

Evaluating the Impact of RDMA on Storage I/O over InfiniBand *

Jiuxing Liu

Dhabaleswar K. Panda

Mohammad Banikazemi^{†1}

Computer and Information Science
The Ohio State University
Columbus, OH 43210
{liuj, panda}@cis.ohio-state.edu

[†]IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
mb@us.ibm.com

Abstract

Recently, several protocols such as iSER (iSCSI Extension for RDMA) and SCSI RDMA Protocol (SRP) have been proposed to improve the performance of storage I/O. These protocols promise a better performance by taking advantage of Remote Direct Memory Access (RDMA) operations. The need for design and development of new hardware and management infrastructure have made it difficult to evaluate these protocols and the impact of RDMA on storage I/O. On the other hand, recently InfiniBand solutions with RDMA capability have become available. Even though InfiniBand has been designed to be a unified interconnection network for both inter-processor and storage communication, its use has been mostly limited to inter-processor communication and not storage I/O. In this paper, we propose a new design through which the use of InfiniBand for storage I/O and the impact of RDMA operations on storage I/O can be evaluated.

Our design, which is called RDMA assisted iSCSI, extends the iSCSI protocol to exploit InfiniBand RDMA operations for data transfers. By keeping the TCP/IP based transport for management, control, and status messages unchanged, RDMA assisted iSCSI maintains compatibility with existing networking protocols and management infrastructure and makes its implementation in a reasonable time possible. We present the basic idea and several design and implementation issues and describe a prototype which has been implemented for the Linux operating system. Our evaluation demonstrates that RDMA can be very effective in improving the performance if certain requirements are met. We discuss what these requirements are. Our performance results show that with RDMA operations, we can improve the file read bandwidth from 96 MB/s to over 417 MB/s. We

also show latency and host CPU overhead can be reduced compared with the original iSCSI. Although our study is conducted in the context of InfiniBand, our results can be also applicable to other RDMA based designs such as iSER.

1 Introduction

During the last decade, the research and industry communities have been proposing and implementing user-level communication systems to address some of the problems associated with the legacy networking protocols. The Virtual Interface Architecture (VIA) [9] was proposed to standardize these efforts. More recently, InfiniBand Architecture [11] has been introduced which combines storage I/O with Inter-Processor Communication (IPC).

Due to its open standard and high performance, InfiniBand is gaining popularity in building high performance computing and database systems [12, 17]. These systems are cluster based, consisting of high-end machines connected by InfiniBand. To maintain compatibility and reduce management overhead, the TCP/IP protocol can be used in these systems for networking. IPoIB protocol [3] is the standard which defines the mapping of IP to InfiniBand. To achieve maximum performance, InfiniBand native protocols [7, 1] can be used for the communication needs of computing and database applications.

One crucial aspect of achieving high performance in cluster based systems is storage access. Many applications routinely process a large amount of data. Based on the interface provided, storage systems can be classified as Network Attached Storage (NAS) or Storage Area Networks (SANs). The NAS based approach has been discussed elsewhere [6, 8, 13]. In this paper, we focus on using the SAN based approach in InfiniBand clusters.

Traditionally, Fibre Channel (FC) has been used to build high performance SANs. However, in order to use FC in an

*This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506 and National Science Foundation's grants #CCR-0204429 and #CCR-0311542.

¹Corresponding author.

InfiniBand cluster, a separate FC network must be installed. For large clusters, the cost of additional hardware can be prohibitive. Therefore, it is desirable to build a SAN with existing hardware in an InfiniBand cluster. To address this issue, several solutions have been proposed.

SCSI RDMA Protocol (SRP) [23] is an ANSI standard which takes advantages of InfiniBand Remote Direct Memory Access (RDMA) to carry SCSI traffic. SRP can be implemented directly over native InfiniBand interfaces and achieve very high performance. However, SRP has to be implemented from scratch and it requires separate protocols and infrastructures to be developed for management.

Recently, iSCSI [26] has been proposed which uses TCP/IP as the underlying communication protocol for storage access. Since iSCSI is implement on top of TCP/IP, usually no separate network is needed for storage. Using TCP/IP protocols also greatly simplifies storage management because the same infrastructure in TCP/IP networks can be used to manage storage networks. Since IPoIB already defines IP protocol over InfiniBand, it seems more than natural to use iSCSI in an InfiniBand cluster to provide storage access. If the storage target does not have an InfiniBand interface, we can use gateways to connect it to the InfiniBand network. Then clients can use iSCSI to access storage through the gateway, as shown in Figure 1.

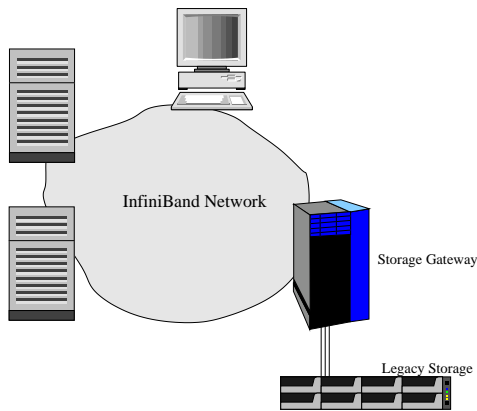


Figure 1. Using iSCSI in an InfiniBand Cluster

One problem in iSCSI is that overhead in TCP/IP may prevent one from achieving the highest storage performance. One major component of this overhead is due to extra memory copies, which greatly degrades storage bandwidth and increases host CPU utilization. To address this problem, special iSCSI customized HBA cards have been designed which can directly place data into destination buffers. However, this approach is not feasible with iSCSI running over InfiniBand without adding new hardware. As an extension to the original iSCSI protocol, iSER [19, 4] was proposed recently which exploits RDMA over IP [20]

to eliminate extra memory copies in the TCP/IP protocol. Similar to HBA based methods, this approach requires special “RDMA enabled NICs” (RNICs) and therefore is not feasible in an InfiniBand cluster either.

