system. Figure 1 shows the basic layout of the cluster.

The overall design that we employ for our clusters is one with a large number of nodes who has the sole-purpose of computing in parallel utilizing a high-performance interconnect to provide MPI layer communication. Each cluster has one or two management nodes that

[2]See URL http://www.scl.ameslab.gov/Projects/NetPIPE

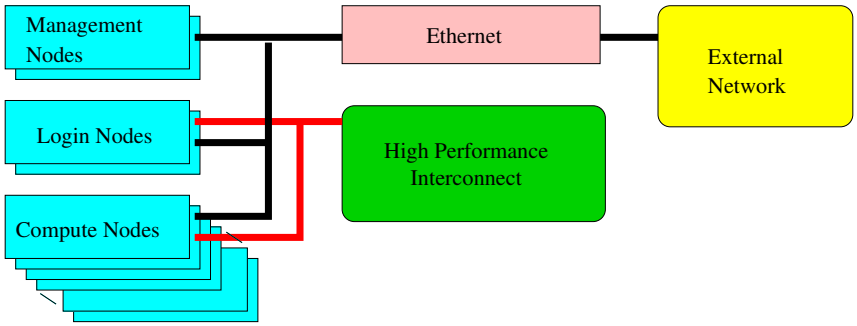

Figure 1: Generic System Layout

provide a centralized configuration point for the rest of the nodes. In addition, we use a number of dual-purpose login/routing nodes which provide a user interface into the cluster as well as a gateway for the compute nodes to speak with the rest of the computing center (e.g. to have access to the center-wide NFS home directories). Using this simple design, **ldev** was created. This table shows the layout of **ldev**:

Table 1: ldev Layout

Nodes	CPU	HCA	Slot
ldev[0-3]	dual-socket, single-core opteron	Mellanox SDR	PCI-e 8x
ldev[4-7]	quad-socket, dual-core opteron	Mellanox DDR	PCI-e 8x
ldev[8-11]	dual-socket, single-core opteron	Infinipath SDR	HTX

1.4 Operating System Software

Throughout this document you will probably see many references to "chaos", specifically pertaining to the operating system running on our clusters. Chaos is a derivative of the linux operating system produced by RedHat, and follows closely to what RedHat delivers. Chaos 3.1 (which is the latest version, but still in beta as of this writing) is based on RHEL4-U3. Strictly speaking, chaos is a stripped down version of RedHat's operating system layered with extra cluster-related tools that are homegrown. More information on Chaos can be found on the Linux@Livermore[3] website.

As far as the infiniband stack that is being used for this testing, we are focusing on using the OpenIB[4] software stack backported to RedHat's 2.6.9 kernel with mvapich-gen2 (as provided by the OpenIB stack). To properly compare the Infinipath hardware, though, we will also be including testing done with PathScale's proprietary ipath stack (for this testing it will be running on CentOS-4.2). Table 2 shows the software stack versions.

Table 2: Infiniband Software Versions

Nodes	Software Stack	Kernel	IB Version
ldev[0-7]	Chaos with OpenIB	2.6.9-40chaos	svn6829
ldev[8-11]	CentOS with ipath	2.6.9-22.0.2.ELsmp	1.2-7856.1303_fc3_psc

[3]See URL http://www.llnl.gov/linux
[4]See URL http://www.openib.org

For part of the testing, we will also be seeing how the Infinipath handles with openib, so ldev[8-11] will be rebooted onto the same stack as the rest of the cluster. Also, throughout this testing, the subnet manager will be running via the native subnet manager running inside of the switch.

1.5 Key Descriptions

This would probably be a good time to give a little help to what the information in the keys of the various graphs actually stands for. Table 3 shows the basic breakdown of all of the keys used throughout this report:

Table 3: Key Descriptions

Key	Nodes	Software Stack	HCA
SDR	ldev[0-3]	OpenIB	Mellanox SDR
DDR	ldev[4-7]	OpenIB	Mellanox DDR
openib-pathscale	ldev[8-11]	OpenIB	Infinipath
pathscale	ldev[8-11]	Ipath	Infinipath

2 TCP Testing

Since many vendors approach infiniband as the one-wire solution to clustering (you can use the infiniband not only for MPI, but for everything else), we're going to need to know what we can expect from this solution. Therefore, we'll start our teseting at the TCP level so that we can judge what kind of throughput to expect from the various infiniband flavors. Figures 2 and 3 shows us just that.

Now, there are a few things to notice on these graphs. The first is that with TCP-based protocols, the best that you are (currently) going to see is around 2.6Gb/s (or 325MB/s) over this interconnect. While this is better than gigabit-ethernet, this is far from the 10Gb/s that infiniband should be capable of. The second piece to notice is that (utilizing the ipath IB stack), the Infinipath performed horribly peaking at about 89Mb/s (which is equivalent to measured performance of 100Mb/s based ethernet). The last piece to notice is that SDR and DDR both peaked about the same, but DDR saw slightly better performance between 1KB and 100KB packet sizes. As for latency, on the openib stack all of the tests were fairly equivalent, but again the ipath stack had extremely poor performance.

3 MPI Testing

In high-performance computing, MPI performance can make or break the system. Not only must we focus on throughput (or bandwidth), but we also need to pay great attention to the latency, bi-directional bandwidth as well as the message injection rates. The following sections will start showing us how things panned out.

3.1 MPI Bandwidth

First, we'll look at the Bandwidth seen by the system. For this, we will use two sets of tests, the OSU test as well as the MPI tests for NetPIPE. Figures 4 and 5 shows the results from these tests.

