Network-Based Data Processing Architecture for Reliable and High-Performance Distributed Storage System

Hiroki Ohtsuji[1,2,3(✉)] and Osamu Tatebe[1,2]

[1] University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
ohtsuji@hpcs.cs.tsukuba.ac.jp
[2] JST, CREST, Kawaguchi, Japan
[3] JSPS Research Fellow, Chiyoda, Japan

Abstract. In the era of post peta scale computing, high-performance and reliable storage systems have become much more important. Close cooperation between network and storage is an emerging issue. This paper proposes a network-based data processing architecture to build reliable and high-performance distributed storage system using future programmable network devices. Distributed storage systems use replication or erasure coding for ensuring reliability. However, they require additional data transfer and computing resources. Satisfying both reliability and performance is an important issue for storage systems. Recent studies related to Software Defined Networking (SDN) imply that programmable network switch will become more functional. Currently, SDN intends to provide a flexible routing mechanism. Network switches are starting to have intelligent mechanisms and are expected to have a capability for data processing. In our proposed architecture, storage controller functionality is offloaded to a programmable network switch to eliminate additional data transfer. We conducted experiments to show an advantage of the proposed network-based data processing mechanisms for erasure coding and show an optimized design for distributed storage systems. With the proposed method, the performance gain of a reliable data storage system is 44 % compared with a client compute case.

1 Introduction

1.1 Background

Next generation distributed storage systems have to meet the demands of exascale computing systems. In particular, high-performance and reliable data handling mechanisms are critical problems. In order to add reliability to network storage systems, replication [1] and erasure coding are commonly used. However, when a writer node stores data to a storage system, amount of traffic from the writer node increases by the additional data. This additional data degrades the performance because of the bandwidth limitation of a client node. Figure 1 describes how the parity blocks increase the traffic and cause the performance degradation. In this case, a parity block is added to striped blocks.

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 16–26, 2015.
DOI: 10.1007/978-3-319-27308-2_2

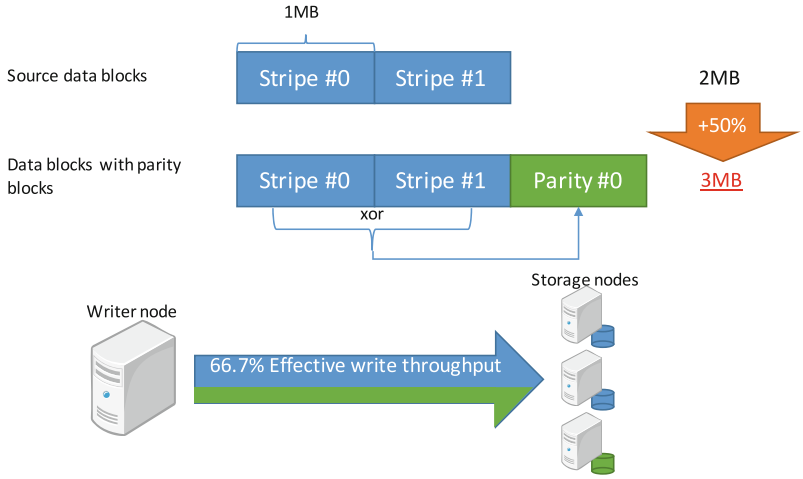

Fig. 1. Parity blocks increase traffic and degrade write throughput.

It causes 50 % traffic increase and 33 % degradation of write throughput. In order to avoid this performance degradation, we need additional mechanisms to eliminate performance bottlenecks.

The target of this paper is optimization of network storage systems which use erasure coding. The optimization utilizes the functions of a programmable network switch.

Existing systems provide reliability through use of the computational power of storage or client nodes. However, as typified by Software Defined Networking (SDN), network hardware has started to shift toward programmable devices. This movement suggests the possibility of implementing a data processing mechanism on the storage network. This study assumes future network devices with programmable functions and proposes a method for utilizing them.

