Filesystem Performance Evaluation of Obsidian Longbow Routers

Xiangyong Ouyang, Hari Subramoni, Mark Arnold, Dhabaleswar K. Panda

{ouyangx, subramon, arnoldm, panda}@cse.ohio-state.edu

Department of Computer Science and Engineering

The Ohio State University

3/24/2011

1 Introduction

High performance interconnects such as InfiniBand have been widely deployed in HPC systems. Recently, long-haul InfiniBand technologies have enabled applications running on an InfiniBand cluster to reach its counterpart over a wide distance. Obsidian Longbow router implements such a technique to support IB-WAN routing. It allows an InfiniBand fabric to be extended via optical fiber over varying distances, such that two distant InfiniBand clusters can be seamlessly interconnected with their native InfiniBand infrastructures. This InfiniBand router also provides hardware encryption mechanisms to ensure data transfer security and its ports can be configured to inject a certain amount of latency to emulate the behavior of a real network over a geographically wide area.

2 Experiment Testbed Overview

In this work we conducted experiments to evaluate the actual performance of Obsidian Longbow Routing on state-of-the-art InfiniBand clusters. We used Obsidian Longbow routers to connect two InfiniBand compute clusters, varied the routing ports latency to emulate the effects of different physical distances between the two clusters, and evaluated the performance of Obsidian Longbow InfiniBand router under different workloads, either with or without the hardware encryption feature enabled.

3 Experimental Results

In the experiments, we ran two benchmarks with filesystem related workload to measure Obsidian Longbow router performance:

- "IOzone" benchmark to measure data movement performance between compute and storage cluster.
- "Dsync" benchmark to measure data movement performance between two clusters.

3.1 IOzone Benchmark

In this testing, we interconnected two InfiniBand compute clusters with two Obsidian Longbow routers, as illustrated in Figure 1. Details of the experiment settings were as follows:

- All compute nodes in the two clusters were equipped with InfiniBand DDR (Double Data Rate) interface cards with network bandwidth=16 Gbit/s.
- Luster 1.8.3 was installed on the two clusters. In cluster1, one node acted as the single Metadata Server and four nodes as Lustre Object Storage Servers. The other cluster, cluster2, contained four Lustre client nodes which mounted Lustre filesystem in their local mount point.
- The IOzone benchmark was run on the four client nodes to read/write files stored in the Lustre servers.
- Each IOzone thread read/wrote 5GB file in 1MB record size.
- We varied IOzone thread number from 1 to 32, and all threads were evenly distributed across the 4 client nodes.

- We varied the port latency from 0 to 1000 us, which led to RTT from 0 to 2000 us. In each latency setting we either enabled or disabled hardware encryption option.