Although existing approaches cannot be used directly, the basic idea of placing data directly into the destination buffer without extra memory copies, can still be applicable in an InfiniBand cluster. Based on this idea, we propose using native RDMA operations in InfiniBand to enhance the original iSCSI protocol. Our design, which is called *RDMA assisted iSCSI*, preserves many advantages of the original iSCSI protocol such as backward compatibility and ease of management. The key idea of our design is to offload SCSI read and write data transfer in iSCSI from TCP/IP to InfiniBand RDMA operations. With this design, we not only avoid unnecessary copies, but also simplify the communication protocol. Although a similar idea was sketched elsewhere [27], in this paper we present a thorough study of this approach. We not only discuss our main idea as well as many design issues, but also carry out detailed performance evaluation. Our performance study shows that the RDMA assisted design can greatly improve storage bandwidth and reduce client overhead. It can achieve more than four times the bandwidth compared with the original iSCSI protocol in our InfiniBand cluster.

Our study in this paper not only provides a practical storage solution for InfiniBand clusters, but also can give insights into the potential of using RDMA in storage communication protocols in general. For example, iSER protocol requires RNICs that supports RDMA over IP. Although no such NICs are available yet, our design can serve as a close approximation to the iSER based approach. Therefore, some of our conclusions are also applicable to iSER.

In the remaining part of the paper, we first introduce some background information and related work in Section 2. In Section 3, we present the general idea of our RDMA assisted iSCSI design. We discuss detailed implementation issues in Section 4. Performance evaluation results are presented in Section 5. In Section 6, we give our conclusions and briefly mention some of the future work.

2 Background and Related Work

2.1 InfiniBand Architecture and iSCSI

The InfiniBand Architecture (IBA) [11] defines a switched network fabric for interconnecting processing nodes and I/O nodes. It provides a communication and management infrastructure for inter-processor communication and I/O. In an InfiniBand network, processing nodes and I/O nodes are connected to the fabric by Channel Adapters (CA). There are two kinds of channel adapters: Host Channel Adapter (HCA) and Target Channel Adapter (TCA).

HCA's are connected to processing nodes while TCAs connect devices. InfiniBand Architecture supports both channel and memory semantics. In channel semantics, send/receive operations are used for communication. In memory semantics, InfiniBand supports Remote Direct Memory Access (RDMA) operations, including RDMA write and RDMA read. RDMA operations are one-sided and transparent to the software layer at the receiver side.

iSCSI [26, 15] is a protocol that defines a transport layer for SCSI over TCP/IP. In this protocol, SCSI command, data and status are encapsulated into iSCSI Protocol Data Units (PDUs). iSCSI PDUs are then transferred between initiators and targets using the TCP/IP protocol. Since iSCSI is built on top of TCP/IP, it does not require a separate physical network for storage access traffic. Thus, its deployment and management cost is considerably less than the cost of other protocols such as Fibre Channel (FC). Different software/hardware combinations can be used to implement iSCSI [22, 21]. The iSCSI protocol can be handled completely in software. However, to improve storage access performance and reduce host overhead, the protocol can also be offloaded to the network interface cards [15, 16]. In the latter approach, it is possible to directly place SCSI data into the final destination memory without using any intermediate buffers in TCP/IP.

2.2 Related Work

The performance limitations of TCP/IP in high speed networks have been studied in [5]. In this subsection, we discuss the existing work which uses RDMA or similar approaches to address performance issues due to TCP/IP overhead in the iSCSI protocol. These include the iSCSI Extension for RDMA (iSER) protocol [19, 4], direct data placement in iSCSI Host Bus Adapters (HBAs) [21, 15], and the Voltaire iSCSI RDMA approach [27].

The iSER protocol was proposed recently to extend the iSCSI protocol by taking advantage of the iWARP [20] protocol suite. By using RDMA over IP provided by iWARP, iSER allows RDMA enabled NICs (RNICs) to directly put or get data from client buffers, without using intermediate memory copies in TCP/IP. This approach is similar to ours in that our design also uses RDMA to speed up iSCSI protocol and eliminate extra data copies. However, instead of using RDMA over IP and RNICs, which are not yet available, we have based our study on the RDMA operations provided by the InfiniBand Architecture and have shown how we can provide a high performance storage solution for InfiniBand based clusters. In many cases, our design can serve as an approximation of iSER. Thus, our study also provides insights into the potential of iSER.

The iSCSI protocol itself provides some general mechanisms to enable direct placement of SCSI data without extra

buffering in the TCP/IP protocol [15]. Using these mechanisms, a network interface card can be customized to allow iSCSI and TCP/IP to be offloaded. These iSCSI aware NICs are often called iSCSI Host Bus Adapters (HBAs). The iSCSI protocol provides information necessary for direct data placement. The HCA's then use this information to do placement and avoid extra copies. The basic idea of direct data placement in HBAs is similar to the aforementioned RDMA based iSER approach. However, instead of using a general RDMA interface, HBAs often rely on ad hoc protocols. Therefore, they tend to be incompatible with similar products. Because they do not have a uniform interface, it is difficult to develop software that works with different HBAs.

Recently, Voltaire Inc. has published a white paper describing how InfiniBand RDMA can be used in iSCSI for InfiniBand clusters [27]. Among all related work, their idea is most similar to ours. However, only high level concepts are described in their work. In this paper, we present many design issues and also conduct detailed performance evaluation.

3 Design of RDMA Assisted iSCSI over InfiniBand

As we have mentioned, using iSCSI in an InfiniBand cluster has the advantage of reusing existing software implementations and management infrastructures, which makes it possible to achieve fast deployment and easy transition. However, TCP/IP protocol over InfiniBand (iPoIB) [3] also has high overhead, making it difficult to obtain high performance and low overhead for storage access. Therefore, we base our design on the original iSCSI protocol and focus on investigating how InfiniBand features can be utilized to improve its performance.