1.2 Our Contribution

This study proposes an architecture design for utilizing data processing mechanism on the network in order to improve the performance of reliable distributed storage systems. The architecture off-loads parity generation processes to a network switch to eliminate the overhead of reliable storage systems. Our proposal includes a design of network storage system and a methodology for utilizing programmable network switches.

The proposed method achieves zero-overhead with erasure-codes generation whereas existing systems degrade performance because of additional data transfer and computing. In preparation for next-generation programmable network devices, this paper shows an efficient way for using them.

Write performance reaches to 5,548 MB/s with redundant data, which is almost same as the network throughput. Performance gain is 14 % to 44 %.

Although we added additional data blocks for reliability, performance does not change. This means that our proposed method realizes the "zero-overhead" reliable network storage system.

2 Related Work

The main focus of our study is a methodology for utilizing programmable network switches to eliminate bottlenecks from reliable network storage systems.

There are existing storage/file systems which use replication and erasure coding in order to improve reliability. RAID [2,3] is a well-known example of reliable storage systems. However, ordinary RAID systems cover only disk failures and cannot recover from node-level failures. In contrast, our proposal intends to provide node-level redundancy for network storage systems. GlusterFS [4], Gfarm [5], Ceph [6], and HDFS [7,8] are network storage/file systems and have a replication mechanism. However, as mentioned before, replication uses twice or more storage space and should be avoided with exa-scale systems. Ceph and HDFS also support Reed-Solomon [9] based erasure coding. HDFS supports erasure coding using the HDFS-RAID [10] module. The most important thing is that all of them does not support "on-the-fly" replication/encoding because of performance issues. Our proposed method generates parity block (erasure code) on an "on-the-fly" basis, hence they are different from our work.

In order to build a zero-overhead reliable storage system, we propose a new architecture which utilizes programmable abilities of a network switch. There are several studies related to this issue that are not only limited to the optimization of storage systems. [11] is a study to optimize MPI collective operations using network switches equipped with FPGA, which utilized NetFPGA [12] and OpenFlow switches and improved the performance of MPI operations. The optimization target is not the same as our work; however, the idea of improving network communication performance using specialized hardware functions is common.

From the perspective of existing network devices, Mellanox provides the function [13] for optimizing MPI communication operations. The target of this hardware is to optimize MPI operations; however, this is only an example of an HPC communication layer accelerated by hardware. Hardware functions that optimize the network of storage systems are in extension of this type of idea. This is the reason we expect that having hardware functions to optimize erasure codes in network storage systems.

In addition, our prototype implementation uses Remote Direct Memory Access (RDMA) to minimize the overhead of network communication by eliminating unnecessary memory copies. Advantages of applying RDMA communication to network storage systems are shown in existing studies. NFS over RDMA [14] is an example of adding RDMA support to NFS [15]. [16] shows performance gain by adding RDMA support to PVFS [17].

3 System Design

3.1 Network-Based Data Processing Architecture

Network-based data processing architecture moves parity generation processes from storage servers to programmable switches. This paper describes a design for data processing architecture for parity generation processes and shows a prototype implementation.

Our target is not a dedicated hardware based large-scale block storage device but a system which consists of multiple storage servers. Conventional network storage systems use computing resources of servers to provide mechanisms for reliability. In that type of system, network only transfers the data between storage servers.

As discussed in Sect. 1.1, bottlenecks come from the reliability issues are owing to the increased amount of data and the limitation of network bandwidth. Utilizing programmable abilities of network switches is a good solution to solve these problems because network switches have enough bandwidth to spread the increased data. At this time, we do not have a network switch (in production and not an FPGA based devices) that has programmable function to implement a mechanism for erasure coding. However, we can propose a method for utilizing the ability of future network switches for reliable storage systems and provide evaluation results with a proof of concept system. The proof of concept system consists of computing nodes with multiple network devices. Following sections describe the proof of concept system of the network-based data processing architecture and the method for reliable storage systems.

