# Benefits of Quadrics Scatter/Gather to PVFS2 Noncontiguous IO [*]

Weikuan Yu    Dhabaleswar K. Panda

Network-Based Computing Lab
Dept. of Computer Science & Engineering
The Ohio State University
{yuw,panda}@cse.ohio-state.edu

## Abstract

*Noncontiguous IO access is the main access pattern in scientific applications. We have designed an algorithm that supports zero-copy noncontiguous PVFS2 IO using a software scatter/gather mechanism over Quadrics. To investigate what impact Quadrics scatter/gather mechanism can have on PVFS2 IO accesses, in this paper, we perform an in-depth evaluation of the scatter/gather mechanism. We also study how much this mechanism can improve on the performance of scientific applications. Our performance evaluation indicates that Quadrics scatter/gather is beneficial to PVFS2 IO bandwidth. The performance of an application benchmark, MPI-Tile-IO, can be improved by 113% and 66% in terms of aggregated read and write bandwidth, respectively. Moreover, our implementation significantly outperforms an implementation of PVFS2 over InfiniBand, which does not support zero-copy noncontiguous IO.*

[*]This research is supported in part by a DOE grant #DE-FC02-01ER25506 and NSF Grants #CCR-0311542, #CNS-0403342 and #CNS-0509452.

## 1. Introduction

Many parallel file systems, including both commercial packages [9, 11, 6] and research projects [12, 8, 1], have been developed to meet the needs of high IO bandwidth for scientific applications. Among them, Parallel Virtual File System 2 (PVFS2) [1] has been created with the intention to support next generation systems using low cost Linux clusters with commodity components. These parallel file systems make use of network IO to parallelize IO accesses to remote servers. High speed interconnects, such as Myrinet [3] and InfiniBand [10], have been utilized in commodity storage systems to achieve scalable parallel IO. An effective integration of networking mechanisms is important to the performance of these file systems.

While most file systems are optimized for contiguous IO accesses, scientific applications typically make structured IO accesses to the data storage. These accesses consist of a large number of IO operations to many small regions of the data files. The noncontiguity can exist in both the clients' memory layout and a file itself. In view of the importance of native noncontiguous support for file systems, Thakur et. al. [18] have proposed a structured IO interface to describe noncontiguity in both memory and file layout. PVFS2 is designed with a native noncontiguous interface, list IO, to support structured IO accesses.

Quadrics interconnect [14, 2] provides a very low latency ($\leq 2\mu$s) and high bandwidth cluster network. It supports many of the cutting-edge communication features, such as OS-bypass user-level communication, remote direct memory access (RDMA), as well as hardware atomic and collective operations. We have recently demonstrated that Quadrics interconnect can be utilized for high performance support of Parallel Virtual File System 2 (PVFS2) [21]. An efficient Quadrics scatter/gather mechanism is designed to support PVFS2 list IO. However, it is to be investigated how PVFS2 IO accesses can benefit from Quadrics scatter/gather, and what benefits it can provide to scientific applications.

In this paper, we first describe PVFS2 list IO and the noncontiguity that exists in PVFS2 IO accesses. We then evaluate the performance benefits of Quadrics scatter/gather support to PVFS2 IO accesses and scientific applications. Our experimental results indicate that Quadrics scatter/gather is beneficial to the performance of IO accesses in PVFS2. Such zero-copy scatter/gather-based PVFS2 list IO can significantly improve the performance of scientific applications that involve noncontiguous IO accesses. We show that the read and write bandwidth of an application benchmark, MPI-Tile-IO, can be improved by 113% and 66%, respectively. Moreover, our implementation of PVFS2 over Quadrics, with or without scatter/gather support, outperforms an implementation of PVFS2 over InfiniBand.

The rest of the paper is organized as follows. In the next section, we provide an overview of PVFS2 over Quadrics. In Section 3, we describe the design of scatter/gather mechanism over Quadrics to support zero-copy noncontiguous PVFS2 list IO. In Section 4, we provide performance evaluation results. Section 5 gives a brief review of related works. Section 6 concludes the paper.

## 2. Overview of PVFS2 over Quadrics

PVFS2 [1] is the second generation file system from the Parallel Virtual File System (PVFS) project team. It incorporates the design of the original PVFS [13] to provide parallel and aggregated I/O performance. A client/server architecture is designed in PVFS2. Both the server and client side libraries can reside completely in user space. Clients communicate with one of the servers for file data accesses, while the actual file IO is striped across a number of file servers. Metadata accesses can also be distributed across multiple servers. Storage spaces of PVFS2 are managed by and exported from individual servers using native file systems available on the local nodes.