The key idea in our design is to offload data transfers in iSCSI (SCSI read and SCSI write) from TCP/IP to InfiniBand RDMA operations. Figure 2 shows the bandwidth of raw InfiniBand RDMA and TCP over iPoIB in our testbed. (The details of the testbed used is described in Section 5.1.) We can see that for large messages, InfiniBand achieves much higher performance than iPoIB. This also suggests that TCP/IP over InfiniBand has much larger overhead. A large portion of this overhead is due to extra memory-to-memory copies in the TCP/IP protocol, which increases with the message size. In the iSCSI protocol, control traffic usually consists of small messages. However, data transfers (SCSI read and SCSI write) usually involve large messages. Therefore, in order to achieve high performance in the storage protocol, one has to reduce or eliminate this overhead in iSCSI data transfers. Next, we will discuss several design issues.

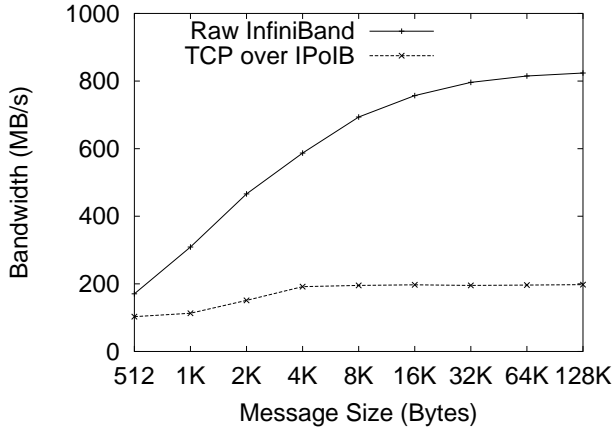


Figure 2. Bandwidth Comparison of Raw InfiniBand and TCP/IP over InfiniBand

3.1 Using RDMA for Data Transfer

In iSCSI, all communication traffic, including control information and data, goes through TCP/IP. Figure 3 shows the communication of SCSI read commands in the iSCSI protocol. To process a SCSI read command, iSCSI first creates a packet (or PDU) to transfer the read request from the initiator to the target. After getting this request, the target processes it and sends back the data to be read. Since iSCSI has a limit for the maximum number of bytes in a single PDU, more than one PDU may be required to send the data. Also the initiator may have limited buffering space. Therefore, after sending a certain amount of data, the target may need to wait for a flow control update from the initiator before it can continue sending more data. After all data has been sent, the target sends back the status information for the SCSI read command. For a SCSI write command, similar procedures happen, as shown in Figure 4.

The above approaches to handle read and write commands have potential performance problems when used directly in an InfiniBand cluster. First, as shown in Figure 2, TCP/IP over InfiniBand is not able to deliver high bandwidth. Therefore, performance for SCSI read and write will also be limited. Second, we see that for a single SCSI read or write command, multiple data packets and flow control packets may be required. Typically, each packet will generate at least one interrupt at the receiver. Since interrupts have high overhead, multiple incoming packets may significantly degrade performance. Also, having multiple packets means that the iSCSI protocol layer and the underlying communication layer must interact with each other multiple times to process these packets. These interactions can also increase communication overhead.

To address these problems, we study the use of RDMA

to offload SCSI reads and writes to InfiniBand RDMA operations. For SCSI read, instead of using TCP/IP, we let the target use RDMA write to transfer data directly to the destination buffer at the initiator. Similarly, in a SCSI write operation, the target uses RDMA read to get data from initiator. Figures 5 and 6 show the communication protocol after the offloading. RDMA operations require the user buffers to be registered. In our design, this registration information, together with buffer addresses, is attached with request messages from initiators.

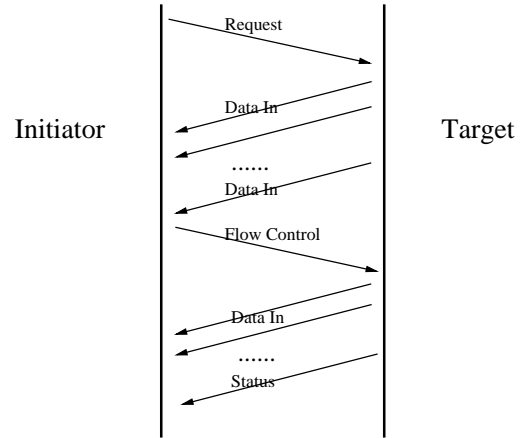


Figure 3. Data Read in iSCSI

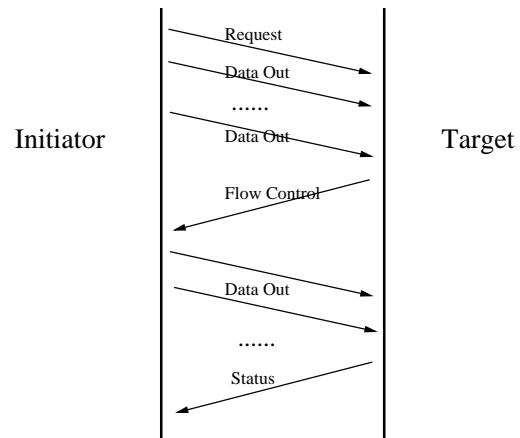


Figure 4. Data Write in iSCSI

We should note that control packets, including SCSI request and status packets, still go through TCP/IP in our design. Since control packets are usually small, communication performance is not significantly limited by the underlying TCP/IP protocol. More importantly, existing iSCSI implementations are usually driven by the processing of various control packets. Therefore, preserving the communication and processing of control packets can minimize port-

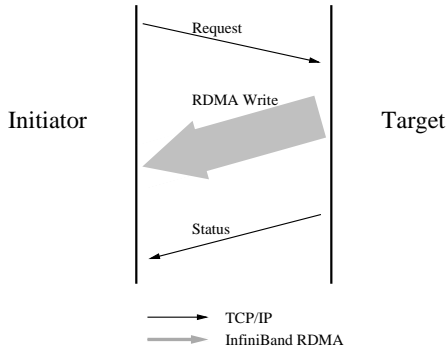


Figure 5. Data Read in iSCSI with RDMA

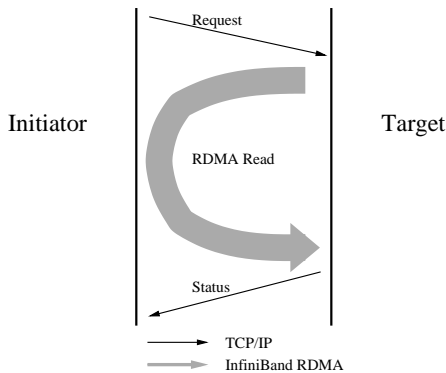


Figure 6. Data Write in iSCSI with RDMA

ing efforts from existing iSCSI implementations. We plan to study using InfiniBand also for control packets in future.

