Design and Implementation of Multiple Addresses Parallel Transmission Architecture for Storage Area Network

Bin Meng, Patrick B. T. Khoo, T. C. Chong
Data Storage Institute Affiliated to the National University of Singapore
{meng_bin, khoo_beng_teck, chong_tow_chong}@dsi.a-star.edu.sg

Abstract

In this paper, we present a parallel transmission architecture for SAN. By using two schedulers on the destination and source addresses of packets, the load of multiple data flows between multiple devices can be balanced in an asymmetrical topology without using special hardware. The SAN performance could be scaled flexibly and additional fault tolerance feature is provided. The load balancing algorithms we provide can be easily implemented and the computation is efficient enough for high-speed transmission.

1. Introduction

The demand for high availability and high performance in Storage Area Networks (SAN) is a driving force behind much of the effort on network architecture design. SAN performance must be able to grow flexibly to meet an organization's information storage and processing needs grow.

The use of parallel channels to build a SAN is an attractive method to achieve this result due to its high availability, scalability and flexibility. This paper outlines a study on multiple addresses parallel transmission architecture for SAN, in particular, its development using the HyperSCSI network storage protocol [1]. The aim of this work is to design and implement a parallel architecture network that can increase the end-to-end network I/O performance between HyperSCSI storage modules by employing more physical communication channels in a packet switching network. In addition, a built-in fault tolerant strategy for surviving and recovering from network failures is provided.

Research on parallel resource modeling is emerging in recent years. Several papers have provided theoretical models [2, 3, 4, 5, 6] for multiple resources scheduling in parallel network. These theories have proven to be very helpful in our efforts to design and evaluate the throughput, delay, and load balancing algorithms.

Previous researches concentrate on Multichannel Local Area Network (MLAN) [7], a bus sharing architecture. However, most new storage network applications work on packet switching networks [1], which is more scalable and flexible.

Other implementations are not designed for end-to-end communication. For instance, parallel SCSI cables [8], Link Aggregation Control Protocol (LACP) [9] and Sun Trunking [10], are designed for node to node communication. They are limited by distance, unbalanced loads and topology dependency. Although GridFTP [11] provides end-to-end parallel TCP streams, it only increases the channel utilization not the physical bandwidth. In our paper, we present a different design in order to avoid these limitations.

2. Theoretical models

In this section we will discuss the theories behind our network topology and data transmission model.

2.1. Multiple addresses parallel transmission model

In a SAN, the topology is arbitrary, the storage devices are expected to have variable bandwidth and packets can have variable sizes. Our model can meet these requirements by providing asymmetrical parallel channels between sender and receiver.

The theoretical model of a connection between a sender and a receiver is composed of five sequential components (Figure 1.): 1. the first stage destination addresses load balancing scheduler, 2. data flow priority controller, 3. the global second stage source addresses load balancing scheduler, 4. hashing table for address restoration and 5. data flow reorder. With this architecture, the bandwidth of each network storage node can be easily increased or decreased independently. This model can work in a normal packet switching network without the use of special hardware.

