NETWORK STORAGE SYSTEM SIMULATION AND PERFORMANCE OPTIMIZATION

WANG CHAOYANG

NATIONAL UNIVERSITY OF SINGAPORE

2005

NETWORK STORAGE SYSTEM SIMULATION AND PERFORMANCE OPTIMIZATION

WANG CHAOYANG

(B. Eng.(Hons), Tianjin University)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRONIC AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

For Rong Zheng

Only your patience and your love and constant support have made this thesis possible.

Acknowledgments

I am sincerely grateful to my supervisors Dr. Zhu Yaolong and Prof. Chong Tow Chong for giving me the privilege and honor to work with them over the past two years. Without their constant support, insightful advice, excellent judgment, and, more importantly, their demand for top-quality research, this thesis would not be possible.

I would also like to thank my ex-colleagues from Data Storage Institute (DSI), especially the former Storage System Implementation & Application (SSIA) group and the current Network Storage Technology (NST) division. Without the collaboration and the associated knowledge exchange with them, this work would again simply impossible. I would like to delivery my special thanks to Mr. Zhou Feng, Miss Xi Weiya, Mr. Xiong Hui, Mr. Yan Jie, Mr. So Lihweon and Mr. David Sim for their long lasting support.

Last, but not least, the support of my parents, parents-in-law and my wife should be mentioned. I would like at this point to thank my dear wife Rong Zheng, who has taken many household and family duties off my hands and thus given me the time that I needed to complete this work. I would also like to thank my parents-in-law, who has taken care of my daughter during this period.

Contents

Summary

Storage systems are generally built by Redundant Array of Independent Disks (RAID) technology to meet the high performance requirement of enterprise applications. Besides RAID technology, the interconnection between the Hard Disk Drives (HDDs) and the RAID controller plays an important role in a high performance storage system.

Recently, the Fibre Channel Arbitrated Loop (FC-AL) has become the most common interconnection in the high-end storage systems. The FC-AL topology provides a high performance serial shared connection between the RAID controller and the attached HDDs. In such shared connection, all participating devices have to compete for the access to the loop. When the loop is occupied by data transmission, the controller has to wait until the loop is free in order to deliver I/O commands to the HDDs. In such situations, the target HDDs may stay inactive, resulting in inefficiencies of HDD utilization and finally affecting the whole RAID system performance.

In order to evaluate the performance of a network storage system, this thesis develops an FC-AL based network storage system simulation model that can simulate the FC-AL protocol up to frame level. The simulation model is developed through a "bottom-up" approach. The FC-AL transmission is modeled in the first place, followed by the development of L_Port's other functionalities including the Loop Port State Machine [LPSM] and the Alternative Buffer-to-Buffer flow control. After that, the HBA model is provided and the system level integration is performed with additional consideration of HBA device driver modeling. Lastly, the FC-AL based network storage system simulation model is calibrated and validated through actual system experiments. The comparison between actual experiments and simulation shows that the simulation model can achieves high accuracy as to 3% mismatching for read I/Os.

A new scheduling algorithm for the FC-AL RAID system, the Command-First Algorithm, is proposed to enable RAID controller to aggressively send I/O commands to the HDDs with higher priority than I/O data. The Command-First Algorithm is evaluated using the simulation model. The simulation results show that the performance improvement contributed by the new algorithm is up to 50% in certain conditions. It is also shown that there are no negative effects for the Command-First Algorithm.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to Data Storage & Storage System

Along with the rapid development of IT technology, the demand for higher performance and bigger capacity on data storage has been constantly increasing in the past decades. Multimedia technology enables people to store videos in the form of hundreds of mega bytes of digital data and to playback anytime. Large databases are widely implemented for decision-making or process controlling, which requires data to be up-to-dated and available constantly. A large number of mission-critical applications demand for high performance for data storage.

The magnetic hard disk drives (HDDs) are used as the primary storage device for a wide range of applications. Since it was invented half-century ago by IBM, the HDDs have undergone continuous technological evolutions, yielding larger-capacity, higher-performance, smaller-form-factor and lower-cost. The areal density of HDD has increased about 35 million times since it was first introduced [6]. The recent CGR (compound growth rate) of the areal density is about 100 percent, or doubling every year, which has broken through the Moore's law of doubling capacity every eighteen months for the semiconductor growing. In year 2005, the HDDs with capacity of hundreds gigabytes are commonly available.

Even with the areal density positively advancement, the total HDD shipment surprisingly does not decrease. The two famous market research companies, TrendFOCUS and IDC, both forecasted over 20 percent grow of total units of HDD shipment from about 305 million units in year 2004 to about 378 million units in year 2005. The essential reason for demanding more HDDs is that the HDD access performance increases much slowly comparing to the capacity improvement. The CGR slopes of the mechanical seeking time and the rotational latency of HDD has only about 25 percent [5]. The individual HDD is therefore not able to meet the enterprise's performance demand.

To fill the performance gap and to optimize the cost and reliability, the storage system that can provide aggregated performance of multiple HDDs has long been one of the corner stones for enterprise data storage. The RAID technology enables the storage system to serve I/O request in parallel through striping user data across multiple HDDs, and to enhance system reliability by parity protection preventing data lost in the event of individual HDD failure. By introducing large memory cache, the storage system can accelerate the I/O requests without reading data from the HDDs. Many other technologies have been developed to optimize the performance. One important technology is the interconnection between the HDDs and the RAID controller, which may limit the storage system performance.

A storage system usually consists of one or more separate control units and multiple HDDs. The control units access to the HDDs through an interconnection. In ideal situations, each HDD shall dedicatedly connect to the storage controller by means of unblocked switching network for high parallelism, but it would require much higher cost. The balance between the parallel performance and the cost is the crucial factor for success. A shared connection is therefore used as an alternative to provide the sufficient bandwidth. After the traditional SCSI bus architecture, the Fibre Channel Arbitrated Loop (FC-AL) has become the most frequently used interconnection for high-end network storage systems.

1.2 Main Contributions

This thesis provides four major contributions to the studies of FC-AL based high-end storage systems as following:

- An effective and detailed simulation model is built to support frame and transmission word level simulation;
- Hardware trace level calibration and actual system experiment comparison are performed for simulation model validation;
- A new schedule algorithm is proposed to aggressively delivery I/O commands to optimize I/O performance;
- The simulation results show that the performance improvement contributed by the new algorithm is up to 50%.

1.3 Organization

The thesis is organized as follows. Chapter 2 presents the basic background of storage systems and investigates the current status of research in FC-AL network storage systems. Chapter 3 conducts operational analysis on FC-AL based storage systems and presents the Command-First Algorithm. In order to effectively evaluate the performance of a network storage system, a detail simulation model for FC-AL storage system is presented in Chapter 4. The simulation model is calibrated and validated in Chapter 5. Chapter 6 presents the I/O performance evaluation of the Command-First Algorithm by simulation. Finally, Chapter 7 summarizes the research and discusses the future research work.

Chapter 2

Background and Related Work

2.1 Fibre Channel Overview

Fibre Channel (FC) is a high speed serial interface defined by the ANSI (American National Standard Institute) as an open industry standard. There are more than 20 published standards or drafts for different aspects of FC [13]. More recent development of the FC standards can be found in the FC Project of the T11 Technical Committee [12].

FC is generally characterized by high speed, long distance, and high scalability for storage. It provides a general transport network platform for Upper Level Protocols (ULP) such as SCSI (Small Computer Systems Interface [38]). The SCSI mapping over the FC is defined in FCP (Fibre Channel Protocol for SCSI) [11].