Using RDMA operations for iSCSI data transfer gives several advantages compared with the original iSCSI. First, since RDMA read and RDMA write are native InfiniBand operations, they have much higher performance than TCP/IP over InfiniBand which suffer from memory copies, as we have seen in Figure 2. Therefore, we can potentially achieve much higher bandwidth for data transfer in iSCSI. Second, since RDMA operations transfer data directly to or from the buffers at the initiator, there is usually no need for software flow control. If the buffers are contiguous, only one RDMA operation is needed for the entire SCSI read or SCSI write request. As a result, the overhead of processing multiple packets is also eliminated.

3.2 Dealing with Memory Registration Cost

By offloading SCSI data transfers to RDMA operations, RDMA assisted iSCSI can eliminate extra copies in the communication protocol. However, RDMA operations require that communication buffers are registered first. Buffer registration usually serves two purposes. First, it ensures the buffer will be pinned down in physical memory so that it can be safely accessed by InfiniBand hardware using DMA. Second, it provides the InfiniBand hardware with address translation information so that the buffer can be accessed

by virtual addresses. Buffer registration usually is a very expensive operation. Figure 7 shows the cost of registration for buffers of different sizes in our InfiniBand testbed. We can see that the cost is very high. Therefore, one central issue in our RDMA based design is to reduce or eliminate memory registration cost whenever possible. This problem can be easily addressed at the target side since storage targets usually have total control over their buffers and it is possible for them to pre-register the buffers to avoid registration cost. Next, we will discuss several techniques to deal with this issue at the initiator side.

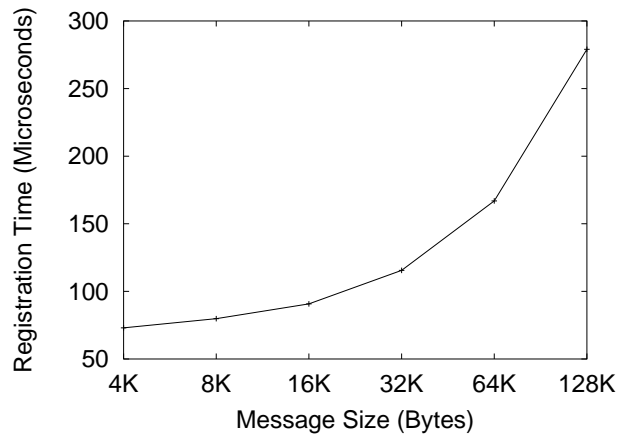


Figure 7. InfiniBand Memory Registration Cost

3.2.1 Memory Registration Cache (MRC)

One way to avoid memory registration cost is to use a memory registration cache. The basic idea is to maintain a cache of registered buffers and to do buffer de-registration in a lazy manner. When a buffer is first registered, it enters the cache. When it is de-registered, the actual de-registration is not carried out. Instead, the buffer stays in the cache. Therefore, when the buffer is used next time, it does not need to be registered since it is still in the registration cache. A buffer is de-registered when there are too many buffers in the registration cache.

The effectiveness of memory registration cache depends heavily on buffer reuse patterns. In MPI applications, we have observed high reuse rate of communication buffers [12]. Therefore, registration cache is very effective in that context. However, the buffer reuse pattern in storage systems may be quite different from those in scientific applications. Usually, data is cached at the client side, either by the operating system (file system cache) or the applications themselves (such as database systems). In these cases, after a I/O buffer is used, it will hold cached data. Therefore, it

is unlikely to be re-used again. As a result, buffer reuse rate may be low for storage clients. Therefore, memory registration cache may not be an effective technique for our design.

3.2.2 Fast Memory Registration (FMR)

Another way to reduce memory registration cost is to use a technique called Fast Memory Registration (FMR). In FMR, memory registration is divided into two steps. The first step is responsible for allocating resource needed by the registration, including address translation table in the InfiniBand HCA. The actual registration is done in the second step using resource obtained from the first step. Since resource allocation and de-allocation can be managed separately and in a batched fashion, usually only the second step will occur in the critical path of data transfer. As a result, FMR can achieve much faster memory registration.

FMR exists in RDMA over IP [20] and some InfiniBand implementations [14]. It is a very general technique and not dependent on application reuse patterns. Therefore, it can be used to reduce memory registration overhead in our design.

3.2.3 Zero-Cost Kernel Memory Registration (ZKMR)

In this subsection, we propose a new technique called Zero-Cost Kernel Memory Registration (ZKMR) to reduce memory registration cost for our storage clients. ZKMR is based on the observation that all buffers handed to our SCSI driver must have been pinned down in memory, otherwise the buffers would not be safely used for I/O operations. As we have discussed, memory registration include buffer pinning and address translation. Since buffer pinning is already done, we only need to inform InfiniBand hardware necessary address translation information.

We achieve the goal of address translation in two steps. First, we register all physical pages. In Linux, since all physical pages are mapped to a contiguous kernel virtual memory area, this step can be done with a single memory registration operation. Also since this memory area is physically contiguous, the registration consumes very little resource in the HCA. In the second step, we translate a buffer address to its physical address. Once this translation is done, registration information will be available because all physical pages are already registered in the first step.

We should note the first step in ZKMR is done during initialization. Thus, only the second step needs to be performed during memory registration. In Linux, this step can usually be achieved by a single arithmetic operation on the buffer address. We name it Zero-Copy Kernel Memory Registration because of its negligible cost. Although our implementation of ZKMR is specific to Linux, the general idea can be applied to other operating systems.