PVFS2 has been ported to various different high speed interconnects, including Myrinet and InfiniBand. Recently, by overcoming the static process model of Quadrics user-level communication [22, 21], PVFS2 over Quadrics has been implemented over the second generation of Quadrics interconnects, QsNet$^{II}$ [2]. This new release provides very low latency, high bandwidth communication with its two building blocks: the Elan-4 network interface and the Elite-4 switch, which are interconnected in a fat-tree topology. On top of its Elan4 network [14], Quadrics provides two user-level communication libraries: `libelan` and `libelan4`. PVFS2 over Quadrics is implemented completely in the user space over these libraries, compliant to the modular design of PVFS2. As shown in Fig. 1, its networking layer, Buffer Message Interface (BMI) [4], is layered on top of the libelan and libelan4 libraries. More details on the design and initial performance evaluation of PVFS2/Quadrics can be found in [21].

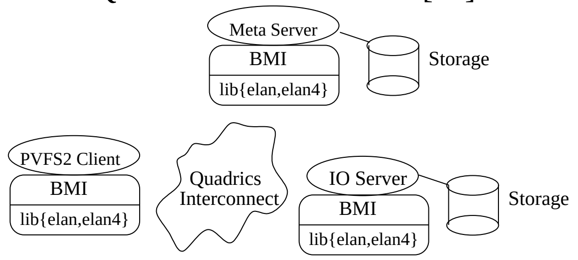

**Fig. 1. PVFS2 over Quadrics Elan4**

## 3. Designing Zero-Copy Quadrics Scatter/Gather for PVFS2 List IO

Noncontiguous IO access is the main access pattern in scientific applications. Thakur et. al. [18] noted that it is important to achieve high performance MPI-IO with native noncontiguous access support in file systems. PVFS2 provides list IO interface to support such noncontiguous IO accesses. Fig. 2 shows an example of noncontiguous IO with PVFS2. In PVFS2 list IO, communication between clients and servers over noncontiguous memory regions are supported over list IO so long as the combined destination memory is larger than the combined source memory. List IO can be built on top of interconnects with native scatter/gather communication support, otherwise, it often resorts to memory packing and unpacking for converting noncontiguous memory fragments to contiguous memory. An alternative is to perform multiple send and receive operations. This can lead to more processing and more communication in small data chunks, resulting in performance degradation.

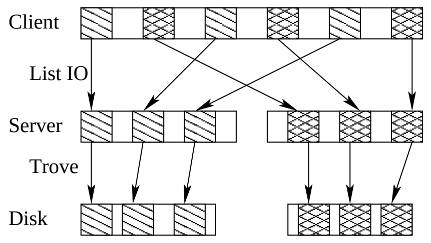

**Fig. 2. An Example of PVFS2 List IO**

There is a unique chain DMA mechanism over Quadrics. In this mechanism, one or more DMA operations can be configured as chained operations with a single NIC-based event. When the event is fired, all the DMA operations will be posted to Quadrics DMA engine. Based on this mechanism, the default Quadrics software release provides noncontiguous communication operations in the form of *elan_putv* and *elan_getv*. However, these operations are specifically designed for the shared memory programming model (SHMEM) over Quadrics. The final placement of the data still requires a memory copy from the global memory to the application destination memory.

To support zero-copy PVFS2 list IO, we propose a software zero-copy scatter/gather mechanism with a single event chained to multiple RDMA operations. Fig. 3 shows a diagram