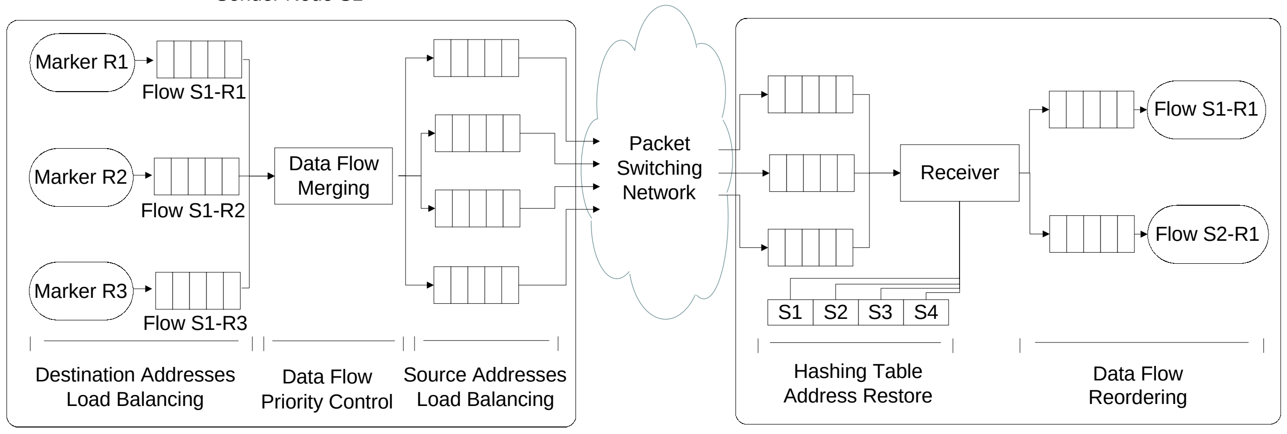

Figure 1. Model for multiple addresses parallel transmission architecture

2.2. The first and second stage load balancing schedulers on sender

We suppose that both sender node and receiver node have multiple network interfaces with unique addresses. These two address tables are exchanged before the data transmission.

In the sender node, two schedulers are used to balance the traffic load. Each data flow has one first stage scheduler respectively (Figure 1. Destination load balancing Markers), and all data flows share one global second stage scheduler (Figure 1. Source load balancing scheduler).

The first stage scheduler is used to balance load at receiver node by applying the load balancing algorithms on destination addresses. The second stage scheduler is used to balance the out going load at the sender node by applying load balancing algorithms on source addresses.

After a packet been assigned destination address and source address by these two schedulers, it is sent to the network and routed by normal routers and switches. Thus, multiple data flows can be balanced in an asymmetrical environment. Load balancing algorithms for two schedulers are derived from a reverse of the fair queuing algorithm [2]. Here, we intend to distribute variable sized packets to multiple channels. We discuss and compare four load balancing algorithms.

2.2.1. The Reverse Generalized Processor Sharing (RGPS). This algorithm is a reversal algorithm of Generalized Processor Sharing on the condition that every connection is backlogged. The original definition of GPS is:

$$\frac{S(i,\tau,t)}{S(j,\tau,t)} \geq \frac{\phi(i)}{\phi(j)}$$

Where $\phi(1)$, $\phi(2)$, …, $\phi(N)$, are positive real weights for the connection, and the server serves $S(i,\tau,t)$ amount of data from the i th connection in the interval $[\tau,t]$.

The RGPS is used to schedule one queue to multiple connections with service weights. It sends infinitesimally small amount to each connection. In RGPS, every connection is backlogged, because we intend to use all connection resources. From the RGPS definition, we have

$$\frac{S(i,\tau,t)}{S(j,\tau,t)} = \frac{\phi(i)}{\phi(j)}$$

This formula means all connections strictly share the service in proportion to their weights. RGPS is a perfect load balancing algorithm, but it is unimplementable since we must schedule packet, not infinitesimal. However, the concept of RGPS can be used to evaluate other algorithms.

2.2.2. The Reverse Weighted Round Robin (RWRR). The Weighted Round Robin (WRR) algorithm is used to serve a packet instead of infinitesimal from each backlogged queue in turn. The RWRR algorithm is reverse of the Weighted Round Robin. It sends packets to connections in turn with normalized weights. The traffic load is perfectly balanced if the packet size is fixed.

In SAN, the proportion of the similar sized packets is high. Thus we can use RWRR algorithm to approximate a perfect load balancing algorithm. The RWRR algorithm needs less computation overhead and the traffic load is roughly balanced when the packet size is similar. The RWRR work complexity is $o(1)$.

2.2.3. The Reverse Weighted Fair Queuing (RWFQ). This algorithm is a reverse of Weighted Fair Queuing algorithm [12]. The RWFQ algorithm serves channels in order of their finish number (or finish time), where the finish number is computed by RGPS algorithm. The channel with the minimum finish number is selected to send next packet. The weighted finish number equation is

$$F(i,k,t) = F(i,k-1,t) + T(i,k,t) \div \phi(i)$$

$$T(i,k,t) = L(i,k,t) \div r$$

where $T(i,k,t)$ is the service time of k th packet on connection i , $L(i,k,t)$ is the length of k th packet that transmits on connection i at time t , r is the link service rate, $\phi(i)$ is the weight of the connection i . The RWFQ algorithm is better than RWRR when packet size is not fixed. The work complexity of RWFQ is $o(\log(n))$.

