High Performance Block I/O for Global File System (GFS) with InfiniBand RDMA*

Shuang Liang Weikuan Yu Dhabaleswar K. Panda

Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{*liangs,yuw,panda*}*@cse.ohio-state.edu*

Abstract

State-of-the-art network technology has scaled to 10Gbps. However, TCP's high processing overhead and redundant data copies remain a major bottleneck for applications to fully benefit from such high speed technology. Remote Direct Memory Access (RDMA), as an emerging communication protocol, provides an opportunity for efficient storage system design by virtue of RDMA's none data touching semantics. Although RDMA based designs have been proposed to improve network file I/O protocols in several previous works, its benefit for cluster file system block I/O is not clear yet. We propose a novel technique – "buffer management delegation", which offloads message buffer management to remote communication party. Using this technique, we design our zero copy RDMA based block transfer scheme for GNBD, a block access protocol of Red Hat Global File System, to optimize cluster file system performance over 10Gbps InfiniBand network. We evaluate this new scheme with our copy based scheme as well as TCP over the same InfiniBand hardware. The evaluation quantifies the redundant copy impact for both bulk data transfer and file system meta-data operations. The results using open source file system benchmarks and widely used system utilities show that our implementation improves GFS performance up to 47% *compared with copy based scheme, and up to* 136% *compared with TCP.*

*This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506 and NSF Grants #CNS-0403342 and #CNS-0509452; grants from Intel, Mellanox, Cisco Systems and Sun Microsystems; and equipment donations from Intel, Mellanox, AMD, Apple, Appro, Microway, PathScale, IBM, Silverstorm and Sun Microsystems.

1 Introduction

As current technology trend evolves, computing system applications continue to be more data intensive than ever, rendering the throughput of storage systems increasingly important. For enterprise level data management, a shared storage model is usually used, where data is resident on a storage server or a cluster on a local area network and clients access data using file or block based protocols. These high end storage servers are usually equipped with large caches of several or even tens of gigabytes [5, 24]. Therefore, moving data efficiently across network interconnects becomes an important issue.

Today network technology has scaled rapidly to provide 1Gbps data rate for consumer network interface products, and 10Gbps products are emerging. Being able to utilize this abundant bandwidth would provide an exciting performance boost for storage systems. However, several studies [23, 3, 6, 18, 27, 19] have shown that TCP/IP protocol cannot fully utilize the bandwidth of high speed network. Its checksum processing and data copy contribute a lot to the per-byte overhead, leaving high performance network under-utilized especially for large data movement.

The Remote Direct Memory Access (RDMA) protocol [16] provides an opportunity for scalable network based storage system design. RDMA is widely used in high performance parallel computing area for inter-node communication. In the RDMA communication model, the sender can specify a memory location in remote communication party's address space and present a remote access key; then the underlying transport is responsible for placing the messages with no extra copies during the protocol stack processing. This elimination of unnecessary intermediate copies promises potential for system performance improvement, as it decreases the per byte overhead for data movement as well as CPU utilization and processor cache pollution. Together with OS bypass [22], high performance in-

terconnects such as InfiniBand [8], have been able to scale up to 10Gbps for memory to memory bandwidth.

While RDMA is widely used in high end parallel computing, the direct placement feature of RDMA fits well with storage servers, where data blocks are meant to be transfer instead of being touched. Several recent works [28, 12, 3, 11] have studied the benefits of using RDMA capable VIA [7] communication to improve database storage and file I/O protocols for Gigabit networks.

In this paper, we investigate the design issues of RDMA based network block I/O with 10-Gigabit InfiniBand Reliable Connection (RC) transport for Red Hat Global File System (GFS), a shared storage cluster file system. The motivation of our work is twofold. First, previous works on Gigabit networks were based on network technology significantly slower than memory speed. As the speed of 10Gbps network approaches current memory bandwidth, the impact of RDMA and zero copy on block I/O is not clear yet. Second, previous works focused on improving file I/O performance. However, block I/O is different from file I/O in that block I/O request demands data in multiples of block size, while file I/O request length varies depending on the protocol. In addition, file protocols sit one layer above block protocols, thus introducing network I/O at different service layer has different caching implications.

The main contributions of this paper are as follows.

1. We propose *buffer management delegation*, which offloads registered buffer management to GNBD client and eliminates extra control message overhead for our client-initiated RDMA data transfer scheme.

2. We design the block I/O transport for Red Hat GFS with InfiniBand RDMA support. Our experiments show that our implementation improves performance up to 136% and 70% for bulk data transfers and management operations respectively, when comparing with TCP over the same InfiniBand hardware.

3. Our experimental results indicate that for kernel based file system, the OS/FS management constraints the utilization of high performance networks, whose bandwidth is approaching current memory speed. Although the performance impact of redundant copies in the communication stack to block I/O is considerable, optimization of data copy reduction alone can not guarantee file system performance scalability to 10Gbps.