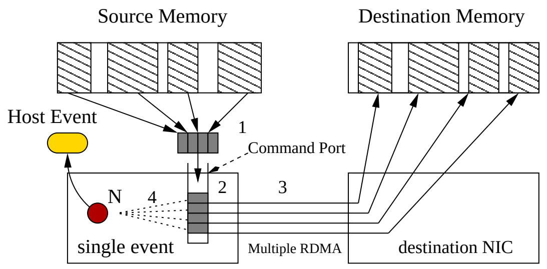

**Fig. 3. Zero-Copy Noncontiguous Communication with RDMA and Chained Event**

about how it can be used to support PVFS2 list IO with RDMA read and/or write. As the communication on either side could be non-contiguous, a message first needs to be exchanged for information about the list of memory address/length pairs. The receiver can decide to fetch all data through RDMA read, or it can inform the sender to push the data using RDMA write. With either RDMA read or write, the number of required contiguous RDMA operations, $N$, needs to be decided first. Then the same number of RDMA descriptors are constructed in the host memory and written together into the Quadrics Elan4 command port (a command queue to the NIC formed by a memory-mapped user accessible NIC memory) through programmed IO. An Elan4 event is created to wait on the completion of $N$ RDMA operations. As this event is triggered, the completion of list IO operation is detected through a host-side event. Over Quadrics, a separate message or an event can be chained to this Elan4 event and notify the remote process. Note that, with this design, multiple RDMA operations are issued without calling extra sender or receiver routines. The data is communicated in a zero-copy fashion, directly from the source memory regions to the destination memory regions.

## 4. Performance Evaluation

In this section, we describe the performance evaluation of zero-copy Quadrics scatter/gather for PVFS2 over Quadrics. The experiments were conducted on a cluster of eight SuperMicro SUPER X5DL8-GG nodes: each with dual Intel Xeon 3.0 GHz processors, 512 KB L2 cache, PCI-X 64-bit 133 MHz bus, 533MHz Front Side Bus (FSB) and a total of 2GB PC2100 DDR-SDRAM physical memory. All eight nodes are connected to a QsNet$^{II}$ network [14, 2], with a dimension one quaternary fat-tree [7] QS-8A switch and eight Elan4 QM-500 cards. The nodes are also connected using the Mellanox InfiniScale 144 port switch. The kernel version used is Linux 2.4.20-8smp. The InfiniBand stack is IBGD-1.7.0 and HCA firmware version is 3.3.2. PVFS2 provides two different modes for its IO servers: *trovesync* and *notrovesync*. The trovesync mode is the default in which IO servers perform *fsync* operations for immediate update of file system metadata changes. The notrovesync mode allows a delay in metadata update. PVFS2 file system can achieve better IO performance when running in this mode. We choose notrovesync exclusively in our experiments to focus on performance benefits of Quadrics scatter/gather.

### 4.1. Benefits of Quadrics Scatter/Gather to PVFS2 IO

To evaluate the performance of PVFS2 IO operations, we have used a parallel program that iteratively performs the following operations: create a new file, concurrently issue multiple write calls to disjoint regions of the file, wait for them to complete, flush the data, concurrently issue multiple read calls for the same data blocks, wait for them to complete, and then remove the file. MPI collective operations are used to synchronize application processes before and after each IO operation. In our program, based on its rank in the MPI job, each process reads or writes various blocks of data, 1MB each, at disjoint offsets of a common file.

We have divided the eight-node cluster into two groups: servers and clients. Up to four nodes are configured as PVFS2 servers and the remaining nodes are running as clients. Experimental results are labeled as $N$S for a configuration with N servers. Basic PVFS2 list IO is supported through memory packing and unpacking, marked as LIO in the figures.

There are little differences with different number of blocks. The performance results are obtained with using the total IO size of 4MB for each process. The aggregated read and write bandwidth can be increased by up to 89% and 52%, as shown by Figures 4 and 5, respectively. These results indicate that zero-copy scatter/gather support is indeed beneficial to both read and write operations. Note that the improvement on the read bandwidth is better than the write bandwidth. This is because the read operations in PVFS2 is shorter and less complex compared to write operations, thus can benefit more by optimizations on data communication across the network.

