PERFORMANCE BENCHMARKS

Chelsio 10G Ethernet Open MPI OFED iWARP with Juniper Switch

Chelsio Communications • www.chelsio.com • sales@chelsio.com • +1-408-962-3600

Executive Summary

Ethernet provides a reliable and ubiquitous networking protocol for high-performance computing (HPC) environments. When used in conjunction with the Internet Wide Area RDMA Protocol (iWARP), Ethernet delivers an ultralow latency, high performance, and highly scalable interconnect solution for application layer environments ranging from dozens to thousands of compute nodes. In tests conducted at the Chelsio facility, results demonstrate successful interoperability between Chelsio's latest SFP+ 10GbE adapter and Juniper's 10GbE SFP+ switch. RDMA iWARP MPI performance results show that the application can sustain line rate with approximately 4us latency.

What is iWARP?

iWARP, also called RDMA over Ethernet, is a low latency solution for supporting high-performance computing over TCP/IP. Developed by the Internet Engineering Task Force (IETF) and supported by the industry's leading 10GbE Ethernet adapters, iWARP works with existing Ethernet switches and routers to deliver low latency fabric technologies for high-performance data centers.

In addition to providing all of the total cost of ownership (TCO) benefits of Ethernet, iWARP delivers several distinct advantages for use with Ethernet in HPC environments:

- It is a multivendor solution that works with legacy switches.
- It is an established IETF standard.
- It is built on top of IP, making it routable and scalable from just a few to thousands of collocated or geographically dispersed endpoints.
- It is built on top of TCP, making it highly reliable.
- It allows RDMA and MPI applications to be ported from InfiniBand (IB) interconnect to IP/Ethernet interconnect in a seamless fashion.

Chelsio's iWARP and TCP Offload Engine Solutions

Chelsio's T420-LL-CR 10GbE iWARP adapters improve HPC application performance by leveraging an embedded TCP Offload Engine (TOE), a technology that offloads TCP/IP stack processing from the host to the NIC. A TOE frees up server memory, bandwidth, and valuable CPU cycles to improve the performance of applications that are sensitive to these parameters. When used with higher speed interfaces such as 10GbE, the performance improvement enabled by a TOE is even more dramatic, since 10GbE delivers data rates that are so high, host-based TCP instruction execution can quickly overwhelm even the fastest servers.

With the rapid adoption of 10GbE, and the resulting increase in data flow into and out of multi-core servers, TOEs have become a requirement to deliver the high throughput and low latency needed for HPC applications, while leveraging Ethernet's ubiquity, scalability, and cost-effectiveness.

Test Configuration

Test tools

• Test tool: Intel® MPI Benchmark Suite V3.2

Host Systems

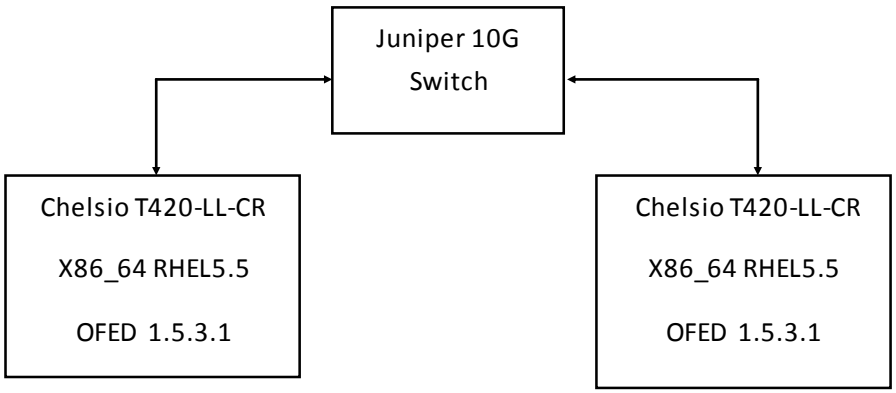
• CPU: 1 x quad-core Intel Core® i7-2600K @ 3.4 GHz
• Root Complex: Intel Sandy Bridge Chipset
• Memory: 8GB
• OS: Linux 2.6.18-194.el5
• OFED 1.5.3.1
• x86_64
• Open MPI 1.4.3

Chelsio T420-LL-CR with Juniper Switch.