The rest of the paper is organized as follows: Section 2 provides background and related work; Section 3 discusses the design issues of RDMA based block I/O; Section 4 presents our design for GFS/GNBD; Section 5 illustrates the evaluation results; finally, we conclude the paper in Section 6.

2 Background and Related Work

Clustering is an important architecture in today's computing environment. On the Top500 list, more than 70% systems are cluster based supercomputers; it is also widely used in enterprises and research institutions for its good cost/performance ratio. Therefore, it is desirable to provide efficient data sharing services for cluster nodes. Global File System (GFS) [20] is a shared storage cluster file system, which provides consistent data access for each node. Although a lot of modern cluster systems are equipped with high performance interconnects such as Quadrics [14], Myrinet [2], and InfiniBand [8], currently GFS only supports TCP based communication. With IP emulations, TCP protocol can be used directly on these interconnects; however, it does not take full advantage of the RDMA semantics supported by the native communication protocol. By designing InfiniBand RDMA based block I/O protocol for GFS, we can quantify the data copy effects and TCP processing overhead over the same hardware. In this section, we first provide an overview of GFS and InfiniBand technology; then we present a summary of related work.

2.1 Global File System (GFS)

Red Hat GFS [17] is an open source project. It supports consistent file access through locking protocol on a shared storage environment such as Storage Area Network (SAN), where block I/O protocol commutes across a system area network. Two different block access protocols are designed with GFS for different hardware configurations. As shown in Figure 1, one is SCSI with direct attached SAN Fabric, the other is Global Network Block Device (GNBD) for remote storage access.

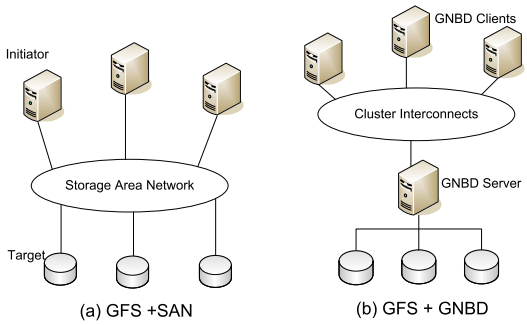


Figure 1. GFS Configurations

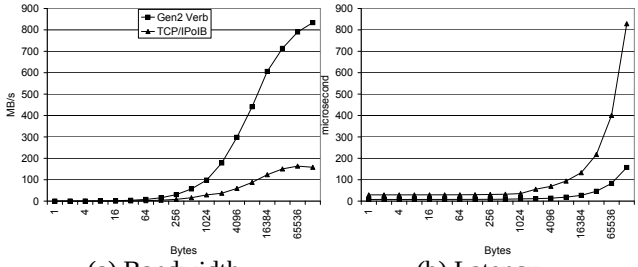
Currently, the direct attached SAN configuration requires hardware support, such as Fiber Channel switch and storage controller, which is still an expensive solution compared with InfiniBand. The GNBD solution provides a more general and cost effective way for shared storage access us-

ing a server node to perform similar functions as a SCSI target.

GNBD server is a user-land program. It processes block I/O requests from GNBD clients and serves data blocks. GNBD client is a pseudo block device acting as a SCSI initiator, which passes block I/O requests from the kernel based GFS and communicate with GNBD server over TCP/IP. In this paper, we use GNBD as the subject for RDMA based block I/O design and evaluation, for the relative cost efficiency of GNBD solution.

2.2 InfiniBand

InfiniBand [8] is an emerging open standard high performance interconnect, featuring low latency of a few microseconds and bandwidth up to 10Gbps with current generation of implementation. It is deployed on several large clusters on the Top500 list and is rapidly picking up as a cluster interconnect in the market. Several levels of Quality of Service (QoS) are supported in InfiniBand. The Reliable Connection (RC) service guarantees reliable transport and supports RDMA in hardware.



(a) Bandwidth (b) Latency

Figure 2. InfiniBand Gen2 Verb and TCP/IPoIB Stack Performance Comparison

InfiniBand kernel space software stack recently has been incorporated into the Linux kernel distribution. Several transport interfaces are supported with different performance and compatibility features. The Verb based API is the native communication software interface with almost raw performance, which provides both memory semantics interface, such as RDMA read and RDMA write, and normal channel semantics interface such as send/recv. IPoIB is an IP emulation for InfiniBand, legacy IP based application can run over InfiniBand without any modification. Figure 2 shows the user level memory to memory bandwidth on IA-32 PCI-X based systems for Verb API and TCP/IPoIB.

2.3 Related Work

In this paper, we investigate the design issues of RDMA based block I/O for GNBD using InfiniBand. The related work falls into two categories: networked storage protocol and InfiniBand based design.