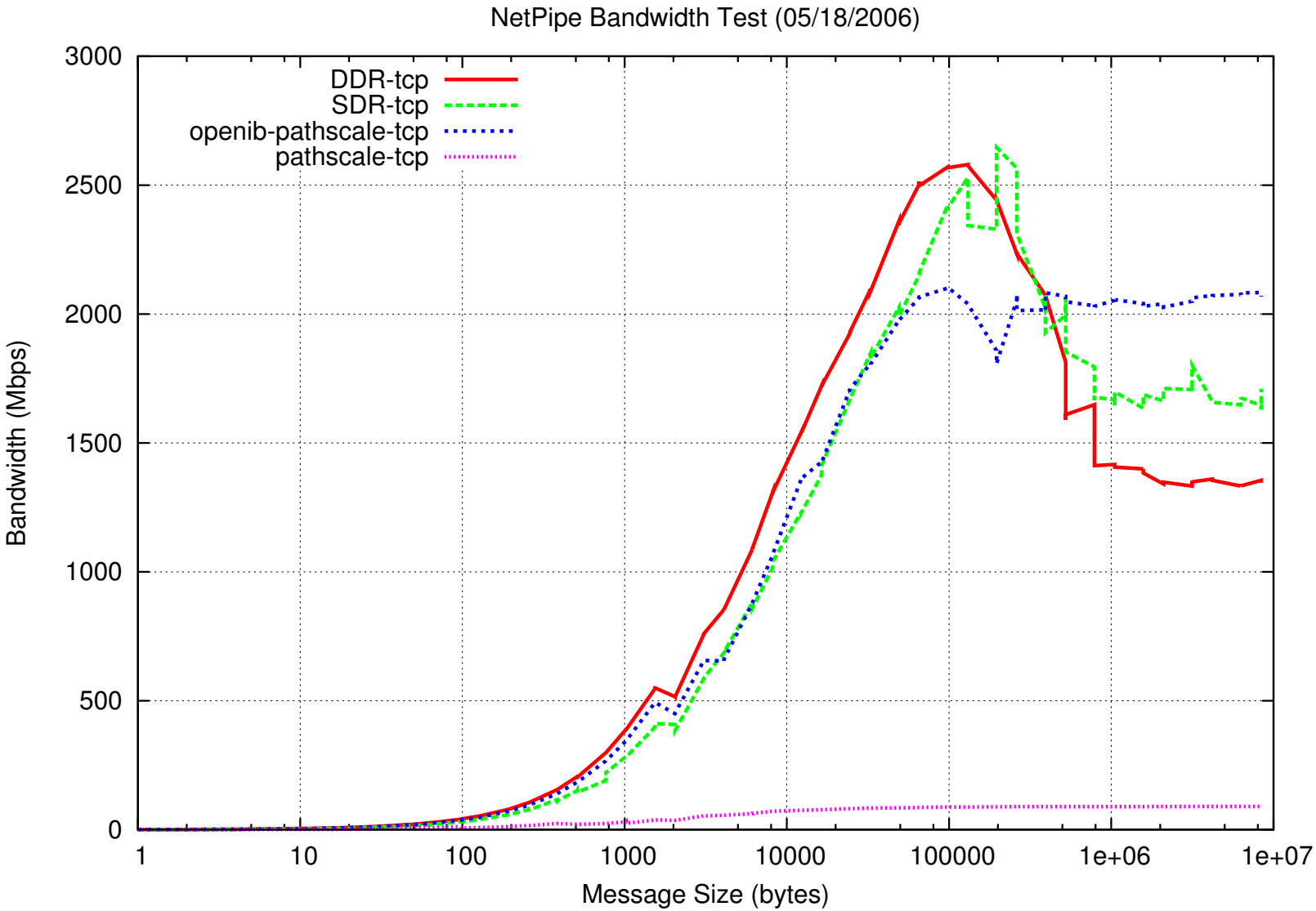

Figure 2: TCP Bandwidth Performance

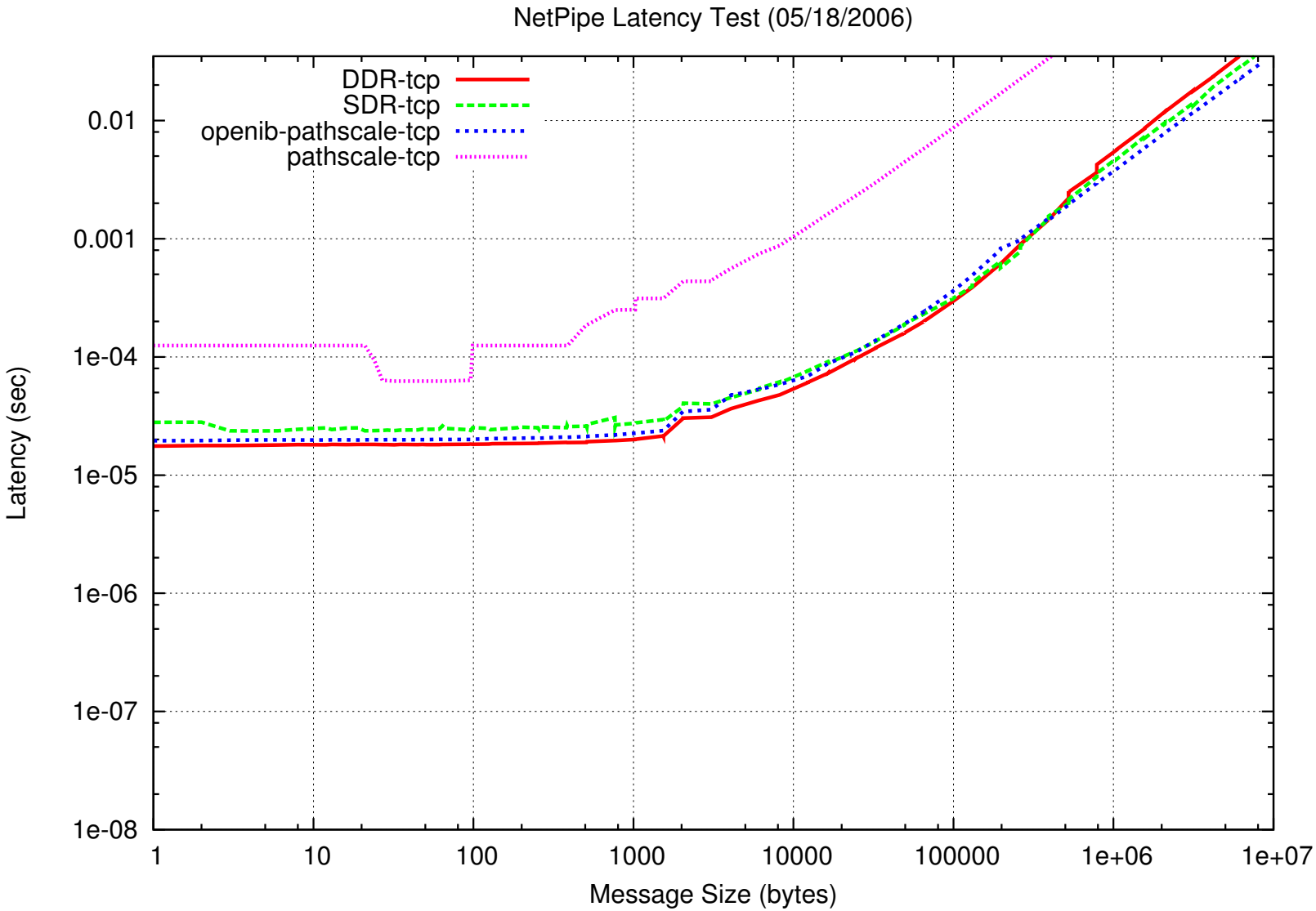

Figure 3: TCP Latency Performance

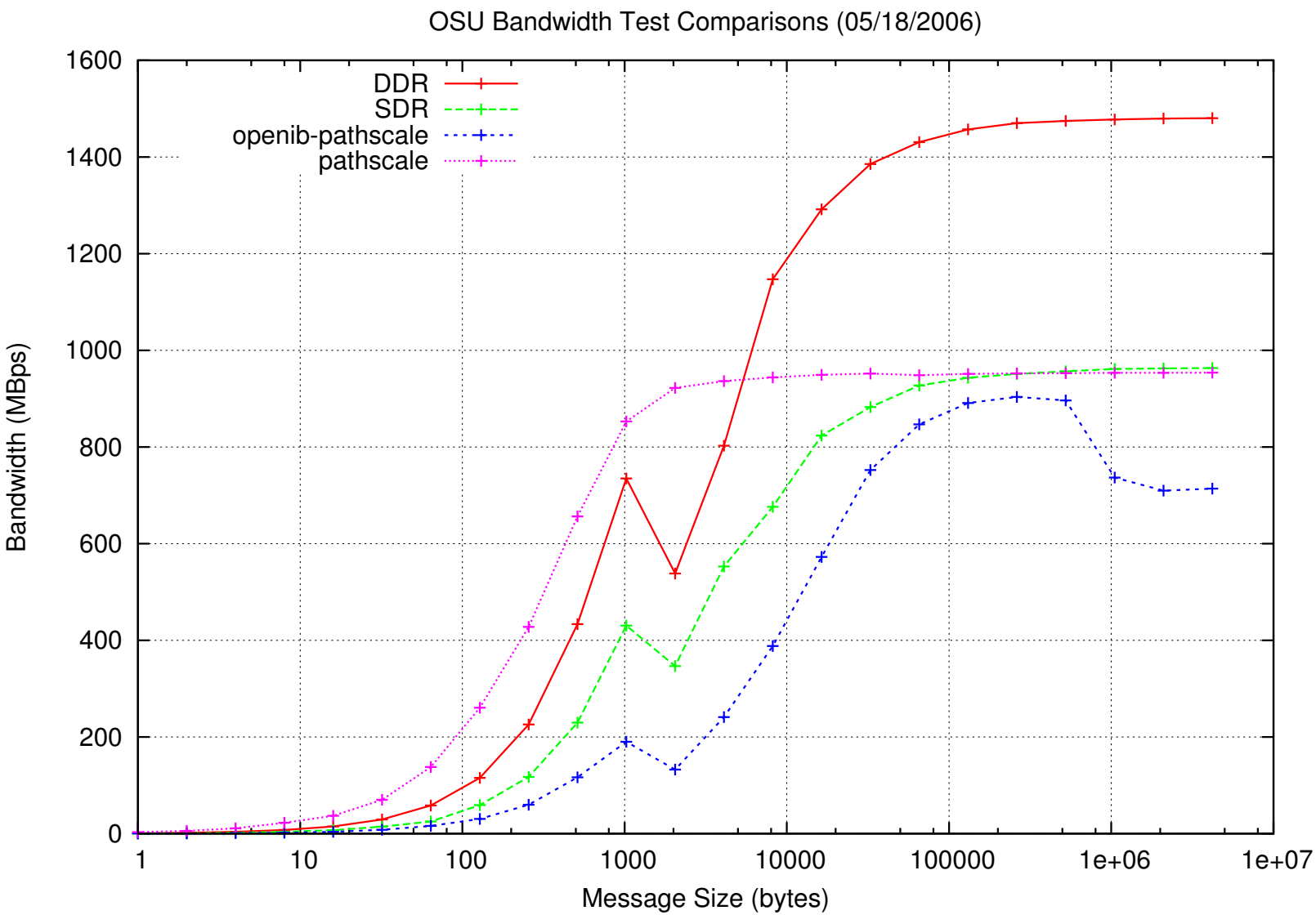

Figure 4: OSU MPI Bandwidth Performance

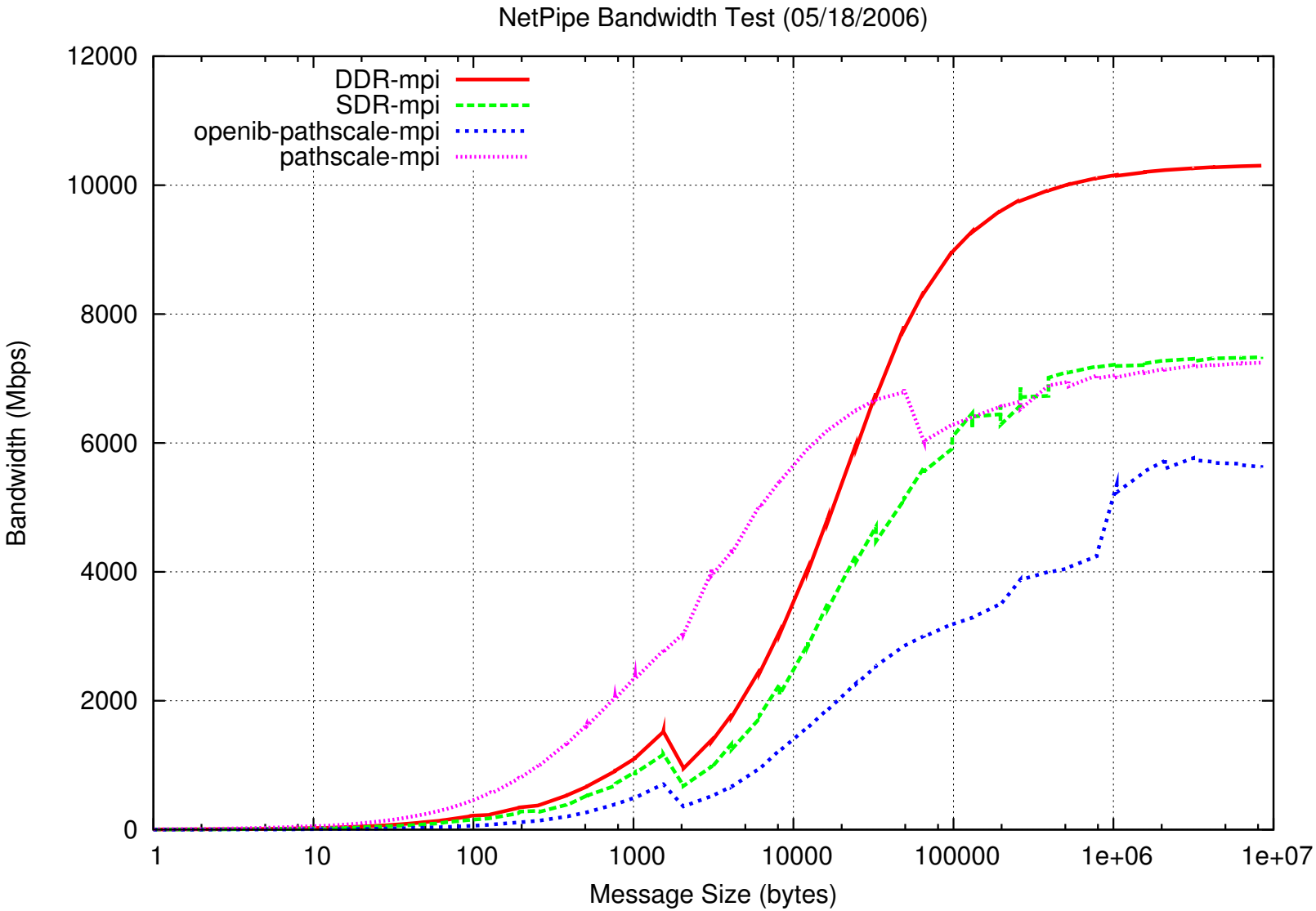

Figure 5: NetPIPE MPI Bandwidth Performance

From both graphs, we can clearly see that DDR peaks much higher than anything else, but the ipath IB stack reaches max bandwidth at lower packet sizes. Running the Infinipath hardware on openib, though, we start to see a much different picture. With a slow rise to a much lower peak, it is clear to see that a lot more development needs to be done on the drivers in openib.

3.2 MPI Latency

Again, we'll use results from OSU as well as NetPIPE for this test. Figures 6 and 7 shows the results.

These graphs clearly show where ipath can shine. At lower packet sizes, the ipath latency numbers are significantly lower than anything else, until you start hitting around 100KB where DDR starts looking better. Again, we can see that the openib port of the Infinipath drivers fall extremely short of their goal.