FC can be logically divided into five logical layers, numbered from bottom to top as FC-0 to FC-4, as shown in Figure 2.1. Similar to layers in the OSI's model, each FC logical layer performs a certain set of functionalities interfacing to nearby layers. The FC-0 layer defines the physical interface for the FC network for the specification of transmitter, receiver and the signal propagation media, which includes the fiber optic cable and the electronic copper cable. The FC-1 layer performs 8bit/10bit coding and decoding and error control. Sitting on top of the FC-1 layer, the FC-2 organizes information into a set of frames, sequences, and exchanges and defines other signaling protocols such as flow control. The FC-3 layer provides additional common services such as multiple link trunking, multicasting and other services. The FC-4 layer

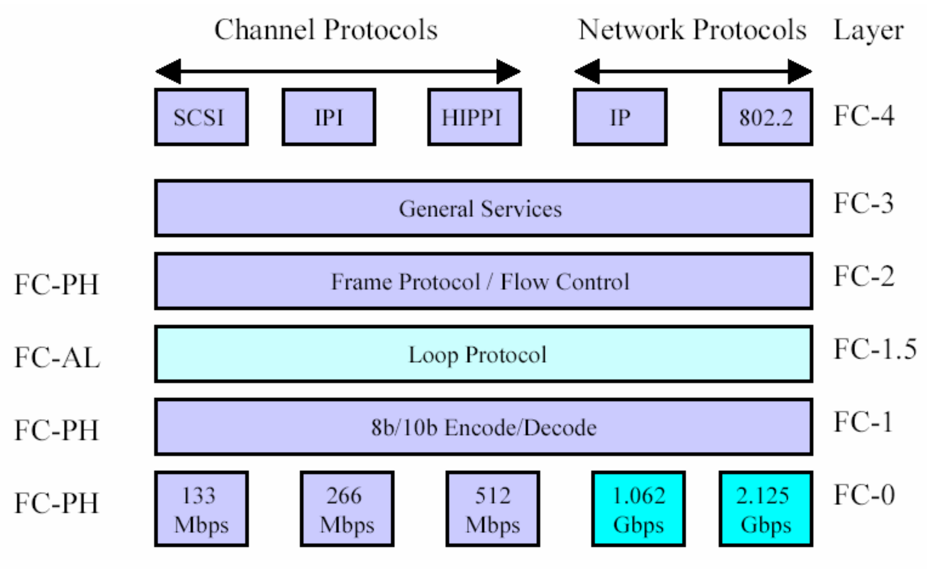


Figure 2.1 Fibre Channel Logical Layer

facilitates the mapping to upper-level protocols such as SCSI, IP, and others. Additionally, there is a Fibre Channel Arbitrated Loop (FC-AL) [9] protocol between the FC-1 and FC-2 layers labeled as FC-1.5 in Figure 2.1, which allows the attachment of multiple devices to a common loop without switches. The FC-0, FC-1 and FC-2 layer are collectively defined in FC-PH [10].

Three basic classes of service are defined in FC standard: Dedicated connection (Class 1), Multiplex (Class 2) and Datagram (Class 3). Class 1 provides circuit switch, dedicated bandwidth connection. The connection must be established before data can be transferred. Once the connection is established, the full bandwidth is guaranteed until one party releases the connection. Class 2 is a connectionless service. Frames are independently routed to the destination port by the Fabric, if present. An end-to-end acknowledgement of frame reception is required for this class. Class 3 is similar to Class 2, except that no acknowledgement of receipt is given. In Class 3, the fabric, if present, does not guarantee the successful delivery of frame and it may discard frames without notification under high-traffic or error conditions; any error recovery or notification is done at the ULP level. Without acknowledgement, the Class 3 service provides the quickest transmission and thus it is the most frequently used in various applications including the SCSI application for storage systems.

2.2 Fibre Channel for Storage

2.2.1 Fibre Channel SANs

A Storage Area Network (SAN) is a dedicated, centrally managed, secured information infrastructure, providing any-to-any interconnection of servers and storage systems. SANs are currently the preferred solution for fulfilling a wide range of critical data storage demands for enterprises [30].

The FC is presently the dominant protocol used in SAN to provide the high performance data connection. The perfect marriage of the two technologies makes the great success of both FC and SAN, although other emerging alternatives such as iSCSI protocol are now developed as the compliments to FC for low cost and other considerations. Many SAN books actually exclusively discussed the Fibre Channel technologies adoption, such as [27], [28] and [29].

Fibre Channel supports three types of connection topologies, Fabric, Point to Point and Arbitration Loop. Since the FC-AL provides a cost effective shared connection among multiple devices without using expensive switches, it has become a popular means of interconnecting the storage controllers to the attached HDDs.

2.2.2 FC-AL for Storage System

Since IBM introduced the world’s first storage device in 1945, the storage system has gone through the same period of evolution as the HDD did [5]. Initially, a storage subsystem was just a HDD. Over time, more hardware and software functions were added to the storage system to achieve higher performance, better reliability and lower cost [6]. The RAID technologies were first proposed in 1980s in [7] to provide a means of parallelism between multiple HDDs to improve the aggregate I/O performance and at the same time to extend the whole system reliability through redundant parity. Since then, various new technologies had been developed to enhance and optimize the I/O performance of the RAID storage system [8], and the storage system has become a cornerstone of the entire data storage industry.

Among other factors in a storage system, the interconnection between the storage controllers and the HDDs is important for the high I/O performance and reliability. Alternative to the traditional parallel SCSI bus architecture, the FC-AL provides a high performance reliable common sharing serial interconnection for multiple devices. Although it is shared topology, the loop has the channel property with which one device can establish a dedicated communication channel with anther device on the loop.

The FC-AL topology supports up to 127 devices within a single loop. With 1 G link rate (precisely 1.0625 GHz clock), the loop provides a common 100 MB/s bandwidth information transport vehicle for all devices. With support of full duplex, one may transmit or receive data frames simultaneously and thus achieves double the bandwidth. The latest development of 4 G link rate further increases the bandwidth to 400 MB/s and 800 MB/s for half duplex and full duplex respectively. With optical cables, the physical distance of a loop may extend to 10 kilometers. Additionally, inherited from the common FC feature, the loop provides higher reliability of communication. All the above mentioned advantages make the FC-AL connection far exceed the traditional parallel ATA and SCSI interface. Figure 2.2 shows such a storage system deploying the FC-AL topology with one initiator (controller node) and multiple HDDs.

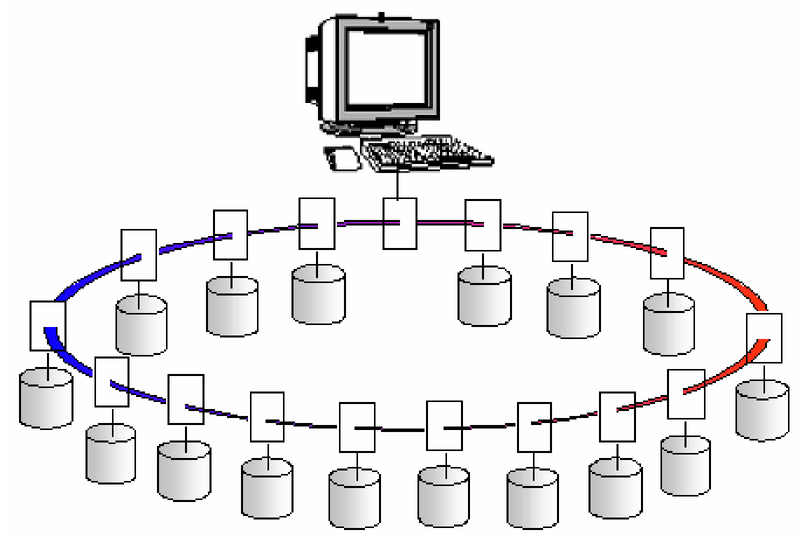


Figure 2.2 Fibre Channel Arbitrated Loop Topology

Nowadays, a large number of Fibre Channel HDDs are shipped every month from every major HDD vendor. These HDDs are mostly (if not all) used as member HDDs in a storage system. They are most frequently connected through FC-AL loops. It is not surprising, then, to see a large number of academic publications on FC-AL related storage system architecture. In the work of Shenze Che and Manu Thapar [22], the performance of the Video-on-Demand server using FC-AL was compared to traditional SCSI interface. The reported performance improvement was 50% better. In [23], the authors provided a software architecture enabling FC-AL based RAID system in a real-time operating system. The potential of low-cost switching architecture for extending FC-AL scalability was studied in [24] and a concreted implementation and study of FC-AL architecture in a real application were presented in [25].