Networked Storage Protocol: Shrimp [1], U-net [22] and VIA [7] are representative works for user-level network communication architecture for high speed network devices. They motivate follow-up works on user-level network based research. DAFS [12, 11] proposed a user space network file system client to take advantage of high performance user-level network. Unlike DAFS, NFSv4 over RDMA [3] proposed to use RDMA for NFS's kernel based RPC transport. However, both work are high performance network optimizations for file I/O protocols with Gigabit VIA based network. In addition, Zhou et al. [28] studied different Direct Storage Access (DSA) implementations with VIA for database storage; Carrera et al. [4] evaluated user-level communication features including RDMA for cluster based WWW server.

Sarkar et al. [19] evaluated storage protocols using software approach, as well as two hardware approaches: TOE and HBA, where the former offload TCP/IP processing to the network interface and the latter offload the whole storage protocol. Radkov et al. [15] compared the performance between NFS and iSCSI for IP networked storage and showed iSCSI based storage protocol's performance benefits from its better aggregation and caching capability. Both works target for IP based network storage.

A recent industry effort proposed iSER and SRP [21], which define SCSI transport over RDMA. But these work are still in prototyping stage. And in this paper, we propose different RDMA schemes for GNBD compared with these efforts.

InfiniBand Based Design: MVAPICH [10] is an RDMA based MPI implementation over InfiniBand. Wu et al. [26] designed an InfiniBand based transport for Parallel Virtual File System (PVFS). Recently, Liang et al. [9] proposed to use InfiniBand RDMA based network block device for remote swapping, where a copy based RDMA transfer scheme was designed. In this paper, we extend that work with a novel zero copy based RDMA design and integrate both designs in GFS for performance study with file system benchmarks, which quantifies TCP/IP's processing overhead and the data copy impact on InfiniBand RDMA transport for shared storage file system block I/O.

3 Issues for RDMA based Block I/O

Block I/O is different from file I/O in terms of the size of an I/O request. File I/O requests can be any number of bytes defined by the file system protocol, while block I/O requests always demand data in units of multiple blocks (typical block sizes are 4K or 8K). Operating system organizes the block buffers and pass them to the appropriate block devices for actual I/O transfer. The aggregate nature

of block I/O allows it to benefit from network bandwidth by larger data transfers. To optimize block I/O with RDMA, it is beneficial to further combine multiple blocks adjacent in memory and use hardware supported scatter/gather I/O to reduce the number of request messages. In addition, RDMA requires special message buffers and different transfer schemes can have different performance implications. In the section, we discuss these design issues.

3.1 Message Buffer Registration

To achieve zero copy RDMA, high performance interconnects depend on Network Interface (NI) aware DMA-able message buffers. Thus, message buffers must be registered with the NI before any transfer. The registration makes sure that the buffers are locked in memory. It also creates an address translation entry for the NI to address these buffers.

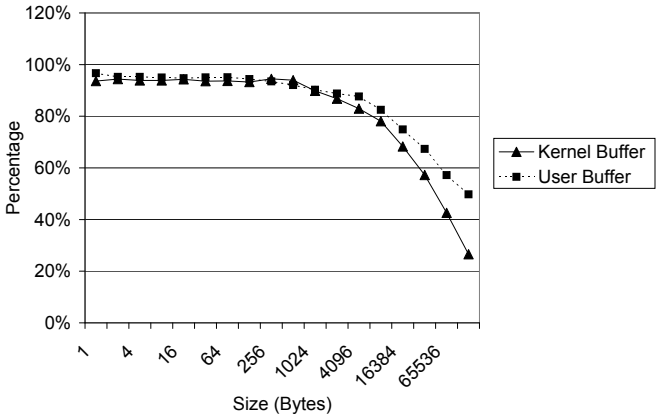


Figure 3. Buffer Registration vs. Message Latency

Memory registration operation is a costly operation. Figure 3 shows the ratio of memory registration overhead over the message latency for different message sizes. To minimize this impact, several solutions such as pre-registered buffer pool, Fast Memory Registration (FMR) [26], and registration cache have been proposed with various trade-offs.

To reduce registration cost for user space message buffers, an interesting observation is that block I/O buffers are locked down in kernel space. For InfiniBand stack, physical memory descriptor can be utilized to avoid expensive on-the-fly registration, if unrestricted RDMA operations to this node can be effectively managed to prevent remote party from corrupting irrelevant memory regions. One simple solution is to avoid being the passive side of RDMA operations; the other one is to use memory windows, which allow dynamic memory access control on registered buffers.

3.2 RDMA Transfer Alternatives

Several RDMA transfer schemes are possible for block I/O. Based on the active side of the operation, they can be classified into: (i) Target based RDMA, (ii) Initiator Based RDMA, and (iii) Hybrid RDMA.