3.2 Overview of the System

This study targets network storage systems with parity (erasure coding) data. The aim of this paper is to propose a method to utilize network data processing functions and to show evaluation results of the proof of concept system.

Figure 2 describes the architecture of the target system. A writer node sends the data blocks (Stripes #0 and Stripes #1) to a network switch. This switch has programmable functions and calculates a parity block from Stripe #0 and Stripe #1. The switch sends stripe and parity blocks to storage nodes.

3.3 Data Layout

Figure 3 describes the data layout of the target system. In this figure, the original data blocks are #0 to #5. Two storage nodes store those data blocks and another stores the exclusive OR (XOR) value (parity) of each original data block.

3.4 Switch Architeture

Figure 4 shows how network-based data processing architecture works. The writer node sends data blocks to the switch, which then splits data blocks into striped blocks and calculates their XOR value. Each parity block is sent to storage nodes for striped blocks and a parity node stores the parity blocks.

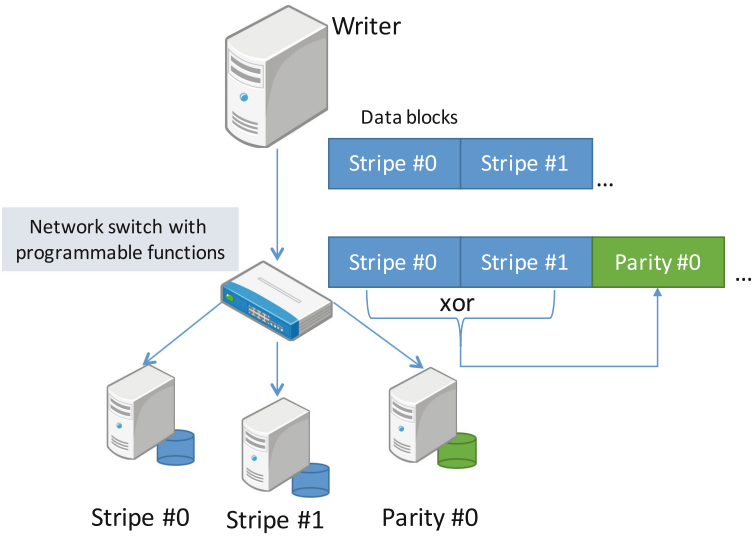

Fig. 2. Target architecture of network storage systems. Our proposal is a method for utilizing a programmable network switch for erasure-coded network storage systems. A writer node sends source data blocks to a programmable switch. The switch generates parity blocks and sends them to storage nodes.

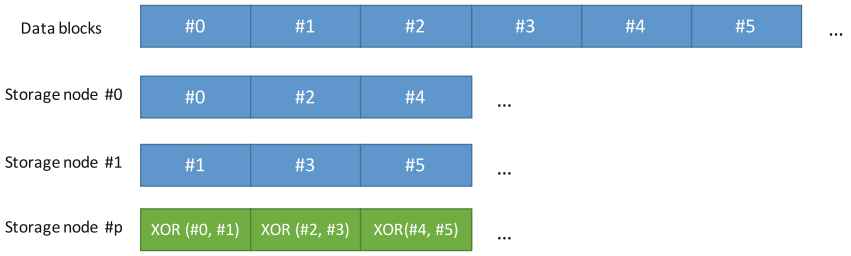

Fig. 3. Data layout of striped blocks and parity block(s). Data blocks are split into striped blocks. Parity blocks are xor value of striped blocks. We can recover missing data blocks by calculating xor value of another block.

3.5 Fallback Mode

Current our experiment environment is only for prototype purpose. However, when we apply the method to a large-scale environment, switch failures become a major trouble.