One way, 1 byte Latency: 4.6 us

Unidirectional Bandwidth: 1111.65 MB/s

Bidirectional Bandwidth: 2224.11 MB/s



Tests Performed

· **PingPing:** Measures the startup (latency) and throughput of a single message sent between two processes, with each node issuing the commands concurrently.

· **Sendrecv:** Creates a chain of sending and receiving between upstream and downstream neighbors.

· **Exchange:** Creates a chain of sending and receiving between upstream and downstream neighbors, with each node issuing the send/receive commands concurrently.

· **Allreduce:** Measures the time to perform the allreduce collective operator where each process in a group owns a vector whose elements are combined with the corresponding elements owned by other processes, producing a new vector of the same length.

· **Reduce:** Measures the time to perform the allreduce collective operator where each process in a group owns a vector whose elements are combined with the corresponding elements owned by other processes, producing a new vector of the same length, changing the root of the process cyclically.

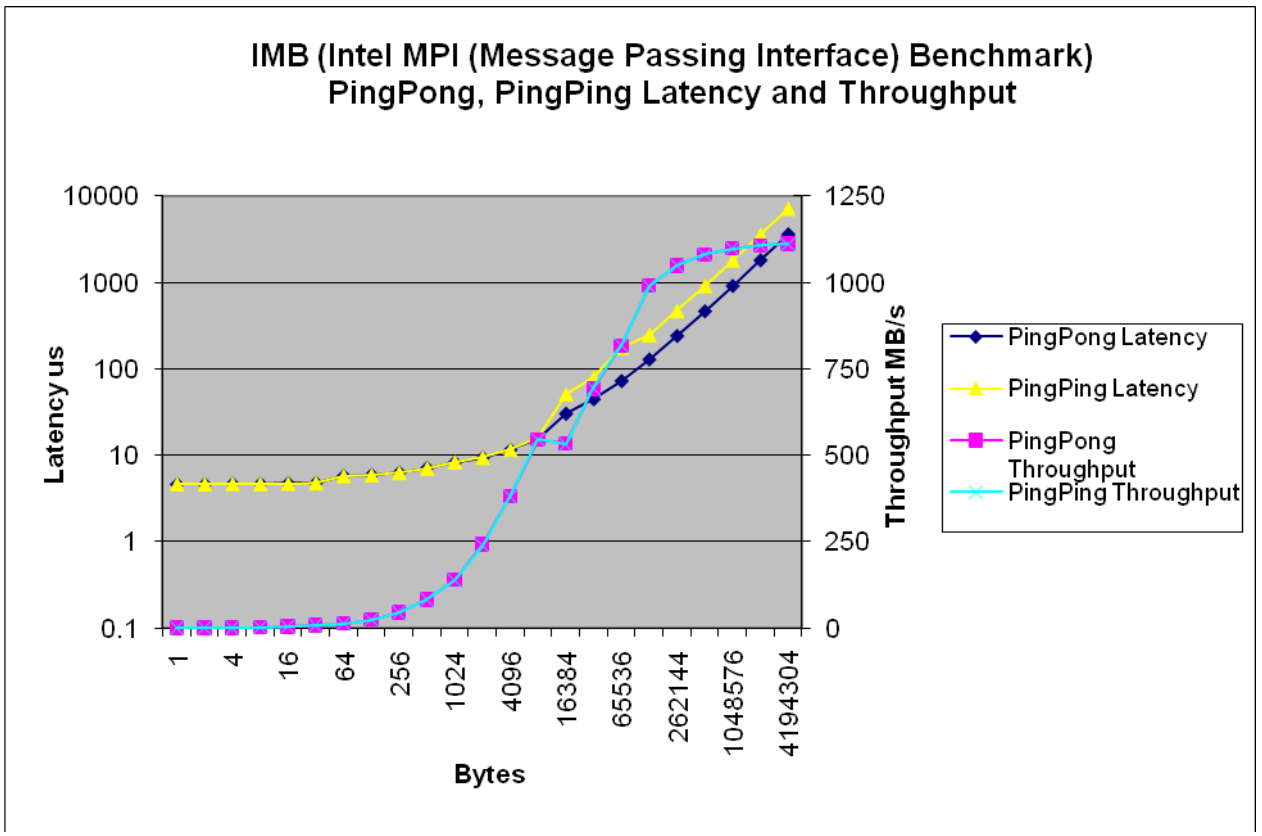
· **Reduce_scatter:** Measures the time to perform the allreduce collective operator, split between all processes, where each process in a group owns a vector whose elements are combined with the corresponding elements owned by other processes, producing a new vector of the same length.

