2009 International Conference on Computational Science and Its Applications

Design and Implementation of Networking Virtualization for Cluster File Systems

Junghan Kim, Inhyuk Kim, Taehyoung Kim, Young Ik Eom
School of Information and Communication Engineering
Sungkyunkwan University
Suwon, Korea
{gtgkjh, kkojiband, kim15m, yieom}@ece.skku.ac.kr

Hong-Yeon Kim, Youngkyun Kim
Storage System Research Team, Internet Platform Research Department, S/W & Content Research Lab.
Electronics and Telecommunications Research Institute
Daejeon, Korea
{kimhy, kimyoung}@etri.re.kr

***Abstract*—An effective network layer which reduces cluster file system complexity is definitely needed in cluster computing environments where more than thousands of nodes are connected on a variety of different networks. Therefore, widely used cluster file systems such as Lustre, PVFS2, and GlusterFS provide networking virtualization layer. Through networking virtualization layer, each node accesses other nodes using a single interface over a variety of networks. However, previous networking virtualization technologies are complicated and not portable. In this paper, we propose a simple, portable, and socket-like networking virtualization layer that operates on user-mode. We also evaluate our virtualization layer using benchmark tools**

Keywords-Cluster file system; Storage networking; RDMA; Virtualization;

I. INTRODUCTION

According to rapid growth of Internet services such as blog and User Created Contents (UCC), a large amount of multimedia contents put more burden on Internet infrastructure. Especially, cluster file systems often carry a heavy workload. In order to provide fast search and long term storage, several requirements such as stability, availability, low cost, and scalability are needed for cluster file systems. One of the solutions to these requirements is to improve the network performance.

On existing cluster file systems such as Lustre, PVFS2, and GlusterFS, the network performance is considered as the most important factor for normal operation [1,2,3]. In case of Gigabit Ethernet, there is high overhead to handle TCP/IP software stack. TCP Offload Engine (TOE) was suggested to reduce the overhead [4]. TOE implemented TCP/IP stack on hardware to alleviate the TCP/IP overhead. But it still has a low scalability problem. Also some of the file systems provide network virtualization layer to support a variety of different networks. But, these network virtualization layers are too complicated and not portable.

Our goal is to design a network virtualization layer in the cluster file system, which eliminates dependency on network protocols, for applications to operate without any modifications.

In this paper, we propose a networking virtualization layer on the simple network file system based on the FUSE [5]. Our scheme supports a single interface and connection pool, and also enables both connection-oriented and connectionless interconnections. Furthermore, our scheme has no kernel dependency. We evaluate our virtualization layer using benchmark tools (Unixbench, SQLite) [6,7].

The remainder of this paper is organized as follows: In Section 2, we examine the existing cluster file systems and the storage networking schemes. Section 3 introduces our proposed design and implementation of the virtualization layer. Then, in Section 4, we evaluate our networking virtualization layer and show the evaluation results. Finally, in Section 5, we present our conclusion.

II. BACKGROUND

A. Cluster file system

Networking virtualization layer is shown in Figure 1.

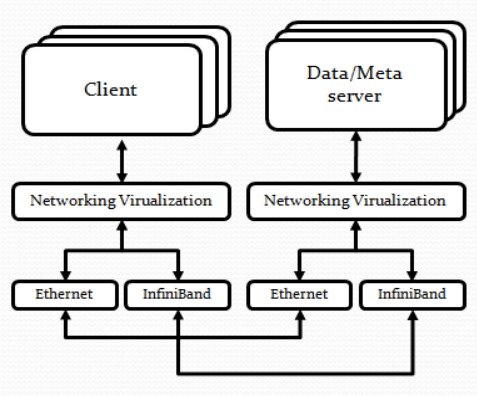

Figure 1. Networking virtualization

A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer [8]. The components of a cluster are generally connected to each other over fast networks. Because a computer cluster runs on the networks such as Ethernet and InfiniBand, a cluster file system has to support these network

978-0-7695-3701-6/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCSA.2009.20

protocols. So cluster file system includes networking virtualization to unify various network protocols as a single network. While clients and servers are working each other, file system clients must access servers using native protocols over a variety of networks. And so, networking virtualization is very important feature for the cluster file systems.

1) Lustre

Lustre is an open source, object-based, cluster file system, generally used for large scale cluster computing environments [1]. The Lustre aims to provide a file system for clusters of tens of thousands of nodes with petabytes of storage capacity. Lustre is designed, developed, and maintained by Sun Microsystems, Inc. Lustre provides networking virtualization layer called LNET(Lustre NETworking) to support many network types including InfiniBand, Gigabit Ethernet, Quadrics, and Myrinet.