2.3 Storage System Performance Study Methods

Many research works have been conducted on storage technology, storage networking, and storage subsystem. All those works eventually aim to achieve better performance in terms of higher throughput, shorter latency and wider bandwidth. The performance analysis becomes the key to predict, assess, evaluate and explain the system's characteristics. There are generally three approaches to conduct performance analysis for computer system: analytical modeling, physical measurement and simulation modeling [41].   A survey on the success stories of using these approaches to study the storage system performance was provided in [14].

The alternative to the analytical modeling and physical measurement is the simulation modeling, in which a computer program implements a simplified representation of the behavior of the components of the storage system, and then a synthetic or actual workload is applied to the simulation program, so that the performance of the simulated components and system can be measured. Simulation can provide a view of the system behavior at any level of detail, provided that enough modeling manpower is available. Trace-driven simulation is an approach that controls a simulation model by feeding in a trace, a sequence of specific events at specific time intervals. The trace is typically obtained by collecting measurements from an actual running system.

2.3.1 Performance Study by Simulation

The physical measurement performs testing and collects measurements performance data of a running system. By analyzing the relationship between the performance characteristics, the workload characteristic, and the storage system components, researchers are able to identify problems and give make decisions on purchasing and/or configuration for storage system. In [26], Thomas M. Ruwart had conducted experimental testing on a real system for different combinations of loop distance and hard disk number.

The real system experimental tests however are often subjected to the given implementations of vendor specific loop devices, such as the number of the frame buffer and FC-AL scheduling. Experimental modifications on such hardware are often not feasible for academic research. Meanwhile, real system experiments usually involve a very high cost. To conduct a study like [26] will require expensive infrastructure such as kilometers of fibre optic cables and other equipments.

On the contract, the simulation does not require the presence of an actual system. In [20], John R. Heath and Peter J. Yakutis implemented their simulation models and analyzed the performance of FC-AL based storage systems. They discussed the FC-AL protocol in detail but they did not provide the calibration and validation detail of the simulation model. Similarly, in [21], David H.C.DU and Tai-Sheng Chang et al. compared SSA (Serial Storage Architecture) [39] and FC-AL interfaces for disks by simulation, but the detail modeling method of the FC-AL was not given. Xavier [15] and Petra [16] also developed simulation model for FC but they modeled more on Fabric SAN. Some published simulation tools for other storage system's components can also be found. The DiskSim[17] and Pantheon[19] are the two well known HDD simulators. The former had been used in many HDD performance researches such as the time-critical I/O in [18] and [35], and the HDD schedule optimization in [31] and [32]. A detail simulation model of a system bus (PCI bus) can be found in [36].

Although simulation modeling has been proven to be an effective approach for system performance study and new algorithm evaluation, there are some limitations on current available simulation tools. Firstly, there are few simulation tools that can support detailed enough simulation studies especially when systems under study become more complicated. Secondly, a simulation model is an abstracted presentation of an actual system. Some system reactions are assumed to have minimum impact to the overall performance and others are modeled as constant overheads (or random variables with stochastic distribution). The simulation model must therefore be calibrated with actual system measurements for these overhead constants and further be validated by examining the simulation results to agree with experimental measurement, before it can be used for performance prediction in extended situations. Although some of the above mentioned FC-AL studies were done through simulation, the calibration and validation of these simulation models were seldom given. It is therefore worthwhile to develop a new simulation tool that can simulate the detail behavior of the FC-AL network storage system.

2.3.2 Theoretical Estimation by Analytical Modeling

The analytical modeling makes attempts to predict storage system performance as a function of parameters of the workload, storage components, and system configuration by writing mathematical equation. The work in [34] severed as an example of this approach. The analytic analysis can provide insight into the steady-state performance and give theoretical performance bounds of the storage system. It usually needs queuing theory and Markovian analysis, which requires extensive knowledge of probability theory. In addition, analytical modeling requires skill at approximating the storage system with simplified mathematical models.

In most analytical works, the internal components of a storage system are modeled as various service centers that can process requests at a certain service rate. The arrival requests, i.e. the service demands, are assumed to follow certain distribution (mostly in Poison Arrival that describes the independent arrival) and the service rate of the service centers are of some stochastic pattern (such as Poison Process) as well. Although the analytical modeling may lack detail when compared to the real system physical measurement and the simulation, it gives some theoretical insight of the process and effectively predicts the performance bounds of the given storage system.

In [1], Dr. Zhu et al presented their analytical work on SANs for the purpose of identifying performance bottlenecks. A queuing network model for storage system and storage network was established from the host systems, along with the FC fabric network, to the disk array internal components. Six tiers of services centers were defined to model the I/O processing activities, namely Hosts, FC-SW network, Disk Array Controller and Cache, FC-AL Network, Disk Controller and Cache and HDA Center, as shown in Figure 2.3 adopted from the paper. The Fork/Join model was used to analyze the performance of the disk array. The response time and utilization of each component as well as the overall system were derived and analyzed based on the queuing network theory.

With regards to the performance of FC-AL Network, the authors highlighted that the "access fairness" algorithm may be a potential problem for disk array controllers to obtain the optimal overall performance.

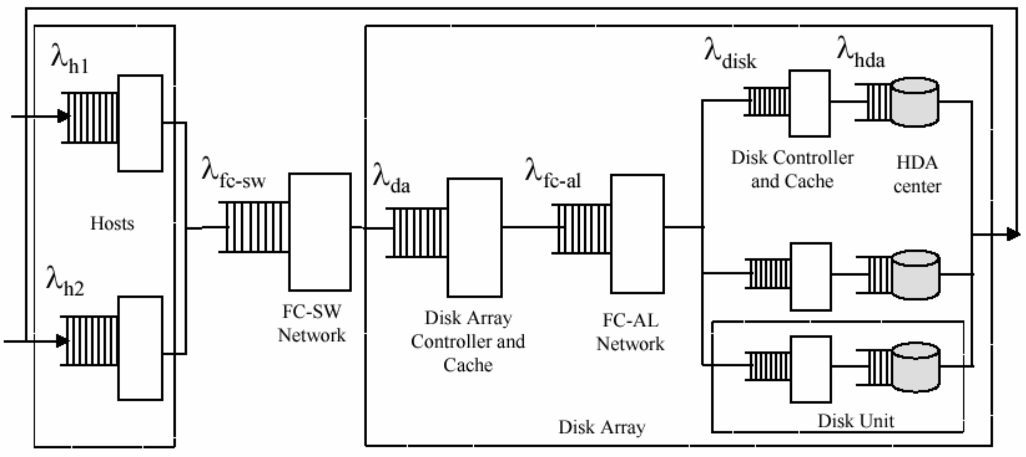


Figure 2.3 Queuing Network for Storage System Adopted From [1]

2.4 Summary

This chapter has presented a basic background of the FC standard and the FC-AL topology used in high-end network storage systems, with an overview of the FC logical layer, followed by a short discussion on the related works on the FC-AL based storage system. The performance study methods for storage system were investigated, and the simulation method has been identified to be an effective approach for detailed modeling.