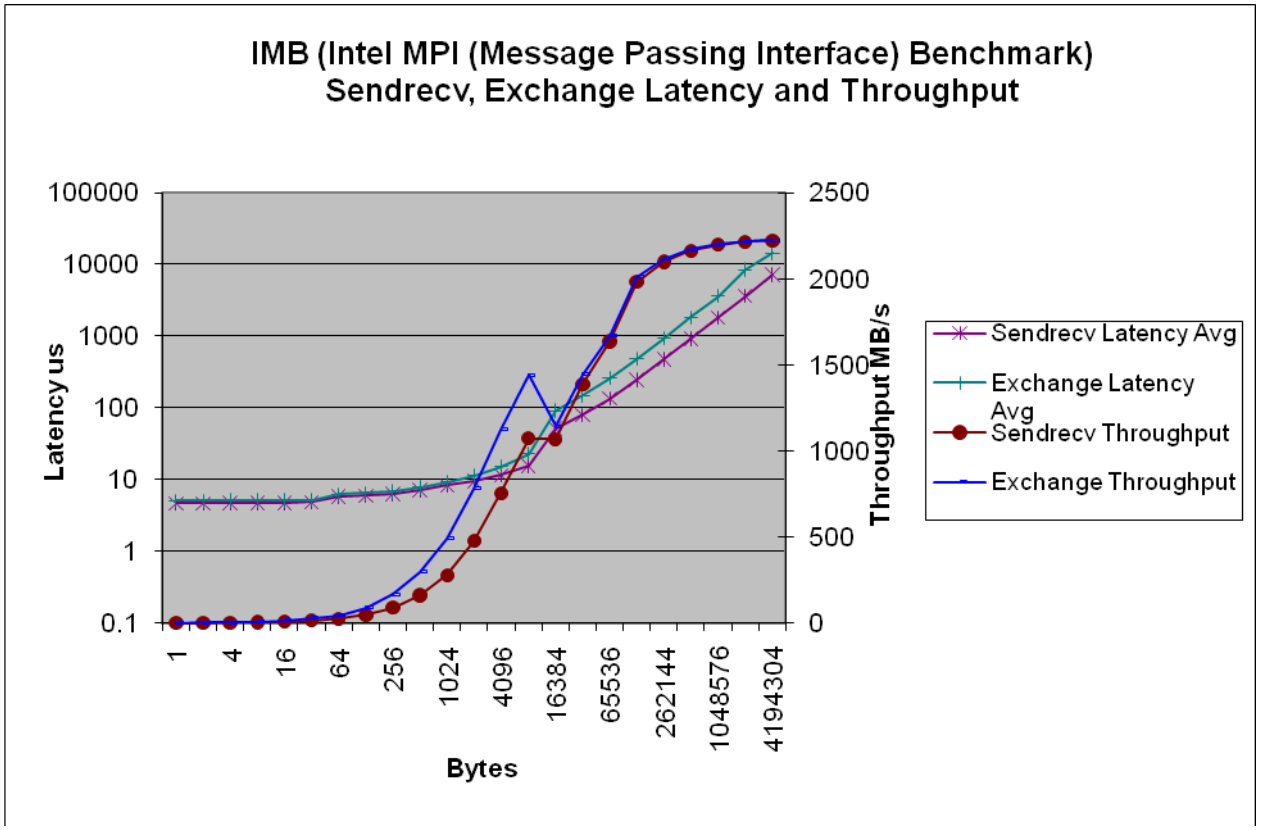
· **Allgather:** Measures the time that it sends X bytes and gathers X*processes bytes.