2) PVFS2(Parallel Virtual File System 2)

PVFS2 is a file system designed from the Parallel Virtual File System project [2]. PVFS2 is an open source cluster file system. PVFS was designed for use in large scale cluster computing environments. It consists of a server process and a client library, both of which are written entirely in user-level code. The client library provides for high performance access via the MPI(message passing interface). PVFS2 provides networking virtualization layer called BMI(Buffered Message Interface), which is designed to meet various network protocols such as InfiniBand, Gigabit Ethernet, and Myrinet.

3) GlusterFS

GlusterFS is an open source cluster file system, capable of scaling to several petabytes of storage [3]. GlusterFS package comes with two components, a server and a client. To mount GlusterFS file systems, the client computers need FUSE supported in the kernel [5]. By using FUSE, no kernel patches are required, and therefore the software can be installed without any server downtime. GlusterFS server operates on Linux, FreeBSD and Solaris, and client runs only on Linux machines. GlusterFS also supports many commonly used network types, such as InfiniBand and Gigabit Ethernet.

Networking virtualization layers in the three cluster file systems are shown in Figure 2. Each cluster file system has different internal architecture, but they have their own networking virtualization layer for identical purpose.

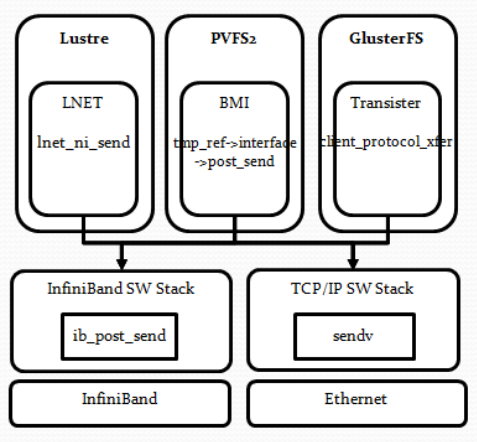

Figure 2. Networking virtualization layers in the three cluster file systems

B. Storage Network

The Gigabit Ethernet and InfiniBand are representative storage networking technologies which support high bandwidth, high scalability, increased stability, low cost, and so on. In this section, we show the characteristics of the two storage networking technologies.

1) Gigabit Ethernet

Gigabit Ethernet offers 1Gbps bandwidth, transfers large amounts of data quickly, and supports full compatibility with the 10/100Mbps Ethernet standard [9]. Also, increased bandwidth and performance enable to eliminate bottlenecks in backbone and data center, to resolve many inherent problems of the network structure. Gigabit Ethernet supports the protocols such as TCP, UDP, SCTP, and DCCP.

Transmission Control Protocol (TCP) provides reliable, ordered delivery between two programs. In order to maintain reliable connection, TCP offers flow control, error control, and congestion control. Through the handshaking, connection between two nodes is established. Using IP, TCP delivers a large amount of data based on a stream of bytes. However several services of TCP restrict the transmission performance. Unlike TCP, User Datagram Protocol (UDP) considers only high performance without guarantee of reliable and ordered link.

The Stream Control Transmission Protocol (SCTP) includes characteristics of both TCP and UDP. With message-oriented feature, SCTP ensures reliability and ordered transport with congestion control. Furthermore, SCTP provides multi-streaming resolving a head-of-line blocking problem of TCP and multi-homing enabling transparent failover.

Data Congestion Control Protocol (DCCP) adds a function of congestion control to UDP characteristic for enhancing reliability. DCCP sets up connection through three-way handshaking like TCP, and checks the connection using

acknowledges even though DCCP format is unreliable datagram like UDP.

2) *InfiniBand*

InfiniBand is a new architecture for server I/O and inter-server communication. It is designed to provide the levels of reliability, performance and scalability around a point-to-point, switched I/O fabrics [10,11]. Structure of InfiniBand software is shown in Figure 3.