Chapter 3

Command-First Algorithm

3.1 Analysis of FC-AL Network Storage System

In today's Information Technology infrastructure, there are two basic technological choices of connecting storage: NAS and SAN. The traditional Network Attached Storage (NAS) provides file level storage for Local Area Network (LAN) clients/servers. When LAN clients/servers need to access the information stored in the NAS, they send file requests to the NAS. The NAS then retrieves the information from the attached storage system and response to the request. The SAN technologies provide high performance connection between multiple SAN application servers to multiple

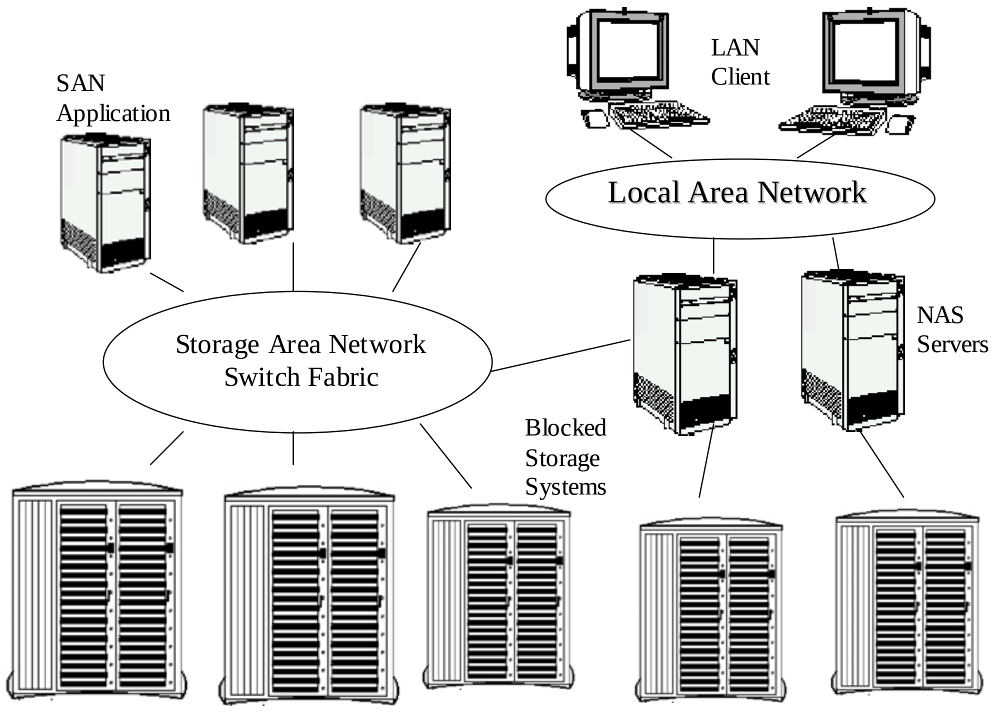


Figure 3.1 Storage System for SAN and NAS

storage systems, characterized by high bandwidth, dedicated connection and great flexibility of space scaling and resource relocation.

In both SAN and NAS scenarios, the storage system plays an important role in the whole picture of networked storage. The storage systems' performance always becomes the key factor to the overall I/O performance. Practically, the storage systems are one of the key components of IT infrastructure. Figure 3.1 illustrates the storage system's position in the overall picture of network storage.

3.1.1 FC-AL Based Storage System

A storage system is generally a collection of hard disk drives (HDDs) that are aggregated and managed by the storage controller in the form of either a compact hardware solution or a relatively more software oriented solution. The RAID technologies are often employed to improve the whole system's reliability.

Upon receiving an I/O command from the host system, the storage controller goes through its software and hardware elements to determine which member HDD to access. Accesses to member HDDs are done through an interconnection between the storage controller and the member HDDs. The interconnection can be either a fabric network or a FC-AL loop in the case of Fibre Channel connection. Although the fabric is the fundamental element of a Storage Area Network (SAN), it does not bring essential benefit for higher performance compared with the FC-AL connection within a storage system. For one example, if a storage system is supposed to have one interface connecting to the external fabric network, the bandwidth bottleneck is on that connection for the reason that all internal traffics from every attached HDDs must go through the single connection. Moreover, putting a fabric switch element in a storage system imposes much higher costs than FC-AL. Therefore, the FC-AL interconnections are widely adopted in today's high-end storage system.