One drawback of ZKMR is that all physical pages must be registered. Since InfiniBand can only registered virtual addresses, this requires that all physical memory must be mapped to kernel virtual address space. In certain cases¹, not all physical pages can be mapped. In these cases, ZKMR cannot be used for memory pages that are not mapped initially.

3.3 Session Management

An iSCSI session is a logical association between the initiator and the target. A session must be established before iSCSI is functional. An iSCSI session usually consists of two phases: a login phase and a full-featured phase.

After the initiator has discovered an appropriate iSCSI target, It establishes a TCP/IP connection with the target. Then the login phase begins. One important task in the login phase is exchanging and negotiation of session parameters. After both sides agree on a set of parameters, the session enters full-featured phase, where the initiator can issue SCSI requests to the target.

Our design does not change the way bootstrapping and target discovery are done. Therefore, existing protocols such as iSNS [25] can still be used. The session login is also carried out as the original iSCSI protocol. However, we add a new session parameter which indicates whether RDMA offloading is supported or not. If both the initiator and the target are in the same InfiniBand cluster and support offloading, RDMA assisted iSCSI will be used to achieve better performance. Otherwise, RDMA offloading will not be used and the communication will be done as in the original iSCSI protocol. Therefore, our implementation has high compatibility with existing implementations. For example, it is able to inter-operate with existing iSCSI implementations on the same InfiniBand cluster that do not support offloading. It can also work with targets across a WAN that do not have InfiniBand connections.

3.4 Reliability and Security

The original iSCSI protocol puts much focus on reliability. The TCP/IP protocol uses checksum to help detect corrupted packets. In iSCSI, Cyclic Redundancy Code (CRC) can be used to achieve higher reliability. The use of CRC is optional and can be negotiated during login negotiation.

In our design, SCSI read and write data transfer uses InfiniBand RDMA operations whenever possible. Control traffic goes through TCP/IP, which also uses InfiniBand as the underlying data link layer. Since InfiniBand provides its own end-to-end CRC [11], CRC is no long necessary in the iSCSI protocol. As a result, iSCSI CRC is always negotiate to “none” when RDMA offloading over InfiniBand is used.

¹For example, Linux on X86 with more than 1GB physical memory.

Security is another important issue. During login phase, different protocols such as Challenge Handshake Authentication Protocol (CHAP) and Secure Remote Password (SRP) can be used for authentication in our design, just as in the original iSCSI protocol. To achieve data confidentiality, IPsec [24] can be used in the original iSCSI protocol. However, since SCSI read and write data transfer bypasses TCP/IP in our RDMA assisted iSCSI, IPsec cannot be used directly and confidentiality must be addressed explicitly. Since currently InfiniBand clusters usually consist of trusted nodes, most of the time performance is of higher priority than data confidentiality. However, we plan to investigate how to provide data confidentiality in future.

4 Implementation

We have implemented a prototype of our design. In this section, we provide an overview of our prototype, followed by discussions about issues such as memory registration, buffer re-organization and target implementation.

4.1 Implementation Overview

The Linux operating system organizes its SCSI subsystem in a layered structure [10, 18], as shown in Figure 8. At the upper level, Linux has different components to provide SCSI device abstractions such as block devices (SD for hard disks and SR for CDRoms) and character devices (ST for tape devices and SG for SCSI generic devices). Usually a SCSI hard disk is accessed through the SD interface. The middle layer is common for all devices. It is responsible for transforming commands from upper layer into SCSI requests. It then hands over these requests to lower layer drivers. At the bottom, lower layer drivers interface with Host Bus Adapters or provide an emulation layer for non-SCSI devices.

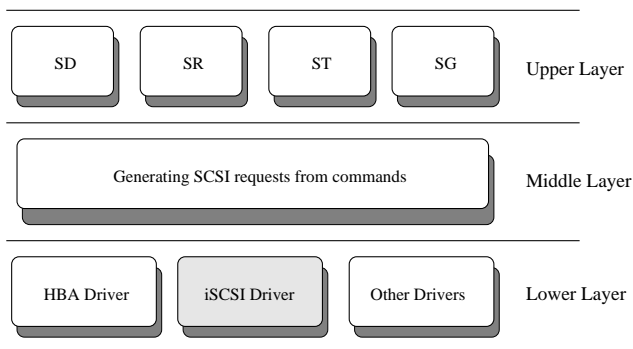


Figure 8. Linux SCSI Subsystem

At the client side, we have implemented our design as a SCSI driver at the lower layer of the Linux SCSI layers. Instead of driving a piece of hardware, our driver uses TCP/IP

and InfiniBand RDMA to deliver SCSI commands to a target node. As a result, storage provided by the target can be used transparently, just like any local SCSI disks. Our implementation is based on the Intel iSCSI project in SourceForge [2]. We used the InfiniBand Access Layer (IBAL) [1] to access functionalities provided by the underlying InfiniBand hardware.

4.2 Implementation of Memory Registration Handling

In the previous section we have proposed three techniques to reduce or avoid buffer registration cost for storage clients: MRC, FMR and ZKMR. As we have discussed, MRC may not be an effective method for our design. Currently, FMR is not yet supported in IBAL. Therefore, we have used ZKMR in our implementation.

4.3 Buffer Re-Organization

In order to improve performance, another optimization in our design is to reduce the number of RDMA operations for each SCSI read or write request. For each request, the target side information includes a LUN, a start address and a size. The client information usually includes a list of buffers. In InfiniBand, each RDMA operation can only access one contiguous buffer at the remote side while the local buffers can be a list of buffers (gather/scatter list). To reduce the number of RDMA operations needed, we re-organize the client buffer list to find as many contiguous chunks as possible. For each chunk, we use only one RDMA operation for data transfer. One example of this process is illustrated in Figure 9. In this example, initially the client side has four buffers. However, we can combine them into a single buffer through re-organization.

Buffer reorganization is very effective when our RDMA assisted iSCSI is used for Linux file system I/O. Because of the way Linux manages its page cache, most of the time the re-organizing process is able to combine the client buffer list into a single buffer. Thus, the data transfer process can be achieved using a single RDMA operation with very low overhead.