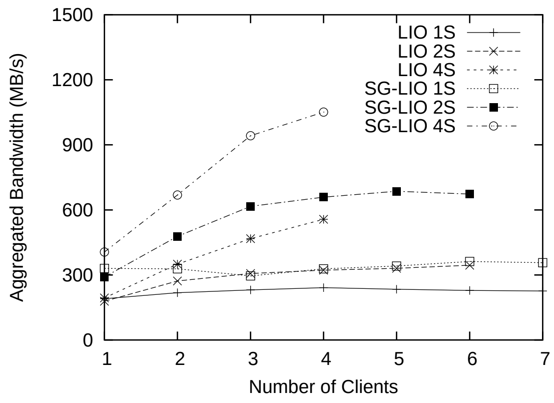

**Fig. 4. Performance Comparisons of PVFS2 Concurrent Read**

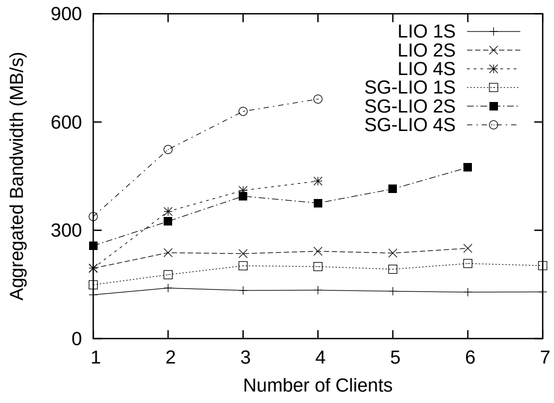

**Fig. 5. Performance Comparisons of PVFS2 Concurrent Write**

## 4.2. Benefits of Quadrics Scatter/Gather to MPI-Tile-IO

MPI-Tile-IO [16] is a tile reading MPI-IO application. It tests the performance of tiled access to a two-dimensional dense dataset, simulating the type of workload that exists in some visualization and numerical applications. Four of eight nodes are used as server nodes and the other four as client nodes running MPI-Tile-IO processes. Each process renders a $2 \times 2$ array of displays, each with $1024 \times 768$ pixels. The size of each element is 32 bytes, leading to a file size of 96MB.

We have evaluated both the read and write performance of MPI-Tile-IO over PVFS2. As shown in Fig. 6, with Quadrics scatter/gather support, MPI-Tile-IO write bandwidth can be improved by 66%. On the other hand, MPI-Tile-IO read bandwidth can be improved by up to 113%. These results indicate our implementation is able to leverage the performance benefits of Quadrics scatter/gather into PVFS2. In addition, using MPI-Tile-IO, we have compared the performance of PVFS2 over Quadrics with an implementation of PVFS2 over InfiniBand from the release PVFS2-1.1.0. MPI-Tile-IO achieves less aggregated read and write bandwidth over InfiniBand (labeled as IB-LIO), compared to our Quadrics-based PVFS2 implementation, with or without zero-copy scatter/gather. This is because of two reasons. First, memory registration is needed over InfiniBand for communication and the current implementation of PVFS2 over InfiniBand does not take advantage of memory registration cache to save registration costs. In contrast, memory registration is not needed over Quadrics with its NIC-based MMU. The other reason is that PVFS2/IB utilizes InfiniBand RDMA write-gather and read-scatter mechanisms for non-contiguous IO. These RDMA-based scatter/gather mechanisms over InfiniBand can only avoid one local memory copy. On the remote side, memory packing/unpacking is still needed. So it does not support true zero-copy list IO for PVFS2.

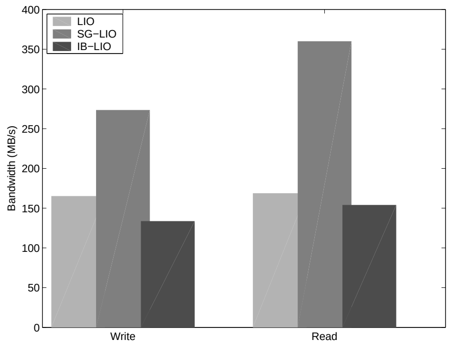

**Fig. 6. Performance of MPI-Tile-IO Benchmark**

## 5. Related Work

Optimizing noncontiguous IO has been the topic of various research, covering a spectrum of IO subsystems. For example, data sieving [17] and two-phase IO [15] have been leveraged to improve the client IO accesses of parallel applications in the parallel IO libraries, such as MPI-IO. Wong et. al. [19] optimized the linux vector IO by allowing IO vectors to be processed in a batched manner.

Ching et. al [5] have implemented list IO in PVFS1 to support efficient noncontiguous IO and evaluated its performance over TCP/IP. Wu et. al [20] have studied the benefits of leveraging InfiniBand hardware scatter/gather operations to optimize noncontiguous IO access in PVFS1. In [20], a scheme called Optimistic Group Registration is also proposed to reduce costs of registration and deregistration needed for communication over InfiniBand fabric. Our work exploits a communication mechanism with a single event chained to multiple RDMA to support zero-copy non-contiguous network IO in PVFS2 over Quadrics. Since communication over Quadrics does not require memory registration, we do not have to be concerned with these issues in our implementation.