The system should have a fallback mode in preparation for switch failures. Figure 5 describes the fallback mode of a storage system. If the switch loses programmable functions (left in the figure), the writer node can split the data blocks into striped blocks and calculate the parity blocks. Next, the writer node sends all blocks to storage nodes. In the case of complete network failure (right

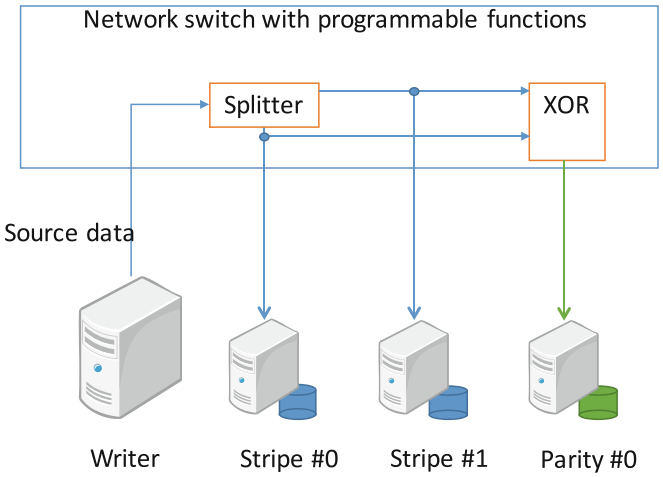

Fig. 4. Architecture of a programmable network switch. A writer sends source data blocks. A switch has a splitter and an XOR calculation module. The splitter splits the source data blocks into striped blocks. The XOR calculation module calculates the XOR value of the striped blocks. Then, the switch sends all data blocks to storage servers.

in the figure), another network mechanism is required. If there is an available network path, the writer can use it to apply the fallback mode.

3.6 Prototype Implementation Overview

Currently, we do not have actual hardware for the programmable network switch; therefore we implemented it with a computer and multiple network cards. Figure 6 describes the connection of each component.

The node has multiple network cards (in this figure, network cards are InfiniBand HCAs). Because of the limitation of the number of PCI Express lanes, we added three InfiniBand HCAs to the node. Figure 7 describes the data transfer mechanism and parity generation process. We utilized the RDMA function of the InfiniBand HCAs to optimize the data transfer processes and save memory space. The data structure and data processing mechanism are described in the next subsection.

3.7 Optimized Data Transfer and Processing with RDMA

Figure 7 describes the zero-copy data structure of the data processing mechanism. Each node has ring buffer(s) to transfer and process the data blocks. In order to use the RDMA data transfer functions, we have to register the memory to the hardware in advance to the actual transfer process. If we use different buffer blocks, each time RDMA transfer occurred, we would have to register it. However, this

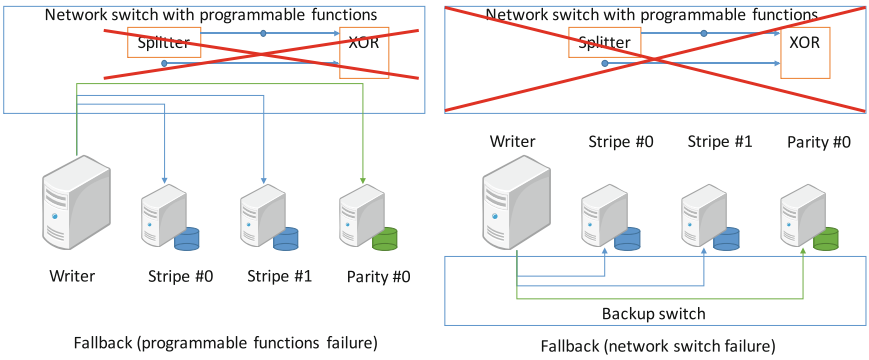

Fig. 5. Two cases of failure of a network switch.

requires considerable time and causes performance degradation [18]. Therefore, we use ring buffer(s) for RDMA communication. Once the ring buffer is registered, we do not require time for memory registration processes. In Fig. 7 the writer nodes send data blocks to the switch. The switch splits the data into two striped blocks and stores them to ring buffers. Then, the switch sends the striped blocks to storage nodes and calculates the XOR value of these two blocks. Finally, the XOR value is sent to the storage node (p).