4.4 Target Implementation

In our prototype, we implemented a RAM disk based storage target running in the user space. Since we are interested in the benefits of RDMA in iSCSI, using a RAM disk helps us to make sure physical disk access will not become the performance bottleneck. In future, we plan to extend our implementation to a disk based storage target which uses main memory as cache.

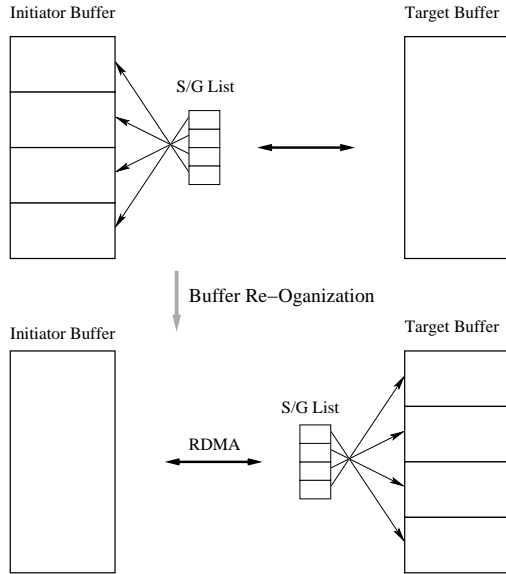


Figure 9. Client Based Buffer Reorganization

Our target implementation is single-threaded and serves one SCSI request at a time. Since most of the time client overhead is the performance bottleneck, currently this design is not a significant restriction. We plan to extend our implementation so that the storage target can handle multiple SCSI request simultaneously.

5 Performance Evaluation

In this section, we compare the performance of our RDMA assisted iSCSI with the original iSCSI implementation [2], which runs on top of iPoIB [3]. We also evaluate the impact of various techniques we used, such as Zero-Cost Kernel Memory Registration (ZKMR) and buffer reorganization.

5.1 Experimental Testbed

Our experimental testbed consists of a cluster system with a number of SuperMicro SUPER P4DL6 nodes. Each node has dual Intel Xeon 2.40 GHz processors with 512K L2 cache and a 400 MHz front side bus. The machines are connected by Mellanox InfiniHost MT23108 DualPort 4X HCA adapter through an InfiniScale MT43132 Eight 4x Port InfiniBand Switch. The HCA adapters work under the PCI-X 64-bit 133MHz interfaces. We used the Linux Red Hat 7.2 operating system with 2.4.18 kernel. SourceForge IB Access Layer version 1.118 was used for InfiniBand and iPoIB.

5.2 Benefits of RDMA Assisted Design

In order to study the benefits of our RDMA assisted design, we compare it with the original iSCSI implementation using micro-benchmarks such as bandwidth, latency and client host overhead. To avoid adding more complexity, we focus on SCSI read performance.

5.2.1 Bandwidth

In the bandwidth test, we use buffered read calls to get data from the iSCSI device file. Thus, the data will first go into the file system cache, and then be copied to the user buffer. It should be noted that buffered read bandwidth is affected by file system read-ahead policies. We set the maximum read-ahead window size to 255 in the tests. We study the effect of read-ahead window size in the next subsection.

Figure 10 shows the results for different block sizes. From the figure we see that without RDMA, iSCSI can only achieve bandwidth up to 96 MB/s. Its performance is limited by both the underlying iPoIB implementation and the extra memory copies from file system cache to the user buffer. However, with the help of RDMA, we can achieve bandwidth up to 417 MB/s for 16 KB blocks. Its performance is limited mostly by the memory copies from file system cache to the user buffer and other client side overhead such as protocol processing and interrupts. From the figure we can also see that when the block size becomes larger than 16 KB (8KB for the original iSCSI) bandwidth starts to decrease. This is due to processor cache effects.

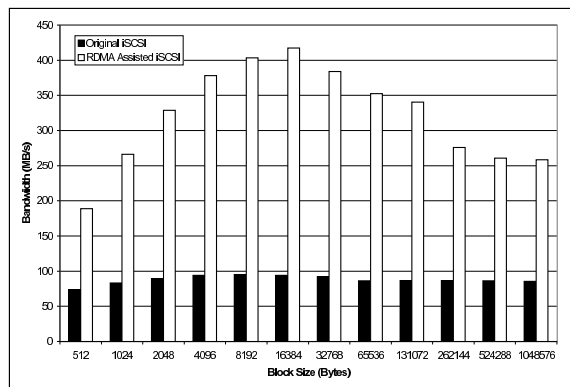


Figure 10. File Read Bandwidth

5.2.2 Impact of Read-Ahead Window Size

In the Linux file system, file read normally goes through file system cache. The operating system also has a mechanism to detect sequential file access patterns and uses file read-ahead (also called prefetching) to improve performance.

The read-ahead window size can have significant impact on performance.

In Figure 11, we show the bandwidth results for different maximum read-ahead window sizes. The block size used in the tests is 16KB. We can see that as the read-ahead window size increases, bandwidth also improves. This is because our tests read data sequentially and a bigger window size allows for more prefetching and more overlapping of different requests. For the original iSCSI, a window size of 15 is enough to achieve peak performance. However, RDMA assisted iSCSI needs a much larger window size (255) in order to achieve peak performance.