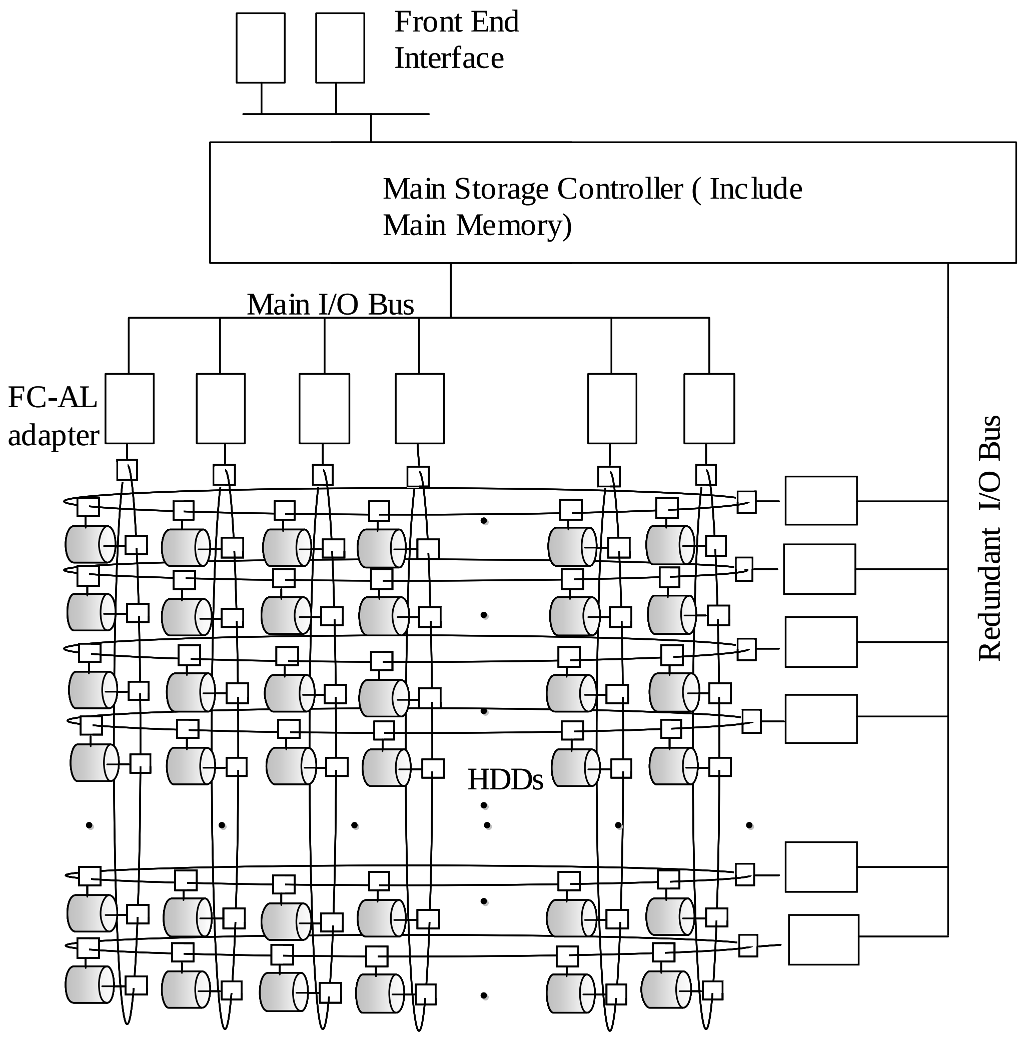


Figure 3.2 FC-AL Storage System Architecture

The FC-AL based storage system referred to in this thesis means the storage system where the interconnection between the storage controller and the attached HDDs is based on Fibre Channel Arbitrated Loop. With FC-AL, the storage system may physically easily connect hundreds of HDDs with several interface controllers (FC-AL adapter) each connected to a loop. Today's HDD shipped by most vendor supports dual loop connection. This feature is often explored to form a second independent redundant I/O path for high fault-tolerance. Figure 3.2 shows a typical FC-AL based storage system that have multiple FC-AL adapters where each of the Main I/O bus connects to a vertical loop and each of the Redundant I/O bus connects a horizontal loop. The member HDDs are located on those intersection grids of the two groups of different dimensional loops so that they can be accessed either by the main adapters or by the redundant adapters. Although most adoptions use the second I/O path as redundant to the main one, some other vendors activate both I/O paths with load balance over them to provide doubled overall bandwidth.

3.1.2 Storage Controller

The storage controller is the core of a storage system. It serves every external I/O request, and initiates and manages every internal I/O. It is a computer system equipped with various intelligent and value-added functional modules in either hardware or software forms. Figure 3.3 shows an example of storage controller internal architecture. The storage controller consists of three I/O buses and one system bus connecting by a chipset bridge. One target HBA (Host Bus Adapter) is sitting on the front bus to receive external I/O request. Multiple initiator HBAs are used and inserted

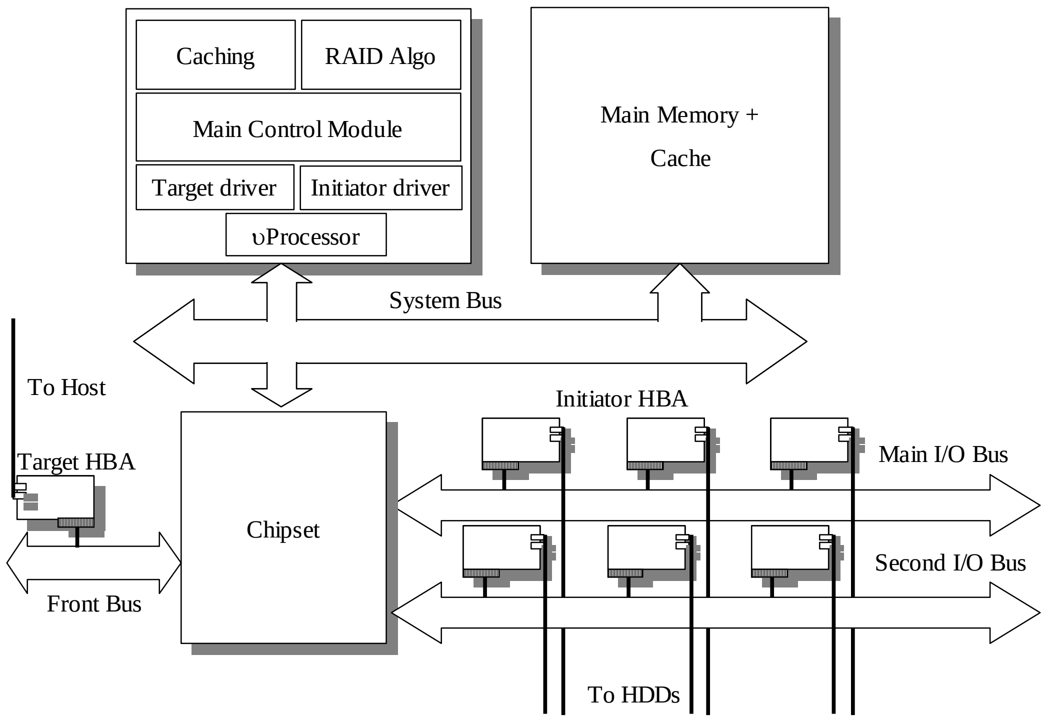


Figure 3.3 Storage Controller Internal Architecture

into Main I/O Bus or Second I/O Bus, and each of them connects to a FC-AL loop of HDDs. A microprocessor and a large memory module are connected through the system bus on the other end.

A set of software module stacks that handles I/Os is loaded to the microprocessor. The software stack typically includes the device drivers for both target HBA and initiator HBA. A main control software module governs the overall I/O activity. When an external I/O arrives, the target HBA notifies the main control module through the target driver. The main control module passes the I/O to the caching module to see if the data requested is available in the main memory. If the requested data is found in the main memory by the caching module, the I/O is served and data is transferred back to the external requestor by the target driver through the target HBA. If the caching