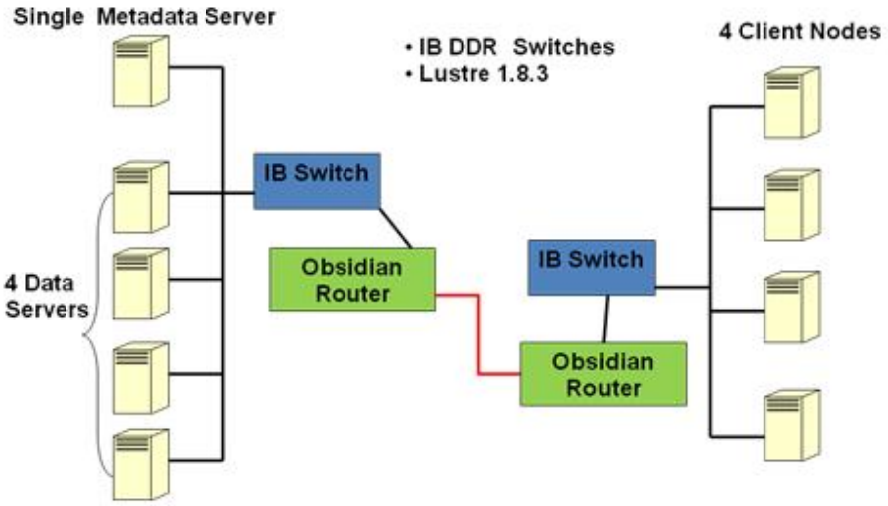

Figure 1: Lustre + IOzone testing

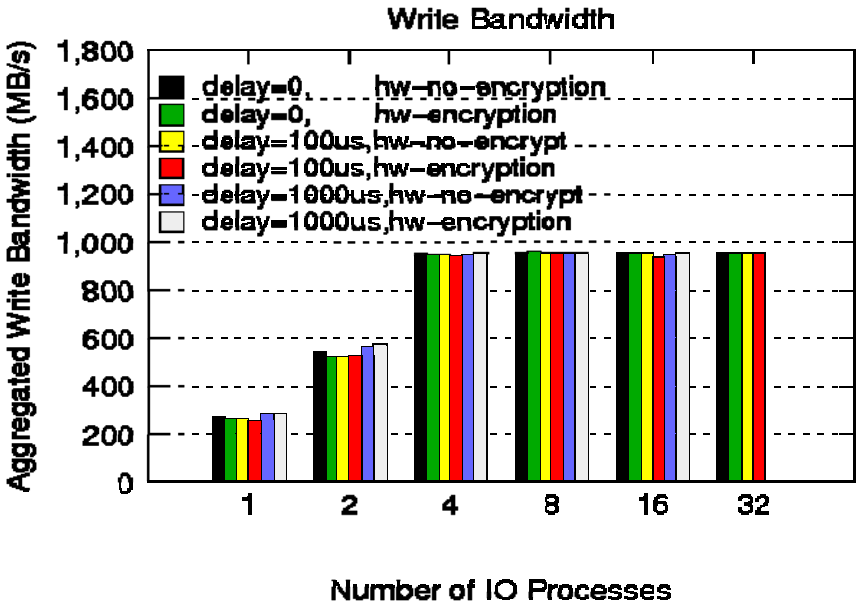

Figure 2: IOzone Write Bandwidth

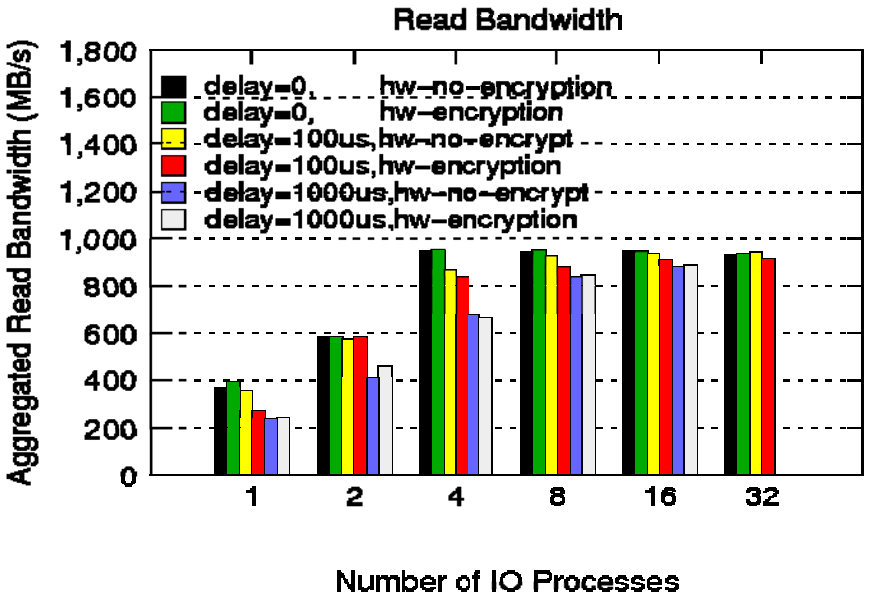

Figure 3: IOzone Read Bandwidth

Figure 2 depicts the aggregated write bandwidth of IOzone. We found that, for a given number of IOzone threads, the total write bandwidth did not vary significantly with different latency. The bandwidth was not sensitive to hardware encryption, implying that hardware encryption did not cause noticeable overhead. The total write bandwidth flattened at four IOzone threads, giving a total write bandwidth of around 900MB/s.

Similar trends were observed in Figure 3. However, we obtained lower read bandwidth as the latency was increased. This degradation became less obvious as we increased IOzone thread numbers.

3.2 Dsync Benchmark

Dsync is a remote directory synchronization tool developed for wide area InfiniBand applications. It is able to transfer file data from a remote compute node to local node to synchronize the two heterogeneous filesystems on two remote sites. Internally, it takes advantages of InfiniBand RDMA verbs to transfer large bulks of data to achieve high throughput.