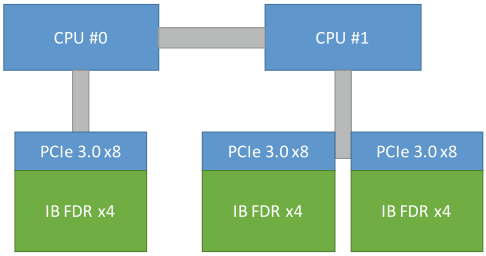

Fig. 6. Components of computer with multiple network devices

4 Evaluation

4.1 Evaluation Target and Conditions

Our proposed method intends to optimize the performance of data write to reliable storage systems.

We conducted an evaluation on the cluster nodes connected with InfiniBand FDR 4x. To implement network-based data processing architecture, we used a node equipped with multiple InfiniBand HCAs. In this evaluation, we installed

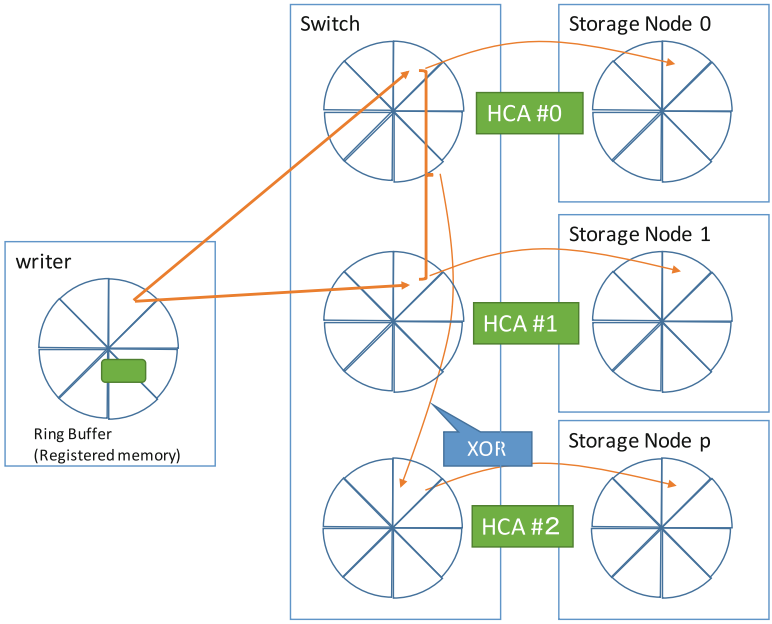

Fig. 7. RDMA data transfer and parity generation. A writer send source data blocks to a programmable switch. The switch splits source data blocks into striped blocks (not described in the figure) and calculates parity blocks. Afterward, the switch sends them to storage nodes using different InfiniBand HCAs. All data blocks are stored in ring buffers and there is no memory copy.

up to three InfiniBand HCAs to a computing node. First, we evaluated the throughput of the multiple InfiniBand HCAs to confirm that the test environment did not have any performance bottlenecks resulting from hardware specifications. Figure 8 shows the results of the total bidirectional bandwidth evaluation. We used perftest tools (ib_write_bw) to evaluate network performance. As can be seen from the graph, the results are in proportion to the number of installed HCAs. We do not see any performance degradation caused by limitation of the PCI Express bus or other interconnect issues.

In addition, we conducted the evaluation without writing to the actual disk because of the limitation of existing storage hardware and the purpose of the evaluation. The aim of the proposed method is optimization of the data transfer mechanism.

All results are average of three measurements.