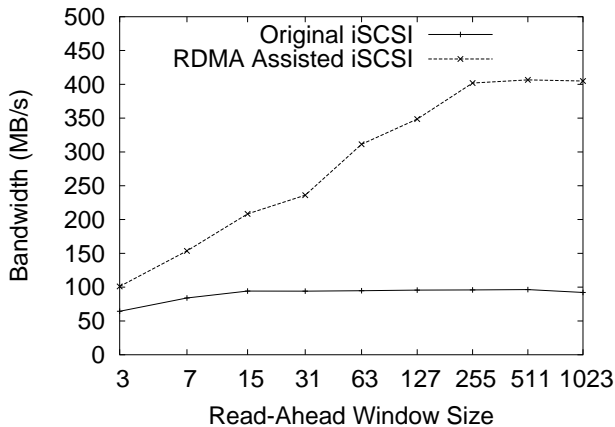


Figure 11. Impact of Read-Ahead Window Size

5.2.3 Latency

In the latency test, we need to eliminate the impact of file system read-ahead. Therefore, we used Linux raw I/O, which bypasses the file system cache and does not trigger file system read-ahead. Figure 12 shows the performance results. We can see that for block sizes larger than 1KB, RDMA assisted design outperforms the original iSCSI. However, factors of improvement are less compared with those in the bandwidth tests of the previous subsection. The reason is that requests for each block are serialized in the latency test. Since the RDMA assisted design uses TCP/IP for control messages, its performance is limited by the latency of IPoIB. In the bandwidth tests, different requests can be overlapped. Therefore, the RDMA assisted design can achieve a much better performance.

For small block sizes (1KB or less), we notice that the original iSCSI performs slightly better than the RDMA assisted design. This is due to an optimization in iSCSI called *phase collapse*. In phase collapse, the requested data and the status can be sent from the target to the initiator using a

single message, which can achieve better performance than using two separate messages. However, in the RDMA assisted design, two separate messages have to be sent: one through InfiniBand RDMA and one through TCP/IP. Therefore, we obtain a slightly lower performance.

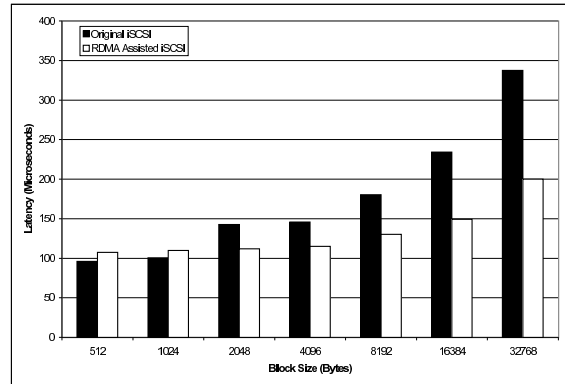


Figure 12. File Read Latency (Raw I/O)

5.2.4 Host Overhead

In the bandwidth tests which use buffered file system I/O, we have observed that client CPU utilization is close to 100% for both the original iSCSI and the RDMA assisted iSCSI. The high CPU utilization is mainly due to extra copies from file system cache to user buffers. This also implies that client side overhead is the performance bottleneck in the bandwidth tests. Since RDMA assisted iSCSI can achieve more than 4 times the bandwidth, it also means that with RDMA, CPU time can be reduced to less than one-fourth of the original time when transferring the same amount of data. Therefore, InfiniBand RDMA can greatly reduce host overhead.

We also show CPU utilization in the raw I/O latency tests. In Figure 13, we can see that RDMA assisted design consumes less CPU time compared with the original iSCSI for large block sizes. In fact, for large block sizes, CPU utilization decreases when the block size increases. However, CPU utilization for the original iSCSI increases with the block size. This is mostly due to the cost of copying. (For RDMA assisted design, there is a drop in CPU utilization for 1KB blocks. We are currently investigating this issue.)

5.3 Impact of Memory Registration and Buffer Re-Organization

In this section, we consider the impact of avoiding memory registration and doing buffer re-organization. As we have mentioned, we use ZKMR to reduce memory registration overhead. We also conducted bandwidth tests without using this technique. That is, we explicitly register and

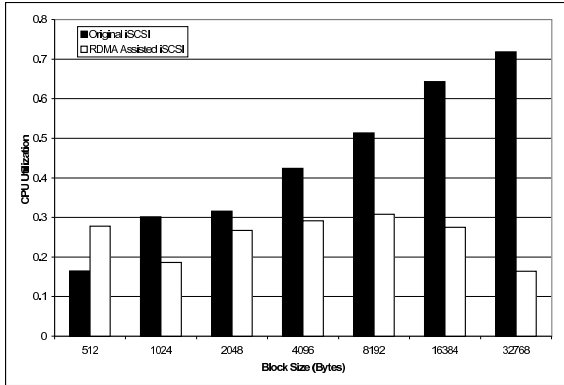


Figure 13. Host CPU Utilization in Raw I/O Latency Tests

de-register buffers during communication. The results are shown in Figure 14. We can see that performance drop significantly in this case. Therefore, avoiding high memory registration cost is very important in the RDMA assisted design.

In Figure 15, we compare bandwidth performance with and without using buffer re-organization. We can observe that using buffer re-organization only slightly improves the performance. This is because in the bandwidth tests, client side CPU is the bottleneck. Although buffer re-organization can reduce target side overhead, it does not bring much improvements to overall performance.

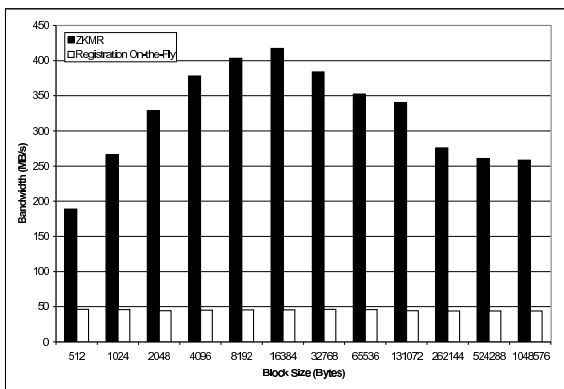


Figure 14. Impact of Memory Registration

6 Conclusions and Future Work

In this paper, we propose a design called RDMA assisted iSCSI that combines iSCSI with RDMA over InfiniBand which provides a high performance storage solution for In-

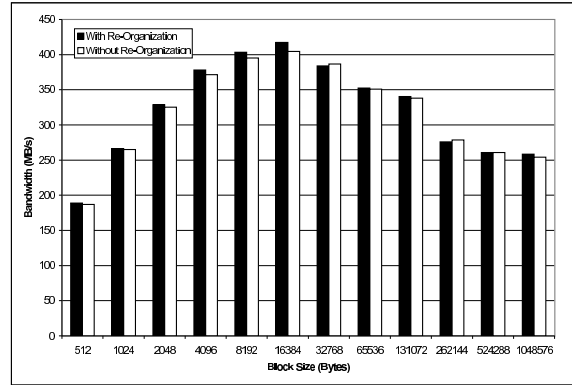


Figure 15. Impact of Buffer Re-Organization

finiBand clusters. The original iSCSI protocol is based on TCP/IP, which means that it can take advantage of existing networking infrastructure and reduce maintenance and management overhead. However, it also suffers from performance inefficiencies in the TCP/IP protocol such as extra copies. Our design offloads data transfer in iSCSI to InfiniBand RDMA operations. Thus, the communication protocol in iSCSI can be simplified and extra buffering in TCP/IP can be avoided, resulting in higher performance and less host overhead. This study can shed light on what can be expected from iSER, the protocol newly proposed to take advantage of RDMA operations.