3.3 MPI Bi-Directional Bandwidth Performance

Next up is the bi-directional MPI performance. Figure 8 shows the results.

There are a few issues that can be seen with this graph. The first is again the poor performance of the openib stack on the Infinipath hardware (so poor, in fact, that the test was never able to complete). Second is the quicker rise to peak that the ipath stack achieves. Third, there seems to be an anomaly with DDR not achieving better performance than it did. This can probably be attributed to something in the MPI layer not utilizing the DDR rates correctly (verbs based testing should show us whether or not this is true).

3.4 Message Injection Rates

The last test for MPI comes in the form of the message injection rate. Figure 9 shows the results.

Included on this graph is a bit of scaling to utilize all of the processes available on the system. The term "ppn" stands for proccess per node. For each system, you can see a reasonable increase for each additional process. For comparable ppn's, you can see that the ipath stack does significantly better than the DDR and SDR counterparts. Again, we can see the extremely poor performance of the openib stack on the Infinipath hardware. Due to a lack of HTX slots on a quad-socket system at the time of this writing, we're unable to see how well the ipath scales to match the DDR system.

4 Verbs Layer Testing

Lastly, we're going to take a look at the verbs layer perforance of these cards. At this level, we should see what we can expect from the performance if the MPI and TCP layers were implemented better. At this time, the ipath drivers do not include access to the verbs layer, so only openib testing could be done for these tests. There are, again, three basic tests performed: bandwidth, latency and bi-directional bandwidth. In each of these tests, we will be focussing on the actual RDMA write transactions between the two nodes.