2.2.4. The Reverse Deficit Round Robin (RDRR). This algorithm is a reverse of Deficit Round Robin algorithm [13]. It uses a Deficit Counter (DC) and quantum of service by measuring the weights of the channel. Each time a channel been selected, the DC is incremented by the quantum for that channel. Packets are sent to the channel and its DC is decremented by the packet size, till the DC becomes non-positive, then the next channel is selected. The RDRR work complexity is $o(1)$.

In our experiment, all three algorithms RWRR, RWFQ and RDRR are implemented. Each algorithm has its advantages. The RWRR algorithm is simple and efficient for scheduling large number of channels with fixed packet size. The RWFQ algorithm conducts a perfect load balancing; it can be used when storage system has excellent computation power. The RDRR algorithm is a better solution when many channels with various packet sizes are used and computation resource is constrained. These three algorithms are easy to implement for load balancing.

2.3. Data flow priority algorithm on sender

Between first and second scheduler, we merge the data flows into one virtual data channel (Figure 1. data flow priority), so that the packets of the conversation can migrate among the network interface without being aware of the underlying physical channels. Furthermore, keeping multiple conversations on one virtual channel is more efficient for packet transmission.

Here a service discipline [14] can be used to provide priority for the data flows merging. In this paper, all data flows are assigned the same priority to demonstrate the flexibility and the scalability of the multiple addresses parallel transmission.

2.4. Data flow recovery -- hashing addresses restoring and reordering on receiver

In the receiver node, data flows are restored and reordered when packets arrive at different network interfaces. Here, a hashing address searching algorithm (Figure 1. hashing table address restore) is used to restore the data flow and provide a bounded searching delay. The search complexity of a chaining with separate lists is $o(n/l)$, where n is the address number, and l is the list numbers.

After the data flows been restored, the out-of-order packets must be reordered (Figure 1. data flow reordering) by receiver. Unlike in a normal network [15], the out-of-order rate in parallel architecture can be large and predictable. In parallel transmission, out-of-order delivery can be caused by different channel bandwidths and packet sizes. The minimum reordering buffer size W is computed by

$$W = \sum_{i=1}^{N} Max(RTT) * \varpi_i / 2$$

where $Max(RTT)$ is the round trip time that a maximum sized packet used to pass through the minimum bandwidth channel, N is the numbers of the channels, ϖ_i is bandwidth of each channel.

Although the out-of–order rate could be large, the out-of-order distance is small and packets can be reordered efficiently without affecting the transmission performance.

2.5. Fault tolerance and detection algorithms

Since the multiple addresses parallel transmission architecture uses redundant channels, the scheduler can provide fault tolerance by detecting channel fault and rescheduling data flows to usable channels. The fault detection algorithm can use either hardware based, software local loop back and/or remote acknowledge detection methods. With different computation complexity, the failure of the first hop link, the nearest switch and the full data path could be detected.

3. Implementation and results

To demonstrate the multiple addresses parallel transmission protocol (MAPTP) in a SAN, we use the HyperSCSI [1] network storage protocol as the upper layer and the Ethernet link layer as the sub channel. The HyperSCSI protocol provides a virtual SCSI interface service by packing SCSI commands and data into network packets.

The MAPTP protocol is implemented as a Linux kernel module. The test bed of the MAPTP is a packet switching network composed of Fast Ethernet (FE) and Gigabit Ethernet (GE) links. The storage device server and clients are connected to the switch with differing numbers of links. We measure the performance by testing the Disk read speed of the HyperSCSI network storage device.

3.1. Disk read performance in asymmetrical and symmetrical parallel network.

The tests used HDPARM, DD and IOZONE to read 5GB data. With more channels added, the asymmetrical and symmetrical disk read performance of the HyperSCSI device increases nearly linearly (Figure 2, 3) until the limit of the system performance is reached. The asymmetrical performance is slightly better, because it use fewer network interfaces, which means fewer schedule overhead.