We have discussed the basic idea and detailed design and implementation issues related to RDMA assisted iSCSI. A prototype has been implemented for the Linux operating system. Our performance evaluation shows that with RDMA assisted iSCSI, we can improve file read bandwidth from around 96 MB/s to over 417 MB/s. Our results also show that RDMA assisted iSCSI can reduce storage latency and host overhead. We have also found that handling buffer registration cost is very important in RDMA based designs. Unless this issue is addressed properly, we cannot obtain any benefit from RDMA.

Currently, we have implemented a RAM disk based target for our RDMA assisted iSCSI. In future, we plan to enhance this design by allowing access to real disks. We are also planning to carry out more performance evaluation by using real world applications. Another direction we are currently working on is to study the benefit of offloading iSCSI control packets from TCP/IP to InfiniBand.

Acknowledgments

We would like to thank Bulent Abali of IBM Research for valuable discussions and his support.

References

- [1] IBAL: InfiniBand Linux SourceForge Project. <http://infiniband.sourceforge.net/IAL/Access/IBAL>.
- [2] Intel iSCSI SourceForge Project. <http://sourceforge.net/projects/intel-iscsi>.
- [3] IP over InfiniBand Working Group. <http://www.ietf.org/html.charters/ipoib-charter.html>.
- [4] M. Chadalapaka, H. Shah, U. Elzur, P. Thaler, and M. Ko. A study of iSCSI extensions for RDMA (iSER). In *ACM SIGCOMM workshop on Network-I/O convergence: experience, lessons, implications*, August 2003.
- [5] J. Chase, A. Gallatin, and K. Yocum. End system optimizations for high-speed TCP. *IEEE Communications Magazine*, 39(4):68–74, 2001.
- [6] DAFS Collaborative. Direct Access File System Protocol, V1.0, August 2001.
- [7] DAT Collaborative. uDAPL and kDAPL API Specification V1.0, June 2002.
- [8] M. DeBergalis, P. Corbett, S. Kleiman, A. Lent, D. Noveck, T. Talpey, and M. Wittle. The Direct Access File System. In *2nd USENIX Conference on File and Storage Technologies (FAST '03)*, March 2003.
- [9] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. Merritt, E. Gronke, and C. Dodd. The Virtual Interface Architecture. *IEEE Micro*, pages 66–76, March/April 1998.
- [10] D. Gilbert. The Linux SCSI Subsystem in 2.4. http://www.torque.net/scsi/linux_scsi_24/.
- [11] InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.0, October 24 2000.
- [12] J. Liu, B. Chandrasekaran, J. W. W. Jiang, S. Kini, W. Yu, D. Buntinas, P. Wyckoff, and D. K. Panda. Performance Comparison of MPI Implementations over InfiniBand Myrinet and Quadrics. In *Supercomputing 2003: The International Conference for High Performance Computing and Communications*, Nov. 2003.
- [13] K. Magoutis, S. Addetia, A. Fedorova, M. Seltzer, J. Chase, A. Gallatin, R. Kisley, R. Wickremesinghe, and E. Gabber. Structure and performance of the direct access file system. In *Proceedings of USENIX 2002 Annual Technical Conference, Monterey, CA*, pages 1–14, June 2002.
- [14] Mellanox Technologies. Mellanox InfiniBand InfiniHost Adapters, July 2002.
- [15] K. Z. Meth and J. Satran. Design of the iSCSI Protocol. In *20th IEEE Symposium on Mass Storage Systems*, 2003.
- [16] J. C. Mogul. TCP Offload Is a Dumb Idea whose Time Has Come. In *9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, May 2003.
- [17] Oracle. Achieving Main-Frame Class Performance on Intel Servers Using InfiniBand Building Blocks. http://otn.oracle.com/deploy/availability/pdf/oracle_IB.pdf.
- [18] A. Palekar, N. Ganapathy, A. Chadda, and R. D. Russell. Design and implementation of a Linux SCSI target for storage area networks. In *5th Annual Linux Showcase and Conference*, November 2001.
- [19] RDMA Consortium. iSCSI Extensions for RDMA (iSER) and Datamover Architecture for iSCSI (DA) Specifications, 2003.
- [20] RDMA Consortium. iWARP Protocol Suite Specifications, 2003.
- [21] P. Sarkar, S. Uttamchandani, and K. Voruganti. Storage Over IP: When Does Hardware Support Help? In *2nd USENIX Conference on File and Storage Technologies (FAST '03)*, March 2003.
- [22] P. Sarkar and K. Voruganti. IP Storage: The Challenges Ahead. In *19th IEEE Symposium on Mass Storage Systems*, 2002.
- [23] Technical Committee T10. SCSI RDMA Protocol, 2002.
- [24] The Internet Engineering Task Force. IP Security Protocol. <http://www.ietf.org/html.charters/ipsec-charter.html>.
- [25] The Internet Engineering Task Force. IP Storage Protocols. <http://www.ietf.org/html.charters/ips-charter.html>, 2002.
- [26] The Internet Engineering Task Force. iSCSI Specification, 2002.
- [27] Voltaire Inc. High Performance SAN Connectivity for InfiniBand Fabrics. http://www.voltaire.com/pdf/storage_wp_final.pdf.