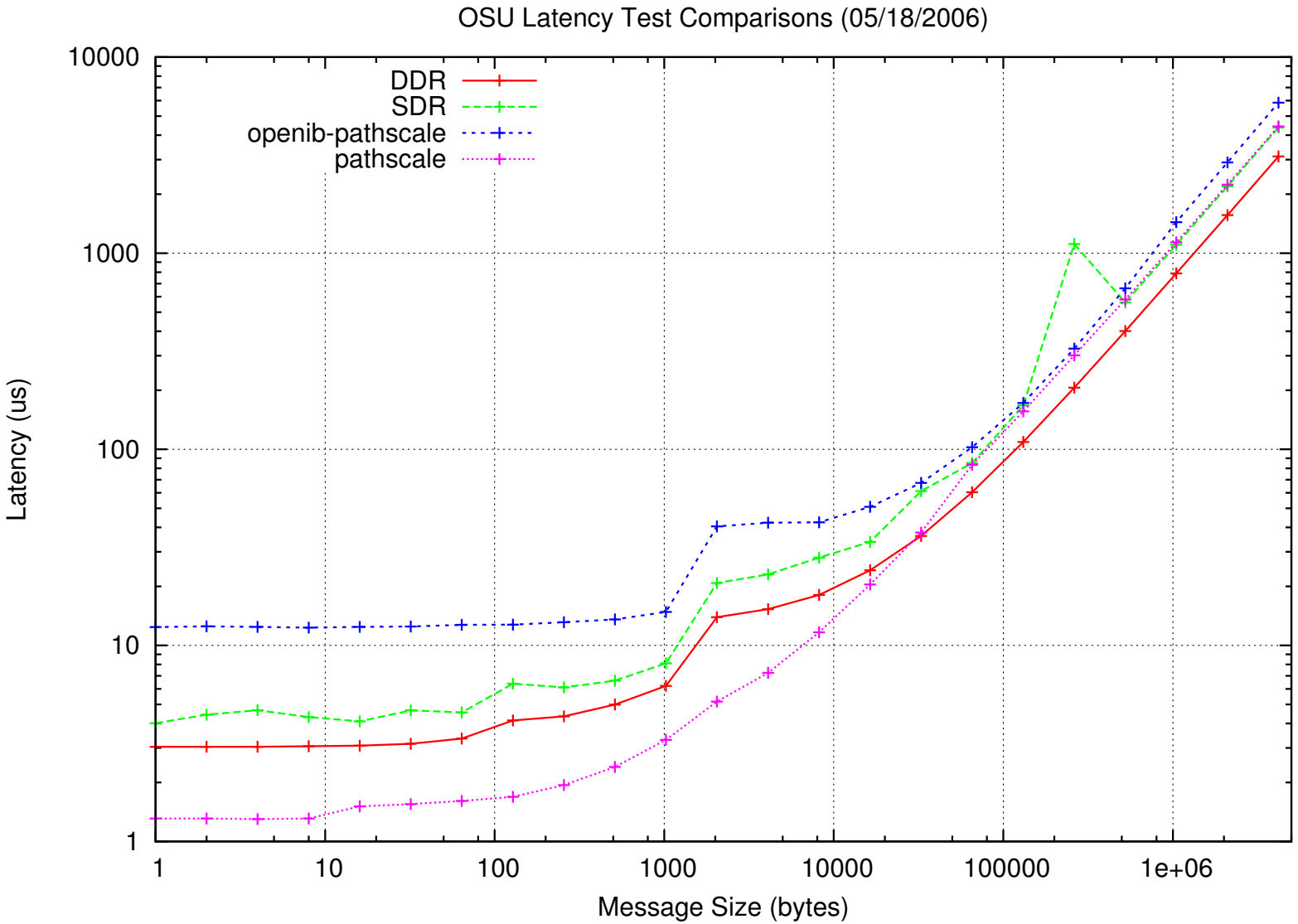

Figure 6: OSU MPI Latency Performance

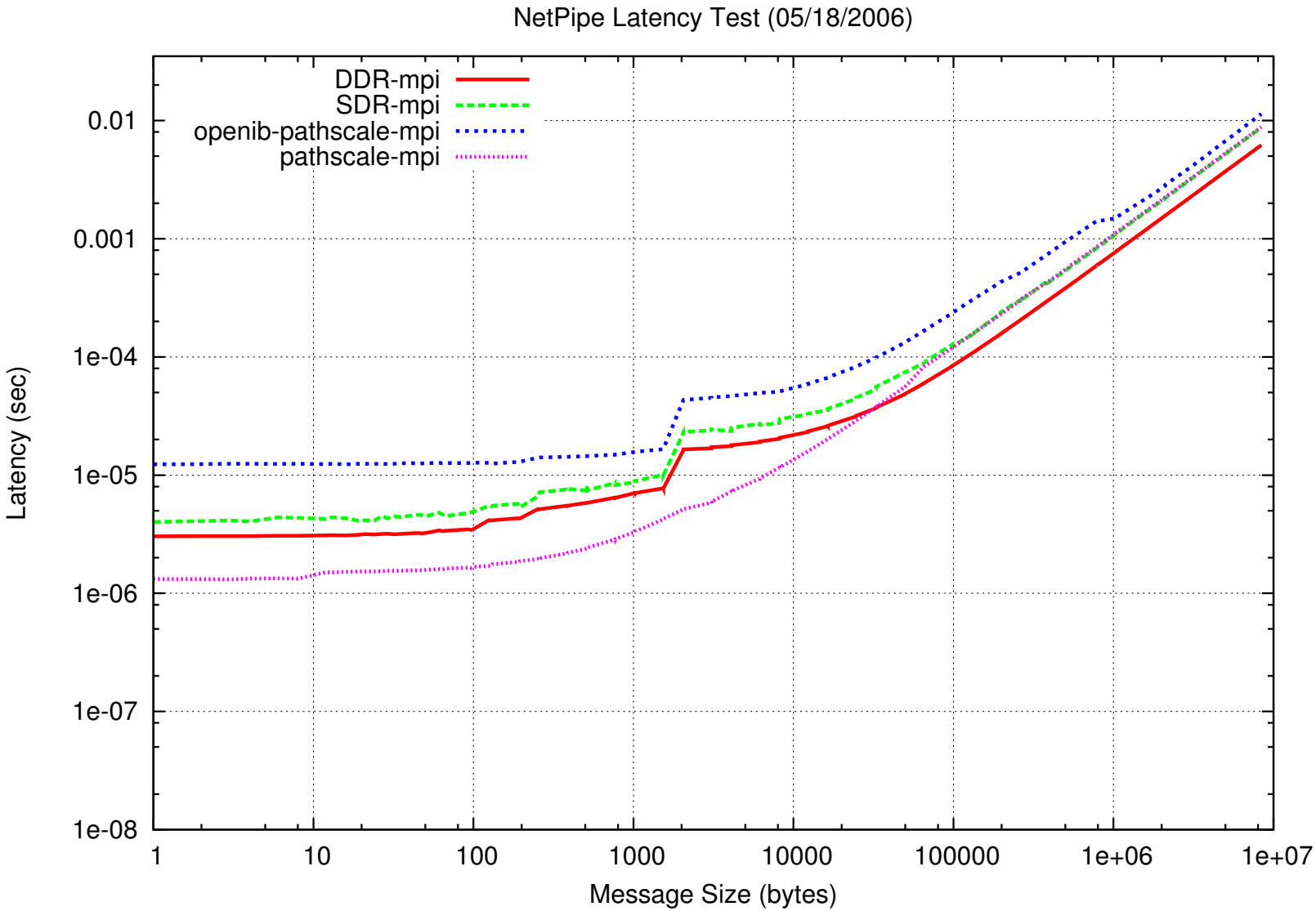

Figure 7: NetPIPE MPI Latency Performance

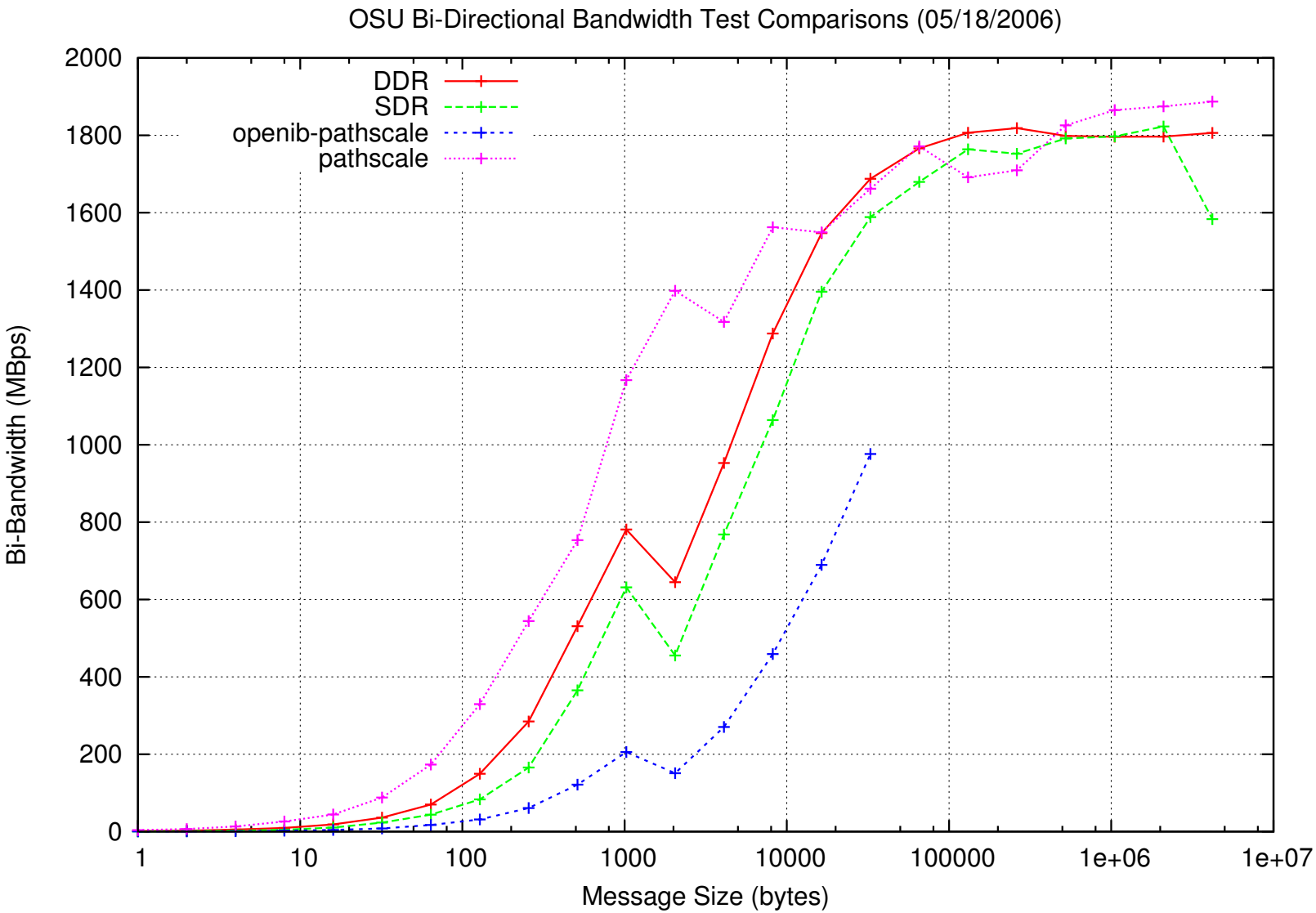

Figure 8: MPI Bi-Directional Bandwidth Performance

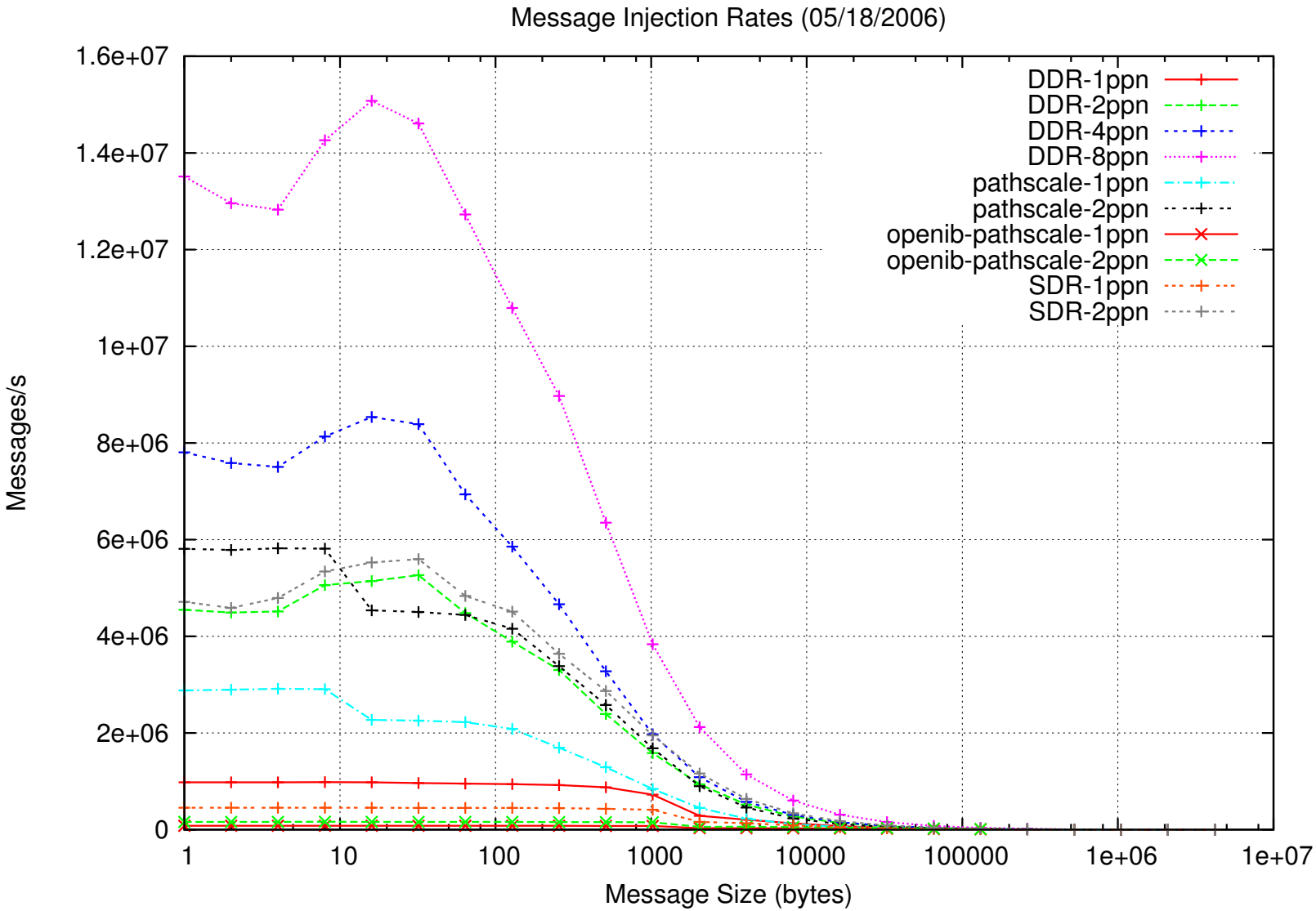

Figure 9: MPI Message Injection Rates

4.1 Verbs Bandwidth Performance

From figure 10, looking at straight node-to-node bandwidth performance we can see the large increase that DDR has over SDR.

4.2 Verbs Latency Performance

From figure 11, we can see that DDR's has much better latency performance than SDR, as expected.

4.3 Verbs Bi-Directional Bandwidth Performance

Figure 12 shows what we should expect from DDR compared to SDR. This shows pretty clearly that there is something that needs to be done in the MPI layer to improve the performance of DDR. Also of note is the fact that (for when it was working) the openib test on Infinipath was doing better than the SDR results.

5 Conclusions

Table 4 shows a quick comparison of RDMA, MPI and TCP between the different infiniband technologies while table 5 shows the N/2 bandwidth for each technology.