Target Based RDMA: As illustrated in Figure 4(a), initiator sends block I/O requests to remote target. For block read requests, the target prepares the blocks and RDMA writes the blocks back to the initiator followed by a reply message. For block write requests, the target obtains the blocks using RDMA read from the initiator and sends a reply message for acknowledgment.

In this scheme, the target needs to cope with both request/reply control messages and RDMA based data transfer processing, while the initiator needs to send the exposed block buffer addresses to the target in the requests. This transfer scheme is used in iSCSI/iSER and SRP protocol.

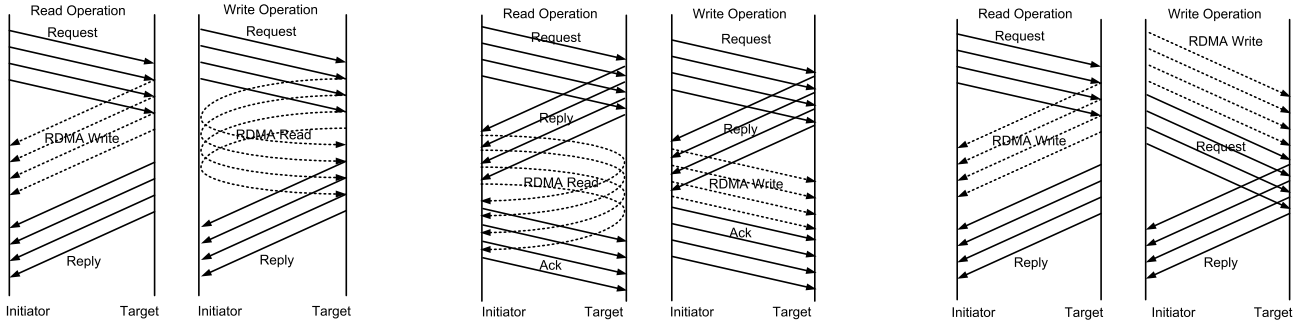
Initiator Based RDMA: As illustrated in Figure 4(b), the target offloads the RDMA transfer processing to the initiator by sending the target side RDMA buffer address in the reply. Extra control messages are required for completion notification to the target for resource management. To reduce the overhead, completion acknowledgment can be piggybacked with the next request when appropriate.

Hybrid RDMA: The above two schemes use RDMA read and RDMA write for data transfer. A potential problem with such schemes is that RDMA read is a round trip operation involving higher latencies. More importantly, RDMA read may not follow the ordering rules according to InfiniBand specification and can complete out of order with respect to later send or RDMA write operations.

Assuming initiator knows the remote message buffer address beforehand, a hybrid scheme can be used. Thus both sides use only RDMA write for data transfer, as illustrated in Figure 4(c). The assumption here seems strong, but it is achievable and can be implemented with a technique called Management Delegation (MD). More details are described in Section 4.1.

3.3 Scatter/Gather I/O and RDMA

InfiniBand scatter/gather operation allows non-contiguous message buffers to be posted in a single operation, thus reducing bus transactions and additional overhead needed for multiple message processing. The benefit of scatter/gather I/O for PVFS file I/O is studied in [25]. For block I/O, even a single request may be composed of multiple blocks that reside in non-contiguous



(a) Target Based RDMA (b) Initiator Based RDMA (c) Hybrid

Figure 4. RDMA Transfer Schemes

memory. Aggregating them in a single message posting transaction promises further performance improvements.

However, InfiniBand architecture has one constraint for RDMA based scatter/gather I/O: non-contiguous message buffers can only be on the active side of RDMA operation; that is, it only supports gather for RDMA write and scatter for RDMA read. But it is possible that both sides have the block buffers in non-contiguous memory, especially on the initiator side, where the file system blocks in kernel buffer cache are getting fragmented over time. So the trade-off here is either we can reduce the message number by a copy to a contiguous buffer or use separate messages for each non-contiguous buffer. One optimization applicable before using either of the above techniques is to do block aggregation, which combines block chunks adjacent to each other into a single chunk to minimize overhead in both cases.

4 The Design of GFS/GNBD

In Section 3, we discussed major issues for RDMA based block I/O. In this section, we present our design choices for GFS/GNBD implementation. Based on the software architecture of GNBD protocol and InfiniBand performance features, we propose our new zero copy RDMA based block I/O scheme and discuss related issues such as message ordering, memory management, and flow control.

4.1 Zero Copy RDMA with Buffer Management Delegation

Three RDMA transfer schemes are possible, as discussed in Section 3.2. Although the server based RDMA is adopted in current proposals for SRP and iSCSI/iSER, it has its limitations for GNBD design. First, to achieve zero copy RDMA for the GNBD client under such scheme, the client has to send a list of scatter/gather entries to the server, which may potentially be very long. Second, for block reads, the server may end up with multiple RDMA write operations for a single request if the blocks are non-contiguous. Third, to avoid memory registration, controlled memory access has to be dealt with.