4.2 Evaluation Results

Figure 9 shows the results of the sequential write throughput evaluation. The X-axis corresponds to stripe size and the Y-axis corresponds to the write throughput from a client node. The blue graph is the result of optimization (using data

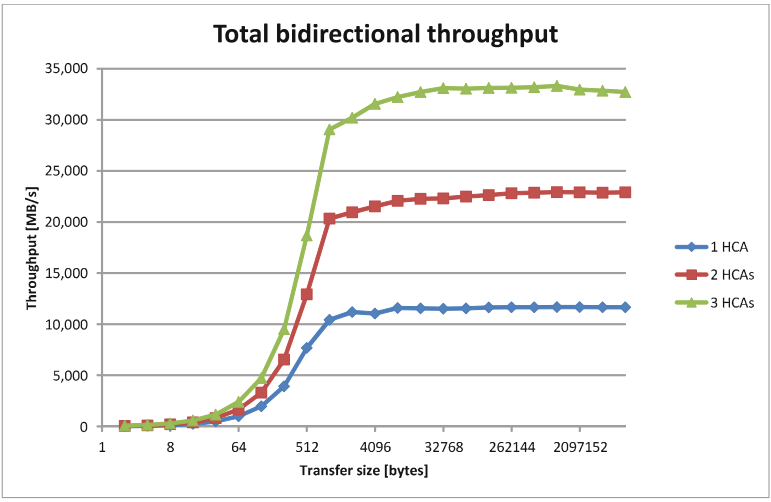

Fig. 8. Bidirectional network (InfiniBand FDR 4x) throughput evaluation with ib_write_bw (perftest tools).

processing architecture of a network switch) and the red graph is the result of naive implementation (the client sends both striped and parity blocks). Performance gain by the optimization was 14 % (8 KB stripe case) to 44 % (64 KB stripe case). With the best case of the optimized implementation, the throughput almost reaches network performance. This means that the proposed method successfully eliminated the bottleneck of the reliable data write. The performance of the fallback mode Sect. 3.5 will be the same as the results of the naive method, provided that the system has the same back up network.

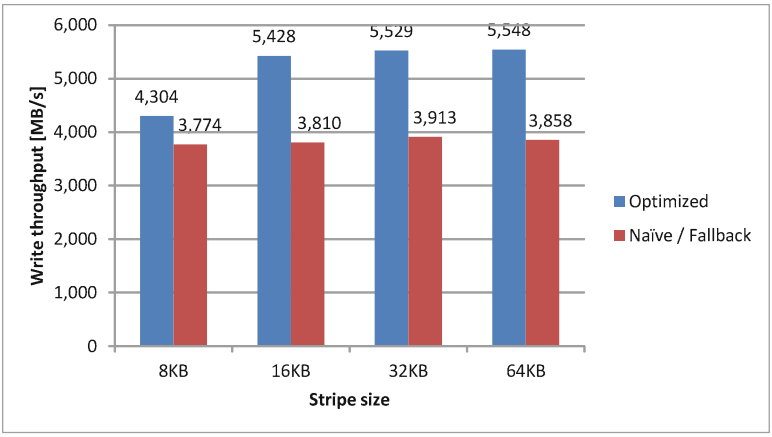

Fig. 9. Write throughput comparison between optimized and naive method

The evaluation results show that the proposed method improves the performance of reliable data write by eliminating the bottleneck. In this evaluation, we implemented the data processing architecture using the node equipped with multiple network devices. However, the method consists of data processing (XOR) pipelines and can be implemented as hardware. This means that immediately after obtaining a network switch with programmable functions, we can implement the proposed method and improve the performance of reliable network storage systems.

5 Conclusion and Future Work

We proposed a methodology for utilizing programmable network switches to eliminate the overhead of reliable network storage systems. Root cause of the performance degradation of reliable network storage system was the increased amount of traffic by additional parity data blocks. Our proposed design moves the parity generation processes to a programmable network switch in order to avoid congestion in a writer node's network. We implemented a prototype system using RDMA transfer operations and conducted evaluations.

The evaluation results showed that the performance gain by the proposed method was 14 % to 44 %.

Currently, we implemented the proposed method using a computer equipped with multiple network devices. However, the method can be implemented as a hardware and can be applied to future programmable network switches.