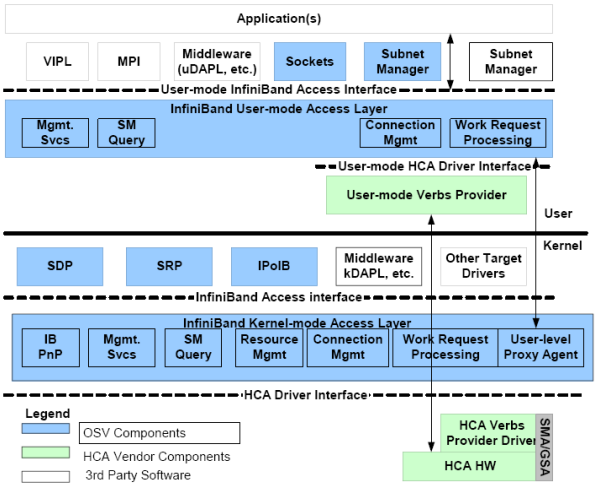

Figure 3. Structure of InfiniBand software [10]

IPoIB provides IP encapsulation over InfiniBand. Data transfer is performed using the unreliable datagram service of InfiniBand. And the IPoIB driver serves as a link layer driver performing address resolution. Therefore traditional upper layer protocols such as TCP/IP are enabled to operate over InfiniBand.

The Socket Direct Protocol (SDP) is a byte-stream transport protocol [11]. InfiniBand supports SDP serving socket communication over InfiniBand. SDP aims to accomplish protocol offload, kernel bypass, zero copy using the supporting capacities of InfiniBand such as remote DMA (RDMA).

Verbs defines abstractions of the mandatory functions to manage and operate the channel adaptor of InfiniBand. Verbs API (VAPI) specifies framework of Verbs following by vendors.

III. DESIGN AND IMPLEMENTATION

Our network virtualization layer is currently implemented only for the Linux environment. But, because our scheme removes kernel dependency, it is compatible with other Unix-like OS environments. In this Section, we present our design and implementation of the virtualization layer for Linux environment. Our networking virtualization architecture is shown in Figure 4.

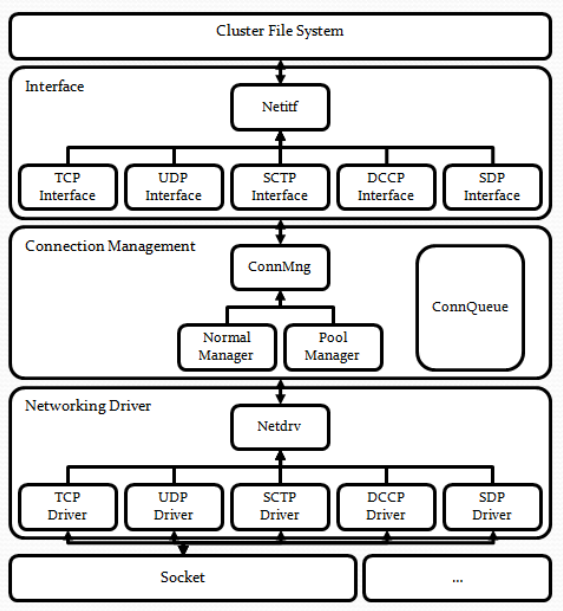

Figure 4. Our networking virtualization architecture

Our networking virtualization layer has three layers. Top layer is Netif to be used by the cluster file systems on the user-mode. Because the Netif layer provides single interface, clients and servers communicate with each other regardless of the networking methods. Therefore, the Netif alleviates the efforts to develop clients and servers and to add new networking methods. We use the socket interface and the VAPI so that it could remove the kernel dependency. This feature also alleviates the efforts to debug and develop upper-layer programs effectively. We show Netif interface in the Table 1.

The second layer is ConnMng, which is a connection management interface, to be used by the Netif. There are two connection types that are connection-oriented and connectionless in ConnMng. We support these connection types without modifying the source code of the cluster file system. Also we include connection pools, which are used to relieve start-up time when connections are established and closed.

In networking driver layer, the Netdrv supports TCP, UDP, SCTP, DCCP, Verbs, and SDP on user-mode to multiplex packet data to the kernel drivers.

Table 1. Netif interface

Return type	Name	Arguments
Int	create	t_conn *conn
Int	delete	t_conn *conn
Int	set_laddr	t_conn *conn
		const char *addr
		unsigned short port
Int	set_raddr	t_conn *conn
		const char *addr
		unsigned short port
Int	listen	t_conn *lconn

Int	accept	t_conn *lconn
		t_conn *nconn
Int	connect	t_conn *conn
Int	disconnect	t_conn *conn
Int	send_msg	t_conn *conn
		const char *msg
		int len
Int	recv_msg	t_conn *conn
		char *msg
		int len

Netif is similar to the BSD socket API. Because we are familiar with BSD socket API, the cluster file systems can use Netif interface relatively easily than other interfaces.