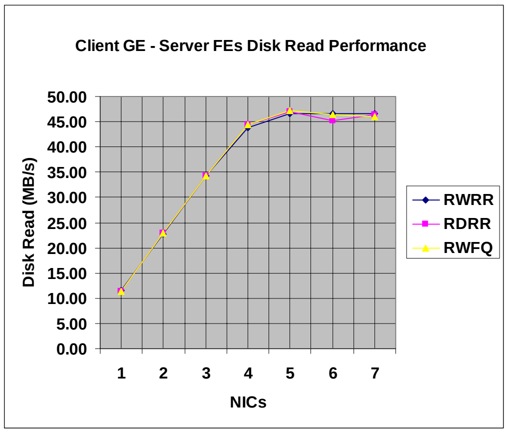

Figure 2. One GE - Multiple FEs asymmetrical disk read performance

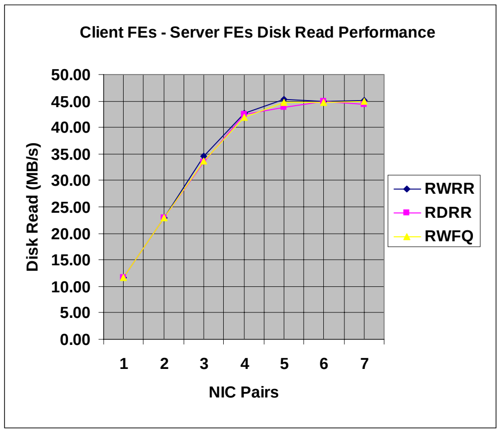

Figure 3. Multiple FE pairs symmetrical disk read performance

With all three scheduling algorithm RWRR, RDRR and RWFQ, HyperSCSI bandwidth can be increased above 380% when 4 links are used. Thus the network utilization rate of the multiple channels HyperSCSI is 95%. The overhead caused by MAPTP is roughly 5% per channel, which is acceptable when compared to the bandwidth increase. Because in SAN, most packets have similar size, all three algorithms have similar performance result. All these algorithms can be used for load balancing with little computation overhead.

3.2. Out-of-order rate analysis

From figure 4 and 5, we found that the out-of-order rate does not affect the disk read performance before the system limit is reached. And the out-of-order rate decreases when more channels are used. When using the FE pairs, the out-of-order rate decreases dramatically.

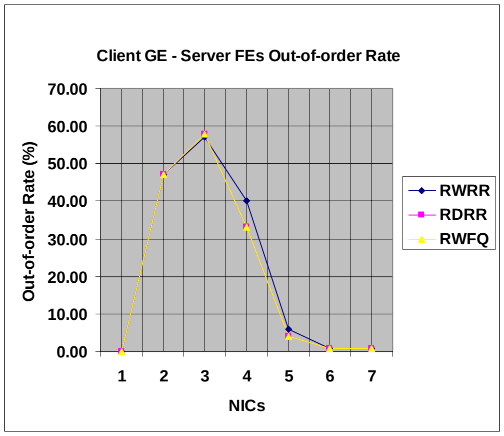

Figure 4. One GE - Multiple FEs asymmetrical out-of-order rate

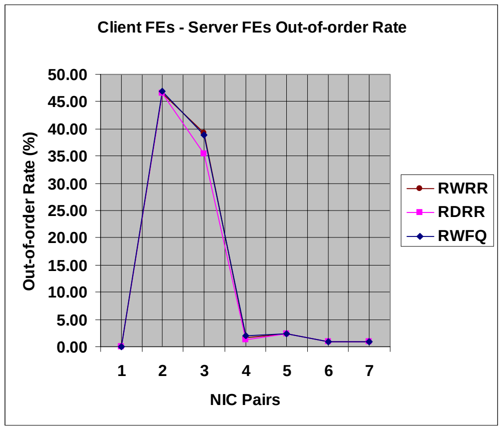

Figure 5. Multiple FE pairs symmetrical out-of-order rate

The reason of this rate drop is that in each schedule round, the schedule delay between sequential packets increase with more channels added. Larger schedule delay between packets means the probability of out-of-order is small. With more channel added, the out-of-order rate can drop rapidly. Similar result can also be observed when the packet size been increased and cause larger schedule delay. With packet size increased, the out-of-order rate also dropped.