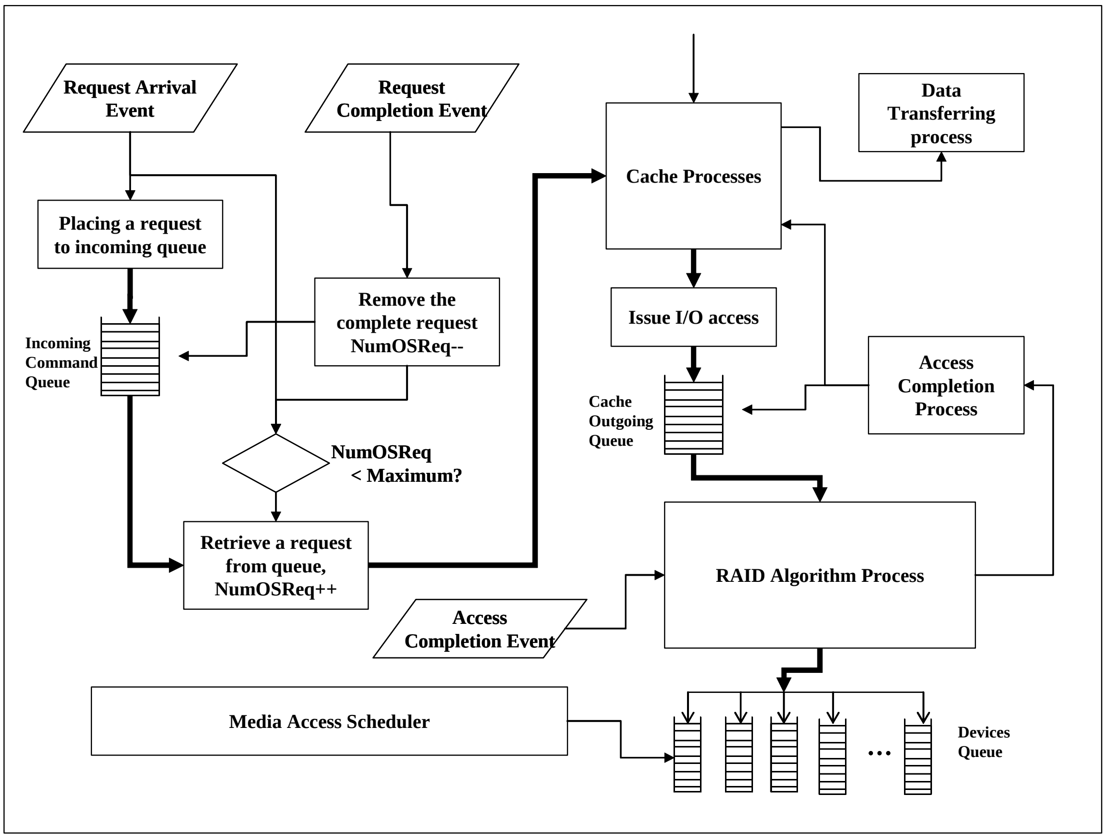


Figure 3.4 RAID Controller Internal I/O Process Flow

module reports a miss, i.e., the request data are not found in the main memory, the request is passed to a RAID algorithm module to determine where to read or write the requested data. Depending on different algorithms used, the RAID algorithm module processing may result in multiple internal I/O requests accessing multiple attached HDDs. These internal I/O requests are scheduled by the main control module and are submitted to the initiator diver so that the initiator HBA can deliver them to the destination HDDs. After these internal I/O requests are served by the HDDs, the requested data are sent back to the controller through the initiator HBA. Figure 3.4 shows an example of I/O processing flow in RAID controller in further detail.

3.1.3 Interfacing to the Host Bus Adapter

The Fibre Channel Host Bus Adapter (HBA) is an important component in a storage system for high performance I/O. It provides completed assistance for Fibre Channel operation with only minimal involvement of CPU of the host. The system involvements are done through the HBA device driver. When an I/O request is issued from the system, the HBA device driver is given an I/O request package with complete information of the I/O, such as operation type of read or write, the location of the destination (LUN+LBA), and the location in the main memory of the data buffer that holds the requested data. The device driver then puts the I/O request package in place and quickly issues a command through memory mapped control registers to the HBA. After that, the device driver rests and the host system is free from the I/O operation until the completion is reported, by means of interruption if necessary. The HBA needs to use the I/O bus from time to time for DMAing data to or from the System Memory. Figure 3.5 illustrates an example of a Fibre Channel HBA operation environment.

3.1.4 FC HBA Internal Operation

The I/O operation path from the storage controller, through the device driver, to the HBA that receives an I/O command has been discussed. The more detail internal operation of the HBA is analyzed in this section. Referring to the same diagram of Figure 3.5, an FC HBA typically contains a microprocessor that acts as the coordinator for I/O operation, a bus control and DMA arbiter that manages the utilization of the system I/O bus and performs DMA operation for accessing system memory, a link control unit that directly deals with the FC physical link, and a frame control that performs the frame management. A pair of FIFOs (First-In-First-Out frame buffer) is used to temporarily hold the incoming and outgoing frames. A set of HBA specific commands is defined for the microprocessor to execute functions, such as reset, status report, I/O command and others. These commands are designed in compact size with only few bytes so that it can be delivered quickly to the HBA through the device driver. The HBA retrieves the information of the I/O request through the Bus Control & DMA

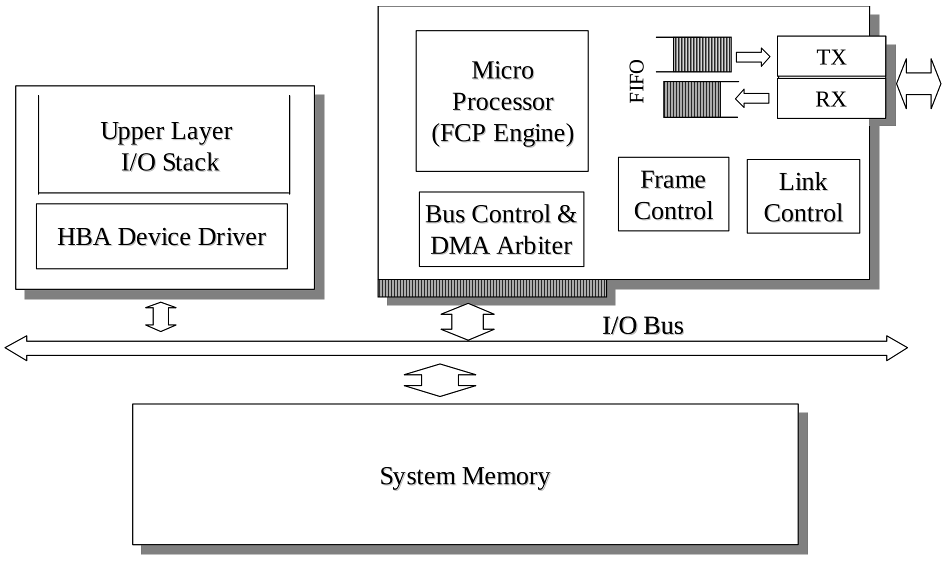


Figure 3.5 Fibre Channel HBA Operation Model

from the system memory and then allocates necessary resource for executing that I/O request. A complete set of indexing information is established, such as the frame header that contains a reference pointing to the I/O request. Per FCP standard, the FCP_CMND frame is then constructed and placed into the outgoing FIFO with assistance from the frame control. The link control establishes a connection with the target and transfers the command frame to the target from the outgoing FIFO.

The target retrieves the I/O information from the FCP_CMD frame and executes the I/O request. For read, the requested data obtained from the media are sent through a sequence of FCP_DATA frames followed by a FCP_RSP indicating the completion status. For write, the target allocates the memory buffer to receive the writing data and sends FCP_XFER_RDY to the initiator. When initiator receives the FCP_XFER_RDY, it looks up the indexing previously established and transfers the data from the data buffer referred by the indexing in FCP_DATA frame sequences. Upon successfully transmitting all data, the target sends FCP_RSP to report the completion.

For read, when the initiator HBA receives a FCP_DATA frame, the frame control unit reports to the microprocessor. The microprocessor retrieves the data payload from the FCP_DATA frame with assistance of the frame control, looks up the indexing to get the data buffer location in the system memory and triggers the bus control unit to DMA the data to system. The process of retrieving data from frame is referred to as de-encapsulation, which may be done with other hardware components to offload the microprocessor. For write, the imitator HBA receives FCP_XFER_RDY in the incoming FIFO. The frame control unit informs the microprocessor about the reception and the microprocessor interprets the information embedded in the frame to get the size of the data to be transferred corresponding to this FCP_XFER_RDY and looks up the indexing for the data buffer location in the system memory. The bus control and

DMA arbiter is then instructed to receive data from the data buffer, and the frame control encapsulates the received data into FCP_DATA frames and places them into the outgoing FIFO. The link control proceeds to transmit the frames from the outgoing FIFO to the destination.

For both read and write, when the FCP_RSP is received, the initiator HBA may or may not interpret the completion status directly, depending on the different implementation. The raw FCP_RSP or the interpreted completions information is sent to the designated memory location that the device driver knows and interrupts the system for attention. The device driver is activated by the interrupt and performs error-free checking based on the completion information. If the I/O request is successfully executed, the device driver reports to the requestor and the I/O is completed. Otherwise, the device driver may re-issue the I/O request to the HBA for retry, depending on different error types. The retry may be conducted several times up to a maximum limit. If it still fails, error recovery routine will be triggered and the I/O status is reported to the requestor.

3.2 Performance Limitation of Command Queuing Delay

3.2.1 External I/O Queue

As previously discussed, a storage system is designed to provide aggregated performance of a set of HDDs. Multiple I/O requests may be concurrently sent to different HDDs. The maximum number of I/O requests that the storage system can simultaneously process directly affects the aggregated performance.