To avoid the problems discussed above, we propose to use client based RDMA design with a technique called *server buffer management delegation* (MD) to achieve zero copy. With MD, the server allocates a block buffer pool exclusively for each client and delegates the management of the buffer to the client, so the client can issue RDMA operations immediately to the server without extra message exchanges for address information. This technique enables the implementation of a hybrid design, as discussed in Section 3.2. However, due to RDMA write's inability of scatter operations and performance features of the InfiniBand hardware, we choose to use RDMA read for block read operations as shown in Figure 5. With such a design, the client remains safe from exposing its physical memory address space. At the same time, the server is safe from client failures by the exclusiveness of these buffers.

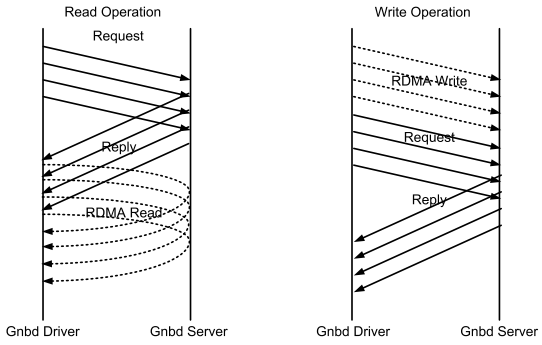


Figure 5. Management Delegation based RDMA Scheme

4.2 Handling Request Ordering

Although InfiniBand RC service is a reliable connection service, request ordering needs to be handled when both

RDMA read and RDMA write operations are used. According to InfiniBand specification, RDMA read operation does not preserve the normal ordering semantics as send and RDMA write. In particular, RDMA read may complete after later send/RDMA write completes due to its round trip nature, though multiple RDMA read operations are ensured to maintain ordering among themselves. Therefore, our design needs to address this problem to maintain ordering constraints that block I/O requires.

In our design, a simple algorithm is used to fence send/RDMA write requests immediately following the RDMA read to maintain proper ordering. A fence flag is used to notify send or RDMA write operations if there is an outstanding RDMA read. When the following order-preserving operation arrives, the flag is canceled. Then when a RDMA read request completes, it checks if it is still the owner of the flag it has set. If the result is positive, the flag is canceled. Our experiments show that the impact of fencing on performance is negligible.

4.3 Memory Management and Flow Control

In GNBD protocol, two types of messages are involved: *control message* and *data block message*. Control messages are small messages of fixed length holding request and reply information implemented using normal send/recv semantics. Data messages are actual blocks from/to storage systems with variable length implemented using RDMA semantics. Although the GNBD client can avoid managing data buffers for block I/O transfer with pre-registered physical address space, it still needs to manage the delegated buffer pool from the server. In addition, the server also needs registered buffer for control messages.

In face of the high registration cost for InfiniBand communication, we choose to use pre-registered buffer pools for the server. For control messages, a static ring buffer is implemented; and for data messages, a first fit algorithm is implemented to manage buffers dynamically. On the client, we use the same ring buffer management mechanism for control message buffer. Although it is possible to use dynamic memory management interface provided by kernel, the simple ring buffer scheme is more efficient.

For client/server based service, the server always needs to deal with unexpected requests from clients with limited resources, thus flow control must be considered. In particular, for InfiniBand RC transport, receive buffers for send operations are expected to be pre-posted.

In our current implementation, we use a similar watermark based flow control scheme as in [9]. We tune the threshold to over-provision the credits, thus excluding the performance impact in our experiments. Although the current scheme is static, a dynamic scheme with client/server negotiation and on-line adjustment is certainly possible within the design framework. More issues such as fairness among multiple clients, strategies for bandwidth allocation deserve further discussion, which are out of the scope of this paper.

5 Evaluation

To evaluate the impact of RDMA transport on GFS, we compare our zero copy based RDMA scheme with TCP over IPoIB, an IP emulation on the same InfiniBand hardware. We also implement a server based buffer copy RDMA scheme for GNBD to quantify the data copy impact. In this section, we present the experimental performance results.

5.1 Experiment Setup

The experiments are conducted on an Intel Xeon 2.4GHz cluster. Each node has 1GB memory and 64 bit PCI-X 133 MHz bus. All nodes are connected to InfiniBand network using InfiniScale MT43132 eight port switch and Mellanox MT23108 HCA. One node is set up as the GNBD server, which exposes the direct attached hard disk as block device. The operating system is Linux with OpenIB 2.6.9-11 kernel. The OpenIB [13] user-space utility is SVN revision 4507.