3.3. Multiple clients - multiple channels disk read performance

For the multiple clients and multiple channels load balancing tests, one server with GE and two clients with multiple FEs are used. The figure 6 shows that the load is evenly balanced on both clients and channels. The sum of the disk read performance can approach the system maximum throughput when 6 FEs are used.

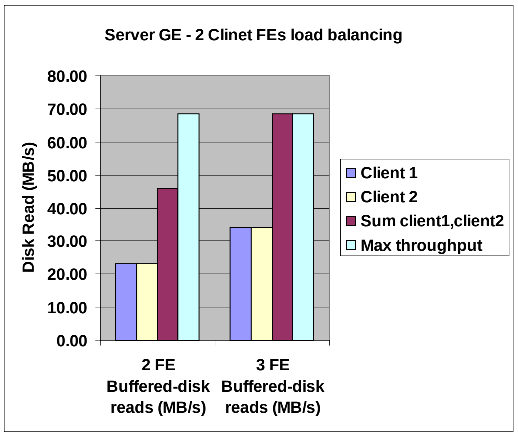

Figure 6. Load balancing on two clients with multiple FEs

4. Future work

Future research work may focus on QoS applications in parallel transmission packet switching SAN. More study should be conducted on out-of-order feature in a parallel transmission.

5. Conclusion

In this paper, we provide a theoretical model and an implementation demonstrating the multiple addresses parallel transmission architecture in action. This architecture supports flexible performance scaling in a packet switching network. As such, SAN topology using this structure will have good scalability and fault tolerance but without much additional complexity. In addition, by supporting Ethernet MAC and IP packets, the network storage devices can work in both LAN and WAN environments. This multiple addresses parallel transmission architecture could be a key factor in deploying SANs which can provide a highly scalable bandwidth with full redundancy over switched, parallel data paths.

References

[1] Patrick Beng T. KHOO and Wilson Yong H. WANG, "Introducing A Flexible Data Transport Protocol for Network Storage Applications," 10th NASA Mass Storage Systems and Technologies Conference / 19th IEEE Symposium on Mass Storage Systems, 15-18 April 2002.

[2] Hari Adiseshu, Guru Parulkar and George Varghese, "A Reliable and Scalable Striping Protrocol," SIGCOMM, 1996.

[3] Josep M.Blanquer and Banu Ozden, "Fair Queuing for Aggregated Multiple Links," ACM, SIGCOMM'01, August 27-31,2001.

[4] Jorge A.Cobb, "An In-Depth Look at Flow Aggregation for Efficient Quality of Service," IEEE/ACM transaction on networking, 2002.

[5] Jorge A. Cobb and Miaohua Lin, "End-to-End Delay Guarantees for Multiple-Channel Schedulers," IEEE, 2002.

[6] C. Brendan S. Traw and Jonathan M. Smith, "Striping Within the Network Subsystem," IEEE, 1995.

[7] M.A. Marsan and D. Roffinella, "Multichannel local area network protocol," IEEE, J. Sel. Areas in Commun., vol. SAC-1, pp. 885-897, Nov. 1983.

[8] Bill Ham, Digital Equipment, "Parallel SCSI Grows, Shrinks and Stays the Same," www.scsita.org, 1997.

[9] "Link Aggregation Control Protocol," IEEE Std 802.3, 2000 Edition.

[10] SUN Trunking Software.http://www.sun.com .

[11] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. " Data Management and Transfer in High Performance Computational Grid Environments," Parallel Computing Journal, Vol. 28 (5), May 2002.

[12] Abhay K. Parekh and Robert G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," IEEE/ACM, Transactions on Networking, Vol.1, NO. 3, JUNE 1993.

[13] M. Shreedhar and George Varghese, "Efficient Fair Queuing using Deficit Round Robin," SIGCOMM, ACM, 1995.

[14] Hui Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," Proceeding of the IEEE, 83(10), Oct 1995.

[15] Vern Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM transactions on networking, VOL. 7, No. 3, JUNE 1999.