Applying the method to existing systems and adding support of random/stride write are important issues. In addition, a scalability issue is an important future work when we apply the method to huge systems because this study used a preliminary evaluation environment, which had a single network switch.

The target for the evaluation in this paper was the sequential write to storage systems; however, the results showed good performance in case of 16 KB to 64 KB stripe block size and thus we can expect good performance with real work loads.

Acknowledgemtent. This works is supported by JST CREST "System Software for Post Petascale Data Intensive Science", JST CREST "Extreme Big Data (EBD) Next Generation Big Data Infrastructure Technologies Towards Yottabyte/Year", and JSPS KAKENHI Grant-in-Aid for JSPS Fellows (261967).

References

1. Chervenak, A.L., Foster, I.T., Kesselman, C., Salisbury, C., Tuecke, S.: The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. J. Netw. Comput. Appl. **23**, 187–200 (1999)
2. Patterson, D.A., Gibson, G., Katz, R.H.: A case for redundant arrays of inexpensive disks (RAID). SIGMOD Rec. **17**, 109–116 (1988)
3. Chen, P.M., Lee, E.K., Gibson, G.A., Katz, R.H., Patterson, D.A.: RAID: high-performance, reliable secondary storage. ACM Comput. Surv. **26**, 145–185 (1994)

4. RedHat: Gluster FS. http://www.gluster.org/
5. Tatebe, O., Hiraga, K., Soda, N.: New generation computing. Gfarm Grid File System **28**, 257–275 (2010). Ohmsha Ltd. and Springer
6. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph: a scalable, high-performance distributed file system. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, OSDI 2006, pp. 307–320. USENIX Association, Berkeley (2006)
7. Hadoop. http://hadoop.apache.org/
8. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST 2010, pp. 1–10. IEEE Computer Society, Washington, D.C. (2010)
9. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. **8**, 300–304 (1960)
10. Fan, B., Tantisiriroj, W., Xiao, L., Gibson, G.: Diskreduce: RAID for data-intensive scalable computing. In: Proceedings of the 4th Annual Workshop on Petascale Data Storage, PDSW 2009, pp. 6–10. ACM, New York (2009)
11. Arap, O., Brown, G., Himebaugh, B., Swany, M.: Software defined multicasting for MPI collective operation offloading with the NetFPGA. In: Silva, F., Dutra, I., Santos Costa, V. (eds.) Euro-Par 2014 Parallel Processing. LNCS, vol. 8632, pp. 632–643. Springer, Heidelberg (2014)
12. Lockwood, J., McKeown, N., Watson, G., Gibb, G., Hartke, P., Naous, J., Raghuraman, R., Luo, J.: NetFPGA-an open platform for gigabit-rate network switching and routing. In: IEEE International Conference on Microelectronic Systems Education, MSE 2007, pp. 160–161 (2007)
13. Mellanox: CORE-Direct The Most Advanced Technology for MPI/SHMEM Collectives Offloads. http://www.mellanox.com/related-docs/whitepapers/TB_CORE-Direct.pdf
14. Callaghan, B., Lingutla-Raj, T., Chiu, A., Staubach, P., Asad, O.: NFS over RDMA. In: Proceedings of the ACM SIGCOMM Workshop on Network-I/O Convergence: Experience, Lessons, Implications, NICELI 2003, pp. 196–208. ACM, New York (2003)
15. Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., Noveck, D.: Network File System (NFS) version 4 Protocol. RFC 3530 (Proposed Standard) (2003)
16. Wu, J., Wyckoff, P., Panda, D.: PVFS over infiniband: design and performance evaluation (2003)
17. Carns, P.H., Iii, Ross, R.B., Thakur, R.: PVFS: a parallel file system for linux clusters. In: Proceedings of the 4th Annual Linux Showcase and Conference, ALS 2000 (2000)
18. Dalessandro, D., Wyckoff, P.: Memory management strategies for data serving with RDMA. In: 15th Annual IEEE Symposium on High-Performance Interconnects, HOTI 2007, pp. 135–142 (2007)