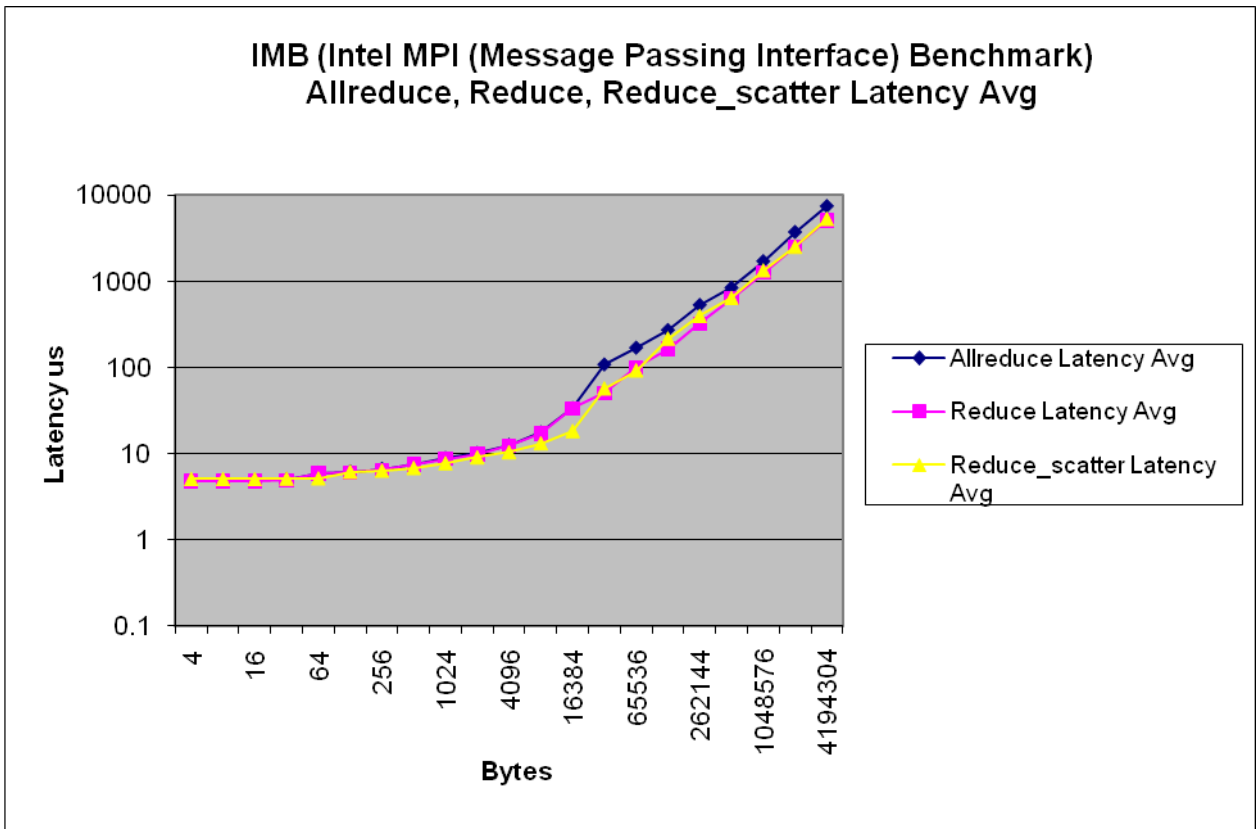
· **Allgatherv:** Measures the time that it sends X bytes and gathers X*(#processes) bytes, and sees if MPI produces more overhead due to the more complicated situation.