When the storage system is used as a virtual disk dive, it may report this maximum outstanding request number (queue depth) to a client in the system initialization procedure. The client can issue no more than that number of outstanding requests at any given time. Exceeding that, additional I/Os are placed in a waiting queue (referred to as client-site queue) until at least one outstanding request is finished. Due to the fact that the maximum outstanding request number is fairly large for a high performance storage system, in the case of a single client, the probability of a request waiting in the client-site queue is small. However, a storage system is often shared by multiple clients in SANs environment. Each client may generate independent workload and cause multiple I/O requests concurrently arriving at the storage system. Furthermore, new I/Os may arrive continually. A number of I/O requests are thus aggregated in the storage system and there is a higher probability that they may exceed the maximum outstanding request number. The extra I/O requests must therefore wait and form a storage system site queue (refers as storage-site queue).

In either case of a client-site queue or storage-site queue, the I/O commands are delayed and considered to be inefficient. If the I/O commands are delivered earlier, the HDD could perform optimal scheduling as studied in [31] and [32]. On the other hand, in a multiple HDDs system, those outstanding I/Os may access a small set of member HDDs only. The other HDDs may stay inactive although I/Os waiting in the queue may need to access them. It is therefore of interest to explore possible method to deliver command earlier.

3.2.2 Internal I/O Queue

As discussed earlier, multiple internal I/O requests may be required by the storage controller to serve an external I/O. Multiplied with possible large number of external I/Os, a fairly large number of internal I/Os may be submitted to the internal initiator HBA's device driver. Depending on different operating systems, the HBA device driver may only be able to handle a limited number of outstanding requests. For example, the windows system can only support up to 255 outstanding requests for one HBA. The rest of the internal I/Os form a waiting queue and these I/O commands are delayed. This is another reason to consider if an I/O command could be sent earlier.

3.2.3 HBA Internal Queue

The HBA device driver issues an HBA specific I/O command to the attached HBA for each internal I/O request. Depending on different implementations, the HBA may concurrently execute a limited number of the I/O commands, with any remaining I/O commands waiting. After execution, the FCP_CMND frame corresponding to each I/O command is placed into the outgoing frame FIFO. It is worthwhile to note that the workload may be a mixture of read and write and thus there may be a number of FCP_DATA frames ahead of the FCP_CMND frame.

3.3 Limitation of Fairness Access Algorithm

3.3.1 FC-AL Operation

The basic elements of a FC-AL loop are those nodes that are connected in a logically unidirectional ring of either fibre optic or copper cable. These nodes in the Fibre Channel terminology are called L_Ports. Each L_Port connects to its proceeding neighbor through receiving fibre (RX), and its succeeding neighbor through transmitting fibre (TX). The control messages and frame data are sent to its next neighbor and received from its previous node. Some messages will travel along the entire loop and come back to the L_Port indicating some specific meaning. Some other messages are only for the designated port and the other ports shall retransmit them in time upon reception. Those control messages are called Ordered Sets, including arbitration (ARB), idles, open (OPN), close (CLS), buffer-ready (R_RDY) and others.

Before an L_Port can send frames to another L_Port, it must arbitrate and win the arbitration for accessing the loop. After the L_Port attains the loop access, it transmits OPN signal that carries the destination port address and becomes the open port. The other port checks the OPN signal and compares the destination address to its own port address. When the addresses match, the port absorbs the OPN signal and becomes the opened port. A logical point-to-point connection is thus established between the open port and the opened port. Frames can then be transferred between the two ports. Either the open or opened port may transmit CLS signals indicating that it desires to close the connection. Upon reception of CLS, the other party may continue to transfer the remaining frames and transmit the CLS to release the loop when the frame transfer completes. The second CLS signal will come to the first port and make the port ready for next operation.

3.3.2 Arbitration Process and Fairness Access Algorithm

Strictly speaking, a FC-AL loop is not a token ring. There is no token for L_Ports to chase for gaining the loop access. Neither is there a central arbitrator that governs the winner of the arbitration if multiple ports are simultaneously arbitrating for the access. The arbitration is actually done in a distributed manner.

An L_Port starts its arbitration by continually transmitting ARB(x) signal, whereby the x in the parenthesis is the address of the port. If there are no other ports that also arbitrate for the access at the same time, the ARB(x) will be retransmitted by all other ports and the L_Port (x) will receives its ARB(x) and the arbitration is won. If another port arbitrates the loop at the same time, it compares its own port address with the x value of the ARB(x) received. If its port address is smaller than x, the port knows that it has higher priority and thus it replaces the ARB(x) with ARB(y) and transmits this signal out to the loop. Upon reception of ARB(y), port x stops transmitting ARB(x) but forwards the ARB(y) since y is smaller than x. The port y will then receive its won ARB(y) and win the arbitration at the end.

From above description, it can be seen that the FC-AL arbitration has priority based on the port address that is called AL_PA (Arbitrated Loop Physical Address) in Fibre Channel term. The port with the smallest AL_PA among all arbitrating ports always wins the arbitration. This may cause problems. Firstly, if the higher priority port continuously accesses the loop, other lower priority ports would not have the chance to gain access and the starvation happens. Secondly, for a busy loop with large number of ports, event though any one port is not likely to arbitrate continuously, multiple higher priority ports may take their turns to arbitrate and cause lower priority port to suffer starvation with increasing probability as the port priority decreases. Thirdly, the pure priority arbitration causes uneven performance among the loop ports even in a less busy loop, since the probability of one port's arbitration being postponed by higher priority ports increases as the port priority decreases. Thus the pure priority based arbitration has starvation and unfairness problems that must be solved.

The Fairness Access Algorithm is used in a FC-AL loop to prevent starvation and unfairness. An L_Port equipped with the Fairness Access Algorithm is not allowed to immediately arbitrate again after it has won the arbitration, unless it discovers that no other ports are arbitrating in the same arbitration window. After winning the arbitration, the L_Port starts sending ARB(F0) signal between frames and monitors its return to test if other ports on the loop desire to gain access to the loop. The ARB(F0) has the lowest priority (F0) among all possible port addresses and thus other L_Ports are given chances to replace the ARB(F0) with their own ARB(X). The moment when the first L_Port receives the ARB(F0) is subsequently delayed if other L_Ports are arbitrating. As long as the ARB(F0) is yet to receive, the winner L_Port is in the same arbitration window and shall continue transmitting the ARB(F0) signal. If there is no more arbitrating, the first L_Port will eventually receive the ARB(F0). Once the L_Port receives the ARB(F0), it then transmits an IDLE signal that indicates the end of the arbitration window and can begin another round of arbitration.