Table 4: Peak Performance Comparisons

Bandwidth			
	RDMA	MPI	TCP
DDR	1413MB/s	1480MB/s	329MB/s
SDR	936MB/s	963MB/s	320MB/s
Openib-Pathscale	923MB/s	903MB/s	263MB/s
Pathscale	N/A	953MB/s	11MB/s
Latency			
DDR	2.6us	3us	17.6us
SDR	3.5us	3.95us	27.9us
Openib-Pathscale	12.1us	12.44us	19.5us
Pathscale	N/A	1.29us	124.9us
Bi-Directional Bandwidth			
DDR	2761MB/s	1818MB/s	N/A
SDR	1714MB/s	1823MB/s	N/A
Openib-Pathscale	1847MB/s	976MB/s	N/A
Pathscale	N/A	1887MB/s	N/A

Table 5: N/2 Comparisons

	N/2 Bandwidth	Packet Size
DDR	637MB/s	800B
SDR	483MB/s	1398B
Openib-Pathscale	N/A	N/A
Pathscale	475MB/s	330B

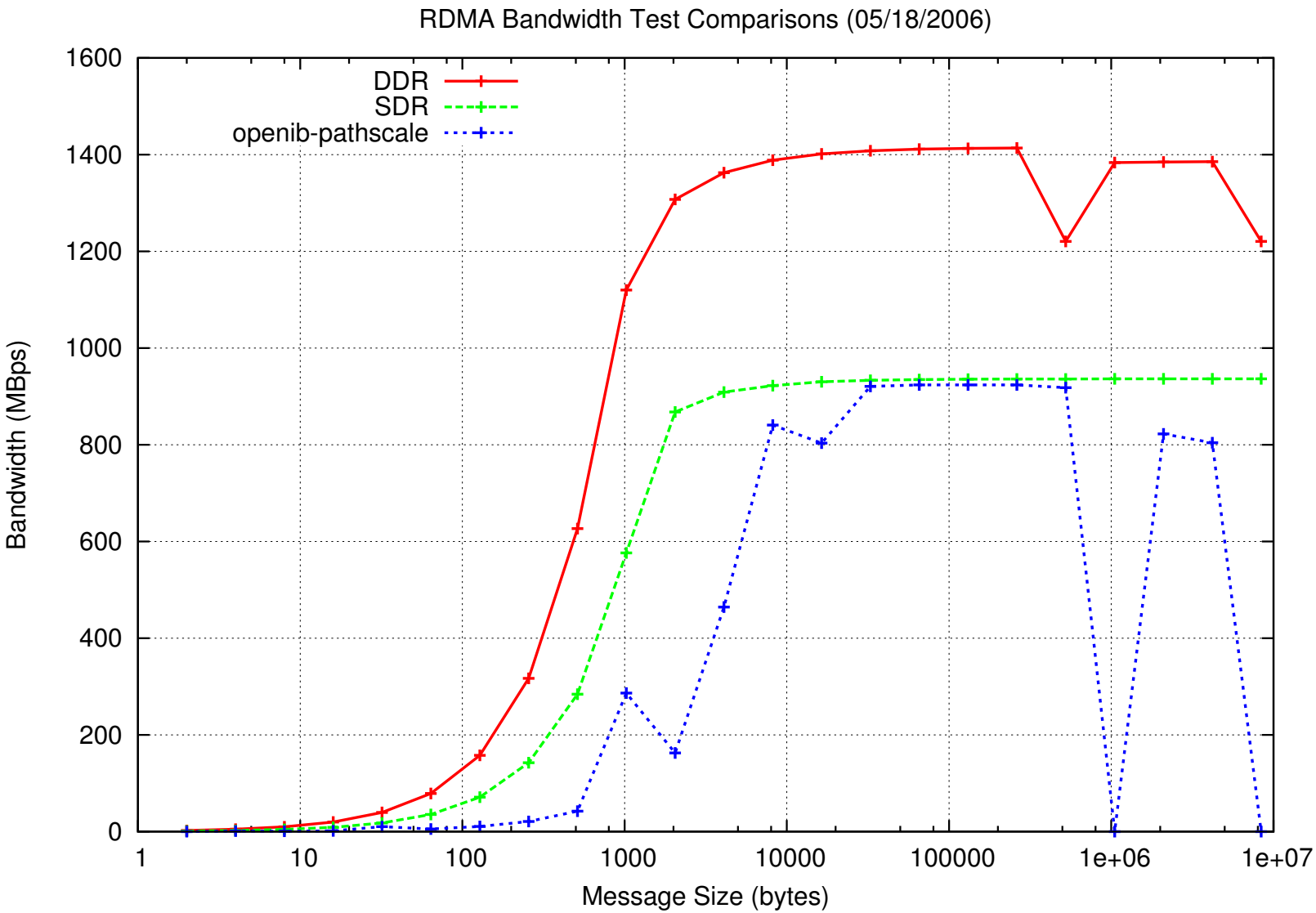

Figure 10: Verbs Bandwidth Performance

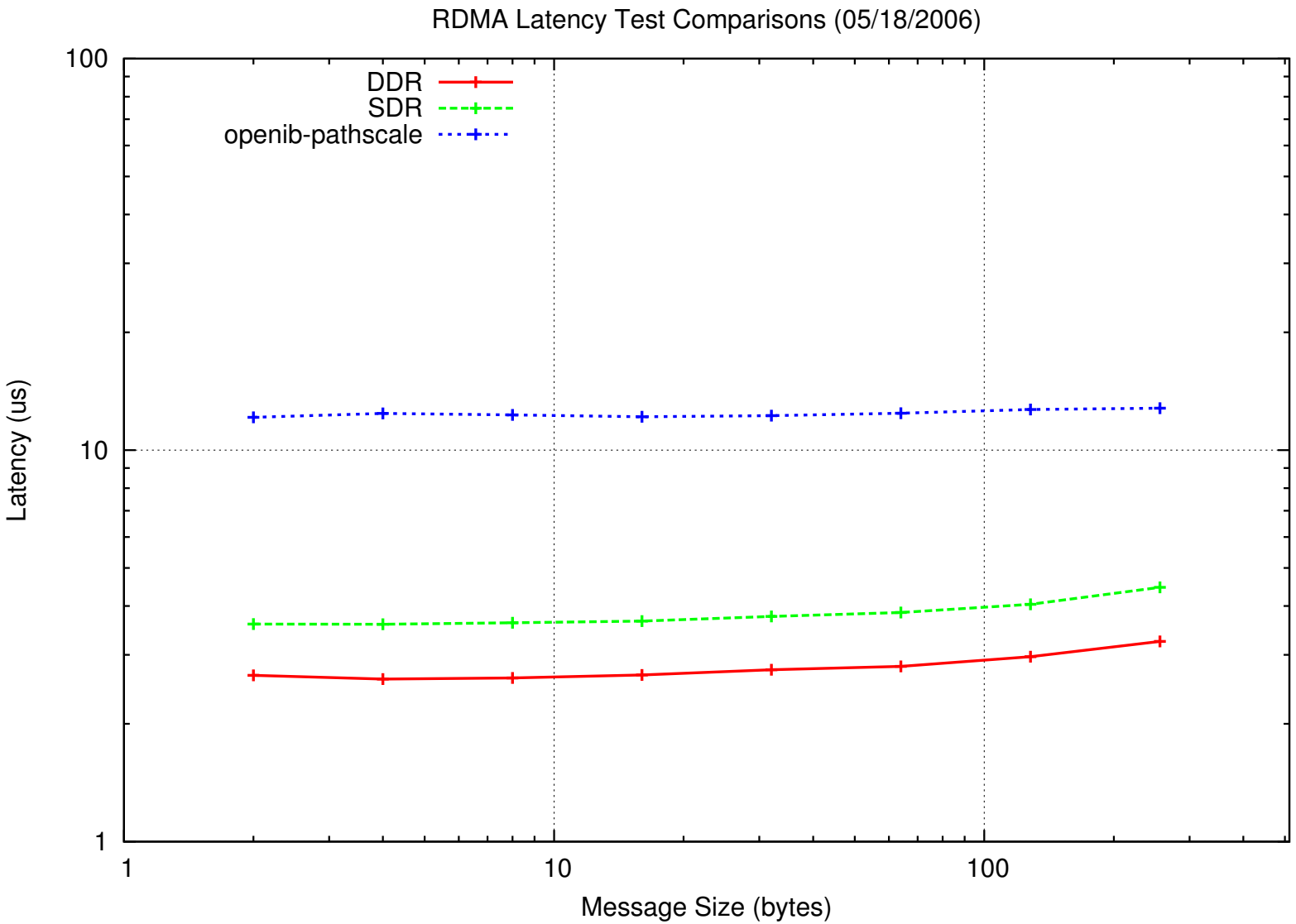