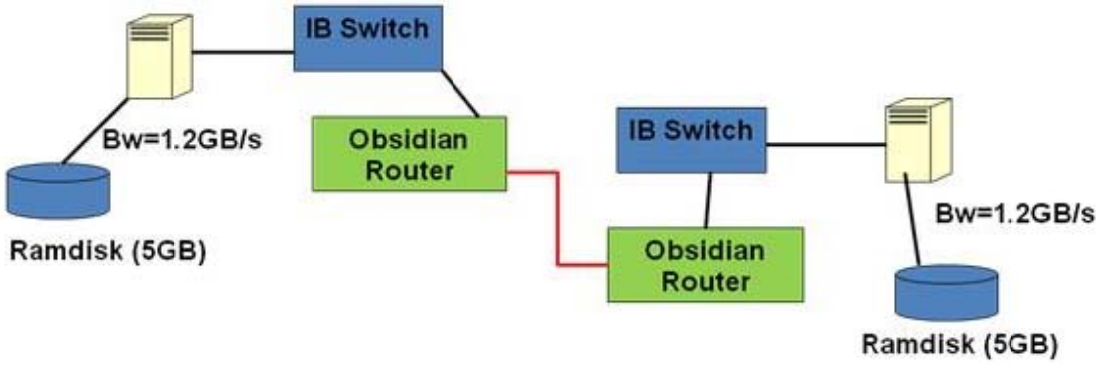

Figure 4: Dsync Experiment Settings

The experimental setting is depicted in Figure 4. We connected two nodes via an Obsidian router. Each node used a portion of its main memory as a ramdisk of 5GB size to store the data files to be synchronized. The bandwidth of this ramdisk was 1.2GB/s. One node was the source node that contained new files, another node was the target node that initiated a synchronization request to the source. Details of the experiments were as follows:

- The source node had one 1GB file in its ramdisk. We created many hard links to this actual file such that we could have arbitrarily large number of files (and large amount of data) to transfer, even if the ramdisk had finite size (5GB).
- We ran one, two, and four instances of dsync on the target node; each copy of dsync transferred 100GB data to amortize the uncertainty in the bandwidth measurement.
- We varied the dsync RDMA buffer size from 64KB to 512MB. We also enabled the option "Dsync::Inplace-Update" to be 1.
- The Obsidian router's port latency was set from 0 to 100ms, giving RTT from 0 to 200ms. We also toggled the hardware encryption option.

Figures 5,6,7 show the data transfer bandwidth using one, two, and four instances of dsync at varied RTT latencies and RDMA buffer sizes. We made several observations:

1. For a given RDMA buffer size, larger RTT latency damaged the bandwidth significantly. Dsync pushed out a RDMA buffer size's worth of data and then waited for acknowledgement from the other side. Increasingly bigger RTT meant dsync had to sit idle for longer time to get the ACK before resuming data transferring, leading to a degraded throughput.
2. For a given RTT value, increasing RDMA buffer size helped improve the throughput. Larger RDMA buffer size could pump more data into the pipeline between source and target nodes and helped achieve a better utilization of the link resource.
3. Using more instances of dsync could improve the bandwidth usage. For example, given RTT=200ms and RDMA buffer size 512MB, using one, two, and four instances of dsync could achieve throughput of 689MB/s, 915MB/s and 981MB/s, respectively.
4. Toggling hardware encryption did not impose noticeable effects on the throughput.

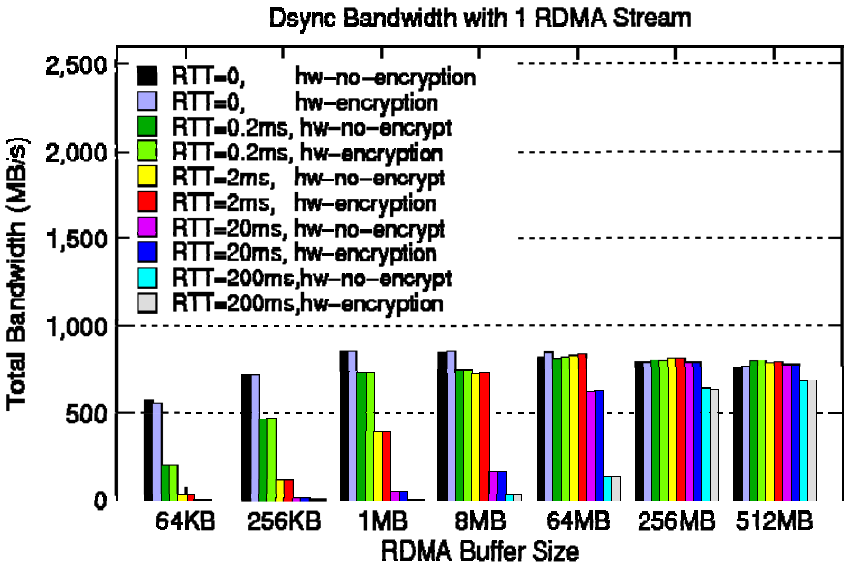

Figure 5: Dsync Transfer Bandwidth (1 instance of dsync)