The workload in our experiment ensures that most of the GNBD server I/O happens within the buffer cache. This simulates the case of storage servers with large cache and high storage I/O bandwidth of several hundred Megabytes. We create GFS file system using the default 4K block size. Multiple runs are conducted for each test; and the average performance numbers are reported.

Several open source file system benchmarks and system utility are used for our evaluation:

IOZone is a file system benchmark, which tests a variety of file operations such as read/re-read, write/re-write with different I/O mode. We use it for file read/write bandwidth test under different modes.

Fileop stresses file system meta-data operations such as link, unlink, create, and delete.

Postmark is designed to measure the transaction rate for a pool of small files. Each transaction is either a read/append pair or create/delete pair. It creates a workload similar to Internet email server.

Tar is a widely used utility program to pack/unpack small files. A lot of file operations are involved using this tool.

In the following, we refer to our zero copy based RDMA scheme as *IB Zcopy*, refer to server based buffer copy RDMA scheme as *IB Bcopy*, and refer to TCP over IPoIB as *IPoIB*.

5.2 Results

IOZone Write Bandwidth: In Figure 6, we show write bandwidth test results with 128M files using three different modes. In "buffer write" mode, data is written back to the file system buffer cache with dirty block flushing happening in background asynchronously. In "flush write" mode, dirty block flushing from buffer cache happens synchronously within the benchmark timing. In "direct write" mode, file system buffer cache is bypassed and data is flushed directly to remote server; however, in this mode the benchmark reports blocking I/O interface results, which only allows one I/O system call outstanding.

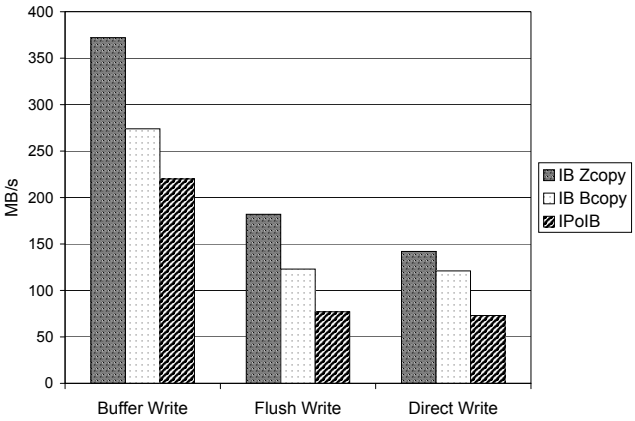


Figure 6. IOZone Write

The numbers shown in Figure 6 reveal that using TCP as block I/O transport for GFS cannot provide satisfactory performance. As shown by the performance difference between IPoIB and IB Bcopy scheme, IB Bcopy performs up to 65% better than TCP/IPoIB among these three modes. This indicates that the processing overhead other than data copy contributes significantly to the network under-utilization problem for 10Gbps interconnect. For IB Bcopy, protocol offloading to InfiniBand hardware reduces such influence.

Comparing the performance of IB Zcopy and IB Bcopy, we see that data copy degrades the bandwidth by up to 47% among these cases. The impact is both on latency and CPU utilization. In "direct mode", the main latency impact causes IB Zcopy to perform 17% better than IB Bcopy. In "buffer write" mode and "flush write" mode, where the background flushing thread (performing *write behind*) competes CPU with the main thread, IB Zcopy performs 36% and 47% better than IB Bcopy respectively. This indicates the data copy's CPU utilization impact on performance is more significant.

IOZone Read Bandwidth: Figure 7 shows the read performance with 128M files. In "buffer read" mode, data is read from buffer cache with processor cache effects on. In "purge read" mode, data is read from buffer cache with processor cache effects for the target data buffer off. In "direct read" mode, reading bypasses buffer cache and data is delivered from remote server directly.

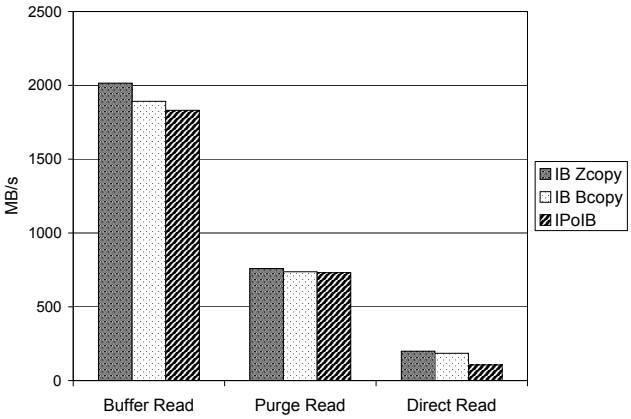


Figure 7. IOZone Read

Compared with the write results, the performance difference among IB Zcopy, IB Bcopy and TCP is minor because in the first two test modes most operations happen locally. The difference between "buffer read" and "purge read" exhibits the processor cache effects, which is as large as 165%. However in "dircet read" mode, TCP's high processing overhead repeats itself, where our IB Zcopy scheme is 87% better. For IB Zcopy and IB Bcopy, the copy elimination improves performance by 7%.