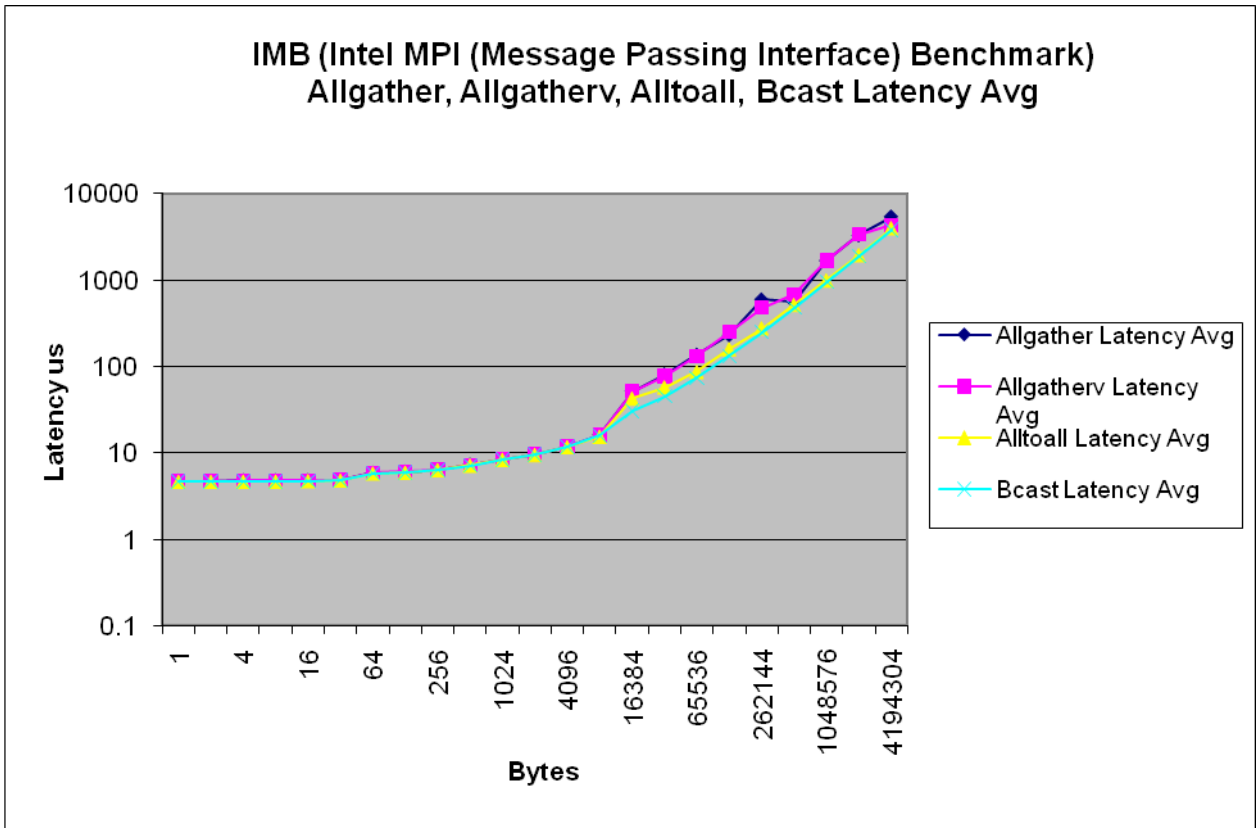
· **Alltoall:** Measures the time that it takes for every process to input X*(#processes) bytes (X for each process) and receives X*(#processes) bytes (X for each process).

· **Bcast:** Measures the time that it takes a root process to broadcast X bytes to all while the root of the operation is changed cyclically.


IMB (Intel MPI (Message Passing Interface) Benchmark)
PingPong, PingPing Latency and Throughput
Latency us
Throughput MB/s
Bytes
10000
1000
100
10
1
0.1
1250
1000
750
500
250
0
1
4
16
64
256
1024
4096
16384
65536
262144
1048576
4194304
PingPong Latency
PingPing Latency
PingPong Throughput
PingPing Throughput


IMB (Intel MPI (Message Passing Interface) Benchmark)
Sendrecv, Exchange Latency and Throughput
Latency us
Throughput MB/s
Bytes
100000
10000
1000
100
10
1
0.1
2500
2000
1500
1000
500
0
1
4
16
64
256
1024
4096
16384
65536
262144
1048576
4194304
Sendrecv Latency Avg
Exchange Latency Avg
Sendrecv Throughput
Exchange Throughput


IMB (Intel MPI (Message Passing Interface) Benchmark)
Allreduce, Reduce, Reduce_scatter Latency Avg
Latency us
10000
1000
100
10
1
0.1
4
16
64
256
1024
4096
16384
65536
262144
1048576
4194304
Bytes
Allreduce Latency Avg
Reduce Latency Avg
Reduce_scatter Latency Avg


IMB (Intel MPI (Message Passing Interface) Benchmark)
Allgather, Allgatherv, Alltoall, Bcast Latency Avg
Latency us
10000
1000
100
10
1
0.1
1
4
16
64
256
1024
4096
16384
65536
262144
1048576
4194304
Bytes
Allgather Latency Avg
Allgatherv Latency Avg
Alltoall Latency Avg
Bcast Latency Avg