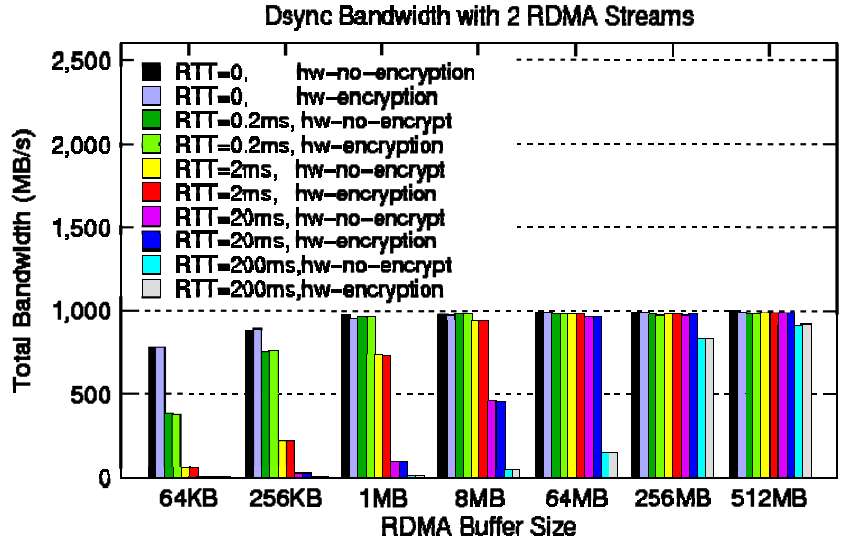

Figure 6: Dsync Transfer Bandwidth (2 instances of dsync)

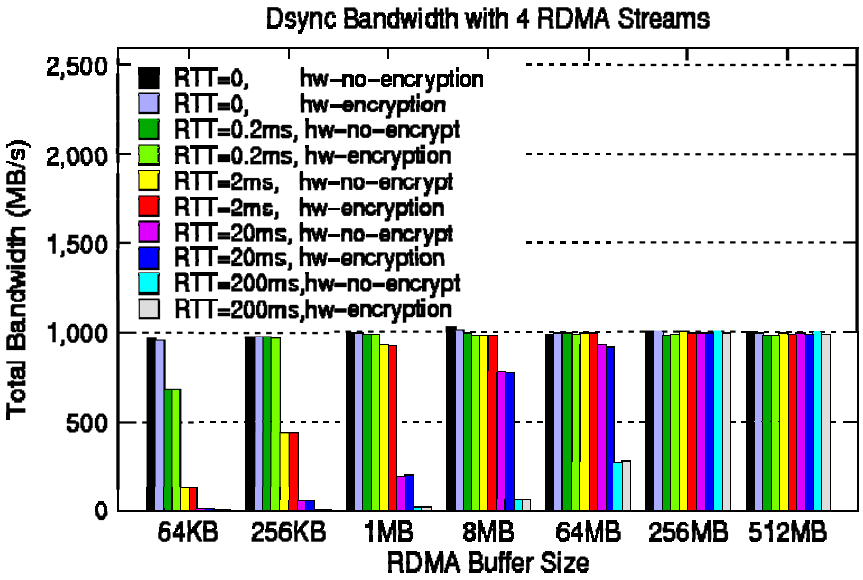

Figure 7: Dsync Transfer Bandwidth (4 instances of dsync)

4 Conclusions

In this report we evaluated the performance of Obsidian Longbow routers in state-of-the-art InfiniBand clusters. Our testbed consisted of two InfiniBand clusters interconnected with a pair of Obsidian routers with varying latencies. We ran the IOzone benchmark to measure data movement efficiency between compute nodes and storage clusters. We found that file write throughput did not vary significantly at increased latency, but longer latency adversely affected file read throughput. Using a higher degree of concurrent IO could help achieve better bandwidth utilization. We also ran dsync benchmark to measure data movement performance between two clusters. We found that dsync performance dropped significantly as the network latency increased. Better throughput was obtained with larger RDMA buffer size and concurrent RDMA streams, since they could pump more data into the data transfer pipeline to attain an efficient bandwidth utilization. Enabling hardware encryption did not generate a noticeable impact at the measured performance in both tests.