Figure 11: Verbs Latency Performance

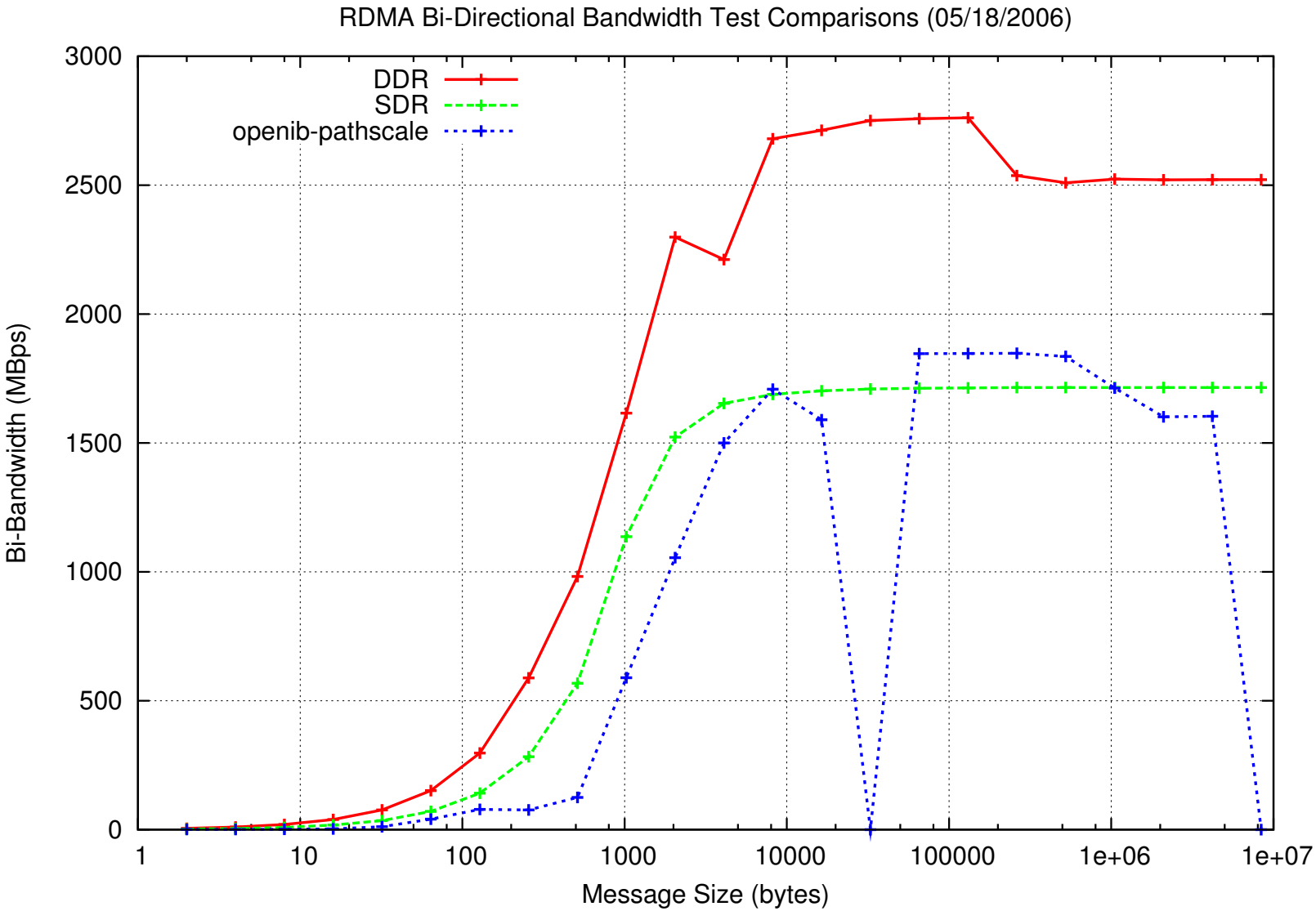

Figure 12: Verbs Bi-Directional Bandwidth Performance

Using the information here, we can see that paying attention to straight peak bandwidth (both uni- and bi-directional) DDR is clearly the winner. Factoring in latency as well as message passing rates, though, we start seeing the added benefit of Infinipath with the ipath software. The Infinipath is only strenghtened if your computational workload generally transmits smaller packet sizes, as the Infinipath excels in this area. The biggest draw-back to the Infinipath hardware is the lack of useful openib support for the RedHat based kernels as well as it's miserable TCP support.