## 6. Conclusions

In this paper, we have described PVFS2 list IO and a software scatter/gather mechanism over Quadrics to support zero-copy list IO. We then evaluate the performance benefits of this support to PVFS2 IO accesses and a scientific application, MPI-Tile-IO. Our experimental results indicate that Quadrics scatter/gather support is beneficial to the performance of PVFS2 IO accesses. In terms of aggregated read and write bandwidth, the performance of MPI-Tile-IO application benchmark can be improved by 113% and 66%, respectively, Moreover, our implementation of PVFS2 over Quadrics significantly outperforms an implementation of PVFS2 over InfiniBand.

In future, we intend to leverage more features of Quadrics to support PVFS2 and study their possible benefits to different aspects of parallel file systems. For example, we intend to study the benefits of integrating Quadrics NIC memory into PVFS2 memory hierarchy, such as data caching with client and/or server-side NIC memory. We also intend to study the feasibility of directed IO over PVFS2/Quadrics using the Quadrics programmable NIC as the processing unit.

## References

[1] The Parallel Virtual File System, version 2. http://www.pvfs.org/pvfs2.

[2] J. Beecroft, D. Addison, F. Petrini, and M. McLaren. QsNet-II: An Interconnect for Supercomputing Applications. In *the Proceedings of Hot Chips '03*, Stanford, CA, August 2003.

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

[4] P. H. Carns, W. B. Ligon III, R. Ross, and P. Wyckoff. BMI: A Network Abstraction Layer for Parallel I/O, April 2005.

[5] A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp. Noncontiguous I/O through PVFS. In *Proceedings of the IEEE International Conference on Cluster Computing*, Chicago, IL, September 2002.

[6] A. M. David Nagle, Denis Serenyi. The Panasas ActiveScale Storage Cluster – Delivering Scalable High Bandwidth Storage. In *Proceedings of Supercomputing '04*, November 2004.

[7] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. The IEEE Computer Society Press, 1997.

[8] J. Huber, C. L. Elford, D. A. Reed, A. A. Chien, and D. S. Blumenthal. PPFS: A High Performance Portable Parallel File System. In *Proceedings of the 9th ACM International Conference on Supercomputing*, pages 385–394, Barcelona, Spain, July 1995. ACM Press.

[9] IBM Corp. IBM AIX Parallel I/O File System: Installation, Administration, and Use. Document Number SH34-6065-01, August 1995.

[10] Infiniband Trade Association. http://www.infinibandta.org.

[11] Intel Scalable Systems Division. Paragon System User's Guide, May 1995.

[12] N. Nieuwejaar and D. Kotz. The Galley Parallel File System. *Parallel Computing*, (4):447–476, June 1997.

[13] P. H. Carns and W. B. Ligon III and R. B. Ross and R. Thakur. PVFS: A Parallel File System For Linux Clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, October 2000.

[14] Quadrics, Inc. Quadrics Linux Cluster Documentation.

[15] J. Rosario, R. Bordawekar, and A. Choudhary. Improved parallel i/o via a two-phase run-time access strategy. In *Workshop on Input/Output in Parallel Computer Systems, IPPS '93*, Newport Beach, CA, 1993.

[16] R. B. Ross. Parallel i/o benchmarking consortium. http://www-unix.mcs.anl.gov/rross/pio-benchmark/html/.

[17] R. Thakur, A. Choudhary, R. Bordawekar, S. More, and S. Kuditipudi. Passion: Optimized I/O for Parallel Applications. *IEEE Computer*, (6):70–78, June 1996.

[18] R. Thakur, W. Gropp, and E. Lusk. On Implementing MPI-IO Portably and with High Performance. In *Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems*, pages 23–32. ACM Press, May 1999.

[19] P. Wong, B. Pulavarty, S. Nagar, J. Morgan, J. Lahr, B. Hartner, H. Franke, and S. Bhattacharya. Improving linux block i/o layer for enterprise workload. In *Proceedings of The Ottawa Linux Symposium*, 2002.

[20] J. Wu, P. Wychoff, and D. K. Panda. Supporting Efficient Noncontiguous Access in PVFS over InfiniBand. In *Proceedings of Cluster Computing '03*, Hong Kong, December 2004.

[21] W. Yu, S. Liang, and D. K. Panda. High Performance Support of Parallel Virtual File System (PVFS2) over Quadrics. In *Proceedings of The 19th ACM International Conference on Supercomputing*, Boston, Massachusetts, June 2005.

[22] W. Yu, T. S. Woodall, R. L. Graham, and D. K. Panda. Design and Implementation of Open MPI over Quadrics/Elan4. In *Proceedings of the International Conference on Parallel and Distributed Processing Symposium '05*, Colorado, Denver, April 2005.