We have an interesting observation from the bandwidth test: Although our 10Gbps InfiniBand can achieve bandwidth of 834MB/s (as shown in Figure 2), our zero copy RDMA based GFS can only achieve a fraction of that bandwidth for bulk data read/write. Two factors possibly contribute to this effect in our experiments. One is kernel requests issuing rate; the other is server requests serving rate. To narrow down the problem, we hacked our implementation to nullify bulk data RDMA and server side data flushes operations for the write bandwidth test to see the impact. The result shows that the performance improvement is still minor. This indicates that management layers between application and zero copy RDMA based block I/O restrict the potential of high performance interconnects. Therefore, kernel based file system needs to provide more efficient services to scale with next generation interconnects.

Fileop Meta Data Operations: Figure 8 shows the results of Fileop. The benchmark creates a 20-ary directory tree of height two, each leaf is a one byte file. Fileop stresses the file system meta-data operations. Although RDMA is not significantly beneficial to small messages than copy

based send, this graph shows that the trend among different network protocols still persists for most cases due to the aggregate nature of meta-data operations using block I/O based storage architecture.

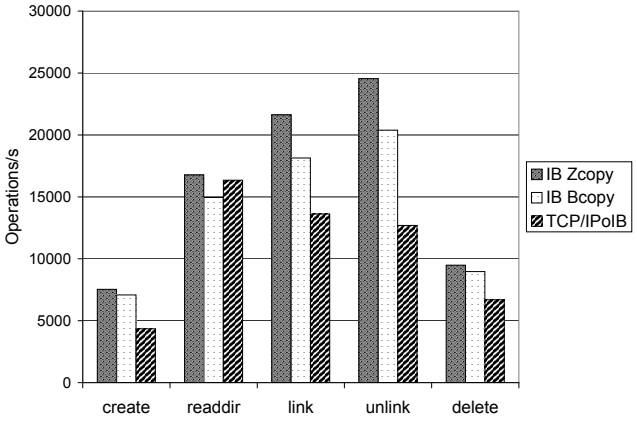


Figure 8. Fileop

Postmark and Tar: Table 1 shows the performance of postmark with 2000 file pool and 2000 transactions as parameters. Table 2 shows the performance of extracting a zipped tarball of around 52M. Both benchmarks involve file creation and small write operations. The result shows that the trend is consistent with those reported by IOZone and Fileop. In particular, the extraction test shows that zero copy based RDMA performs 21% better than copy based RDMA and 47% better than IPoIB.

Table 1. Postmark Performance

Postmark	IBZcopy	IBBcopy	TCP/IPoIB
Transactions/s	666	666	500

Table 2. Extracting Zipped Tarball

tar -zxf	IBZcopy	IBBcopy	TCP/IPoIB
Seconds	13.18	16.621	20.518

In summary, the results show that TCP based processing entails a high overhead for GFS block I/O besides data copy. Data copy reduction is more beneficial for less CPU utilization rather than latency optimization in the conducted tests. The experiments also show that as network bandwidth approaches memory bandwidth, the OS/FS management layer is required to provide more efficient services in order to scale with next generation high performance interconnects.

6 Conclusions and Future Work

InfiniBand's hardware support for RDMA allows efficient data transfer for storage block I/O. We apply this feature in the design of the block I/O access protocol of GFS, a shared storage cluster file system. We propose a new technique, *buffer management delegation*, to implement zero copy RDMA block transfer from client buffer cache to remote GNBD server. The evaluation using several benchmarks and frequently used system utilities shows that zero copy RDMA can improve block I/O performance for up to 47% compared with our copy based RDMA implementation and 136% compared with TCP over the same InfiniBand hardware. The result also indicates that more efficient and scalable management infrastructure is needed for kernel based file system to scale with next generation high performance Interconnects.

In the near future, we intend to investigate buffer caching related issues for GNBD server design. We also would like to evaluate our block I/O design together with upcoming iSCSI/iSER implementations.

References

[1] M. A. Blumrich, K. Li, R. Alpert, C. Dubnicki, E. W. Felten, and J. Sandberg. Virtual memory mapped network interface for the shrimp multicomputer. In *Proceedings of the 21ST annual international symposium on Computer architecture(ISCA'94)*, pages 142–153, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[2] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, 1995.

[3] B. Callaghan, T. Lingutla-Raj, and A. Chiu. NFS over RDMA. In *ACM SIGCOMM 2003 Workshops*, 2003.