IV. EVALUATIONS

We applied our networking virtualization layer to a simple network file system using FUSE [5]. FUSE helps to develop a file system that operates on user-mode. We developed TCP on Gigabit Ethernet, IPoIB, SDP, and Verbs on InfiniBand drivers. So, simple benchmarks were used to evaluate the performance of our networking virtualization layer.

- Unixbench : read/write/copy on a file system
- SQLite : insert/select query on a database

The Unixbench is used to evaluate normal file system operations such as read, write, and copy [6]. And the SQLite is famous file database engine. We also evaluate random access performance by using SQLite's insert/select queries [7].

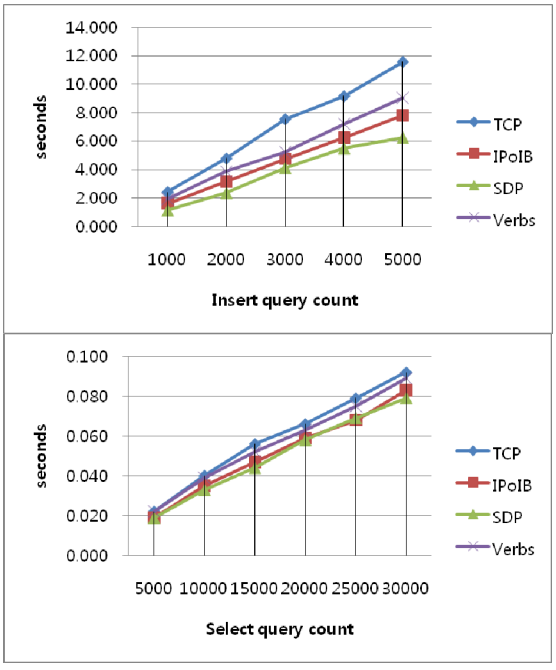

Figure 5. SQLite results

Table 2. Unixbench results

	TCP	IPoIB	SDP	Verbs
Write (Kbytes)	11200	43200	57600	28600
Read (Kbytes)	2568509	2601939	2637838	2585708
Copy (Kbytes)	9539	40756	49233	25432

In Figure 5 and Table 2, we show that SDP, IPoIB, and Verbs on InfiniBand are faster than TCP on Gigabit Ethernet. Unusually, optimized Verbs is faster than SDP and IPoIB [11]. However, our Verbs is not faster than SDP and IPoIB, because our Verbs is not optimized. We can verify that our networking virtualization layer supports several networking methods through these experiments. These results are achieved without modifying and recompiling the source code of the cluster file systems. The cluster file systems use a single interface of our networking virtualization layer.

V. CONCLUSION

In this paper, we proposed the design and implementation of the networking virtualization layer for cluster file systems, to support several networking methods with single interface. Our virtualization layer provided some new features: single interface for various networking methods, no kernel dependency, effective connection management, and easier deployment. To evaluate our scheme, we developed a simple network file system by using the FUSE and supported the TCP/SDP/IPoIB/Verbs protocols on Gigabit Ethernet and InfiniBand. As a result, InfiniBand was faster than Gigabit Ethernet and our non-optimized Verbs was not faster than SDP and IPoIB on the InfiniBand. We are going to optimize the Verbs interface and include other networking methods.

ACKNOWLEDGMENT

This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2009-(C1090-0902-0046))

REFERENCES

[1] Lustre File System. http://www.sun.com/software/products/luster.

[2] Parallel Virtual File System 2. http://www.pvfs.org/pvfs2.

[3] Gluster File System. http://www.gluster.org.

[4] Eric Yeh, Herman Chao, Venu Mannem, Joe Gervais, and Bradley Booth. Introduction to tcp/ip offload engines (TOE). White Paper, May 2002.

[5] File system in userspace. http://fuse.sourceforge.net

[6] D.C.Niemi, unixbench 4.1.0.
http://www.tux.org/pub/tux/niemi/unixbench

[7] SQLite. http://www.sqlite.org/.

[8] D.A. Bader and R. Pennington, “Cluster Computing: Applications”, The International Journal of High Performance Computing, May. 2001, pp. 181-185

[9] H. Frazier and H. Johnson. Gigabit Ethernet: From 100 to 1000Mbps, 1999

[10] InfiniBandSM Architecture Specification Volume 1, InfiniBandSM Trade Association, 2007

[11] Software Architecture Specification (SAS) - Linux System Software for the InfiniBandSM Architecture, Intel co., 2002.