[4] E. V. Carrera, S. Rao, L. Iftode, and R. Bianchini. User-Level Communication in Cluster-Based Servers. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture(HPCA'02)*, page 275, 2002.

[5] Z. Chen, Y. Zhou, and K. Li. Eviction-based Cache Placement for Storage Caches. In *Proceedings of USENIX Annual Technical Conference, General Track*, pages 269–281, 2003.

[6] H. K. J. Chu. Zero-Copy TCP in Solaris. In *USENIX Annual Technical Conference*, pages 253–264, 1996.

[7] D. Dunning and G. Regnier. The Virtual Interface Architecture. In *Proceedings of Hot Interconnects*, 1997.

[8] InfiniBand Trade Association. The InfiniBand Architecture. http://www.infinibandta.org/specs.

[9] S. Liang, R. M. Noronha, and D. K. Panda. Swapping to Remote Memory over InfiniBand: An Approach using a High Performance Network Block Device. In *Proceedings of International Conference on Cluster Computing(CLUSTER'05)*, 2005.

[10] J. Liu, J. Wu, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. *International Journal of Parallel Programming*, 32(3):167–198, 2004.

[11] K. Magoutis, S. Addetia, A. Fedorova, and M. I. Seltzer. Making the Most Out of Direct-Access Network Attached Storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, 2003.

[12] K. Magoutis, S. Addetia, A. Fedorova, M. I. Seltzer, J. S. Chase, A. J. Gallatin, R. Kisley, R. Wickremesinghe, and E. Gabber. Structure and Performance of the Direct Access File System. In *Proceedings of USENIX Annual Technical Conference,General Track*, pages 1–14, 2002.

[13] Open InfiniBand Alliance. OpenIB. http://www.openib.org.

[14] F. Petrini, E. Frachtenberg, A. Hoisie, and S. Coll. Performance Evaluation of the Quadrics Interconnection Network. *Journal of Cluster Computing*, 6(2):125–142, April 2003.

[15] P. Radkov, L. Yin, P. Goyal, P. Sarkar, and P. Shenoy. A Performance Comparison of NFS and iSCSI for IP-Networked Storage. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004.

[16] RDMA Consortium. An RDMA Protocol Specification. http://www.rdmaconsortium.org.

[17] Rea Hat Inc. Red Hat GFS Documentation. http://www.redhat.com/docs/manuals/csgfs/browse/rh-gfs-en/.

[18] G. Regnier, S. Makineni, R. Illikkal, R. Iyer, D. Minturn, R. Huggahalli, D. Newell, L. Cline, and A. Foong. TCP Onloading for Data Center Servers. *IEEE Computer*, 37(11):48–58, 2004.

[19] P. Sarkar, S. Uttamchandani, and K. Voruganti. Storage Over IP: When does Hardware Support Help? In *Proceedings of the 2rd USENIX Conference on File and Storage Technologies*, 2003.

[20] S. R. Soltis, T. M. Ruwart, and M. T. O'Keefe. The Global File System. In *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems*, pages 319–342, College Park, MD, 1996. IEEE Computer Society Press.

[21] Storage Networking Industry Association. iSCSI/iSER and SRP Protocols. http://www.snia.org.

[22] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-Level Network Interface for Parallel and Distributed Computing. In *Proceedings of the 14th ACM Symposium on Operating Systems Principles*, pages 40–53, 1995.

[23] K. Voruganti and P. Sarkar. An Analysis of Three Gigabit Networking Protocols for Stroage Area Networks. In *Proceedings of International Conference on Performance, Computing, and Communications*, 2001.

[24] T. M. Wong and J. Wilkes. My Cache or Yours? Making Storage More Exclusive. In *Proceedings of USENIX Annual Technical Conference,General Track*, pages 161–175, 2002.

[25] J. Wu, P. Wyckoff, and D. Panda. Supporting Efficient Noncontiguous Access in PVFS over InfiniBand. In *Proceedings of International Conference on Cluster Computing (CLUSTER'03)*, 2003.

[26] J. Wu, P. Wyckoff, and D. K. Panda. PVFS over InfiniBand: Design and Performance Evaluation. In *Proceedings of the International Conference on Parallel Processing (ICPP'03)* , pages 125–132, 2003.

[27] W. Yu, S. Liang, and D. K. Panda. High Performance Support of Parallel Virtual File System (PVFS2) over Quadrics. In *Proceedings of International Conference on Supercomputing (ICS'05)*, 2005.

[28] Y. Zhou, A. Bilas, S. Jagannathan, C. Dubnicki, J. F. Philbin, and K. Li. Experiences with VI communication for database storage. In *Proceedings of the 29th annual international symposium on Computer architecture(ISCA'02)*, pages 257–268, 2002.