3.3.3 Command Delay by Fairness Access Algorithm

As mentioned earlier, an FC-AL based storage system may consist of a single controller and multiple HDDs on a FC-AL loop. The controller acts as an I/O initiator and the HDDs act as the targets (or the responder). If all ports on the loop, including the controller, are accorded the Fairness Access Algorithm, the controller may not be able to obtain sufficient loop bandwidth to achieve a high level of parallelism among the HDDs and to optimize the overall performance. For example, when the storage

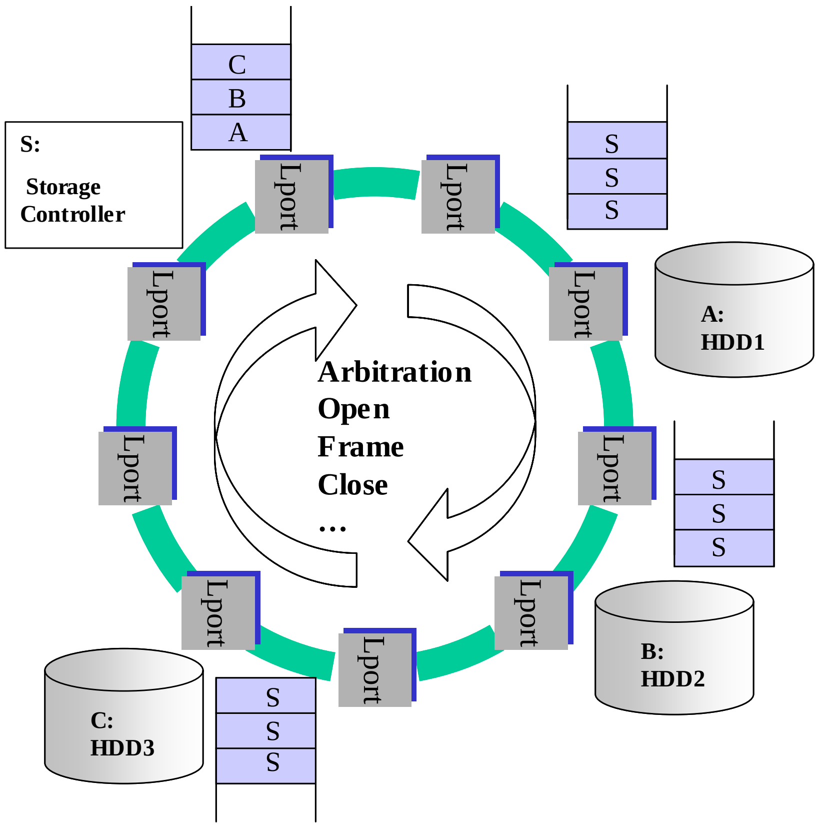


Figure 3.6 Command Delay with Fairness Access Algorithm

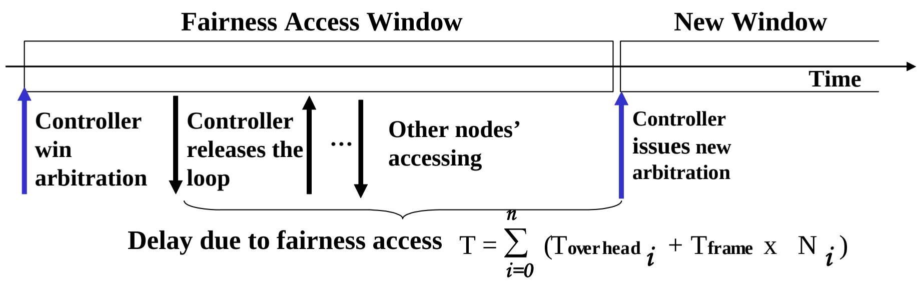

Fairness Access Window
New Window
Time
Controller win arbitration
Controller releases the loop
...
Other nodes' accessing
Controller issues new arbitration
Delay due to fairness access
$T = \sum_{i=0}^{n} (T_{overhead_i} + T_{frame} \times N_i)$

Figure 3.7 Command Delay Timing Model

controller transmits read commands to all HDDs, the HDDs would require some time to prepare the requested data before transmitting data back to the storage controller. Once a disk drive finishes with data transferring, it remains inactive unless it is given another command. If the storage controller follows the fairness algorithm, it will have to wait for all disk drives with pending data before being able to access the loop to send new commands, as shown in Figure 3.6.

The delay of sending I/O commands due to the Fairness Access Algorithm may be very long in some situations. As shown in Figure 3.7, after winning the arbitration and gaining access to the loop, the controller needs to wait for a new access window to send next commands. It is assumed that there are n hard disk drives that are pending for sending read data to the controller and the reading data consists of N_i number of 2KB FC frames for each HDD_i. Collectively, the FC-AL overhead for HDD_i to arbitrate and other overhead is assumed to be $T_{overhead_i}$, and the time for sending a 2K frame over a FC-AL is T_{frame}. The total waiting time for the controller to send a new command can be calculated as:

$$T = \sum_{i=0}^{n} (T_{overhead_i} + T_{frame} \times N_i) \qquad \text{--- Equation (1)}$$

From Equation (1), it can be seen that delay time due to the fairness access is proportion to the I/O size and the number of the HDDs pending for transferring the request data. For 1 G FC-AL loop, the time for transmitting a 2KB size frame is about 20 microseconds (2K/100MBps). The overhead for a HDD to send the first frame may be assumed to be 10 microseconds. For 64KB requests, the total delays may go up to 1 millisecond, when number of HDDs is 16.

3.4 Command-First Algorithm

In view of all above-mentioned potential delays of sending I/O commands, this thesis proposes a new schedule for FC-AL based storage system, which enables the storage system controller to send I/O commands to storage units earlier. Firstly this thesis proposes a Command-First frame buffer management scheme that gives a command frame higher priority than data. Secondly, this thesis proposes to give the controller privilege to arbitrate the loop immediately after an I/O command frame is encountered, regardless of the rule of the Fairness Access Algorithm that prevents the controller to arbitrate again within a same arbitration window. Finally, this thesis proposes to send I/O commands preemptively. These measures are collectively named as the Command-First Algorithm

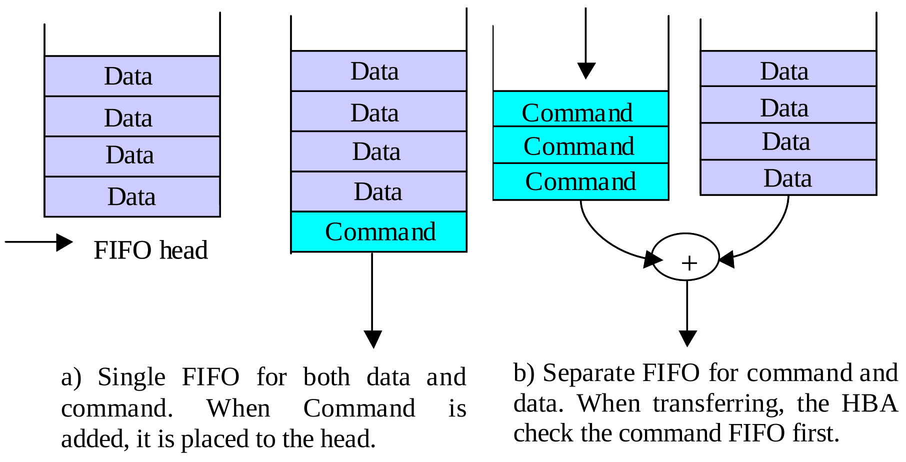

Data
Data
Data
Data
FIFO head
Data
Data
Data
Data
Command
Command
Command
Command
Data
Data
Data
Data
+
a) Single FIFO for both data and command. When Command is added, it is placed to the head.
b) Separate FIFO for command and data. When transferring, the HBA check the command FIFO first.

Figure 3.8 Command Frame Priority Queuing

3.4.1 Command-First FIFO

In the first place, it is proposed to aggressively deliver the I/O commands in system level queue and give the command higher priority than data. The I/O commands shall be transferred to the destination HDD as early as possible. The HBA device driver shall be designed with capability of handling as many I/O commands as possible. The HBA shall give higher priority to the I/O command processing than that of the I/O data. One immediate consideration for giving the I/O command higher priority is to aggressively place the FCP_CMND frame in front of FCP_DATA frame in outgoing "FIFO" buffer of the HBA, or to have a separate higher priority Command FIFO to hold FCP_CMND frames. For the later implementation, the HBA checks the Command FIFO first when it proceeds to transfer frame. Figure 3.8 illustrates the two possible options.

3.4.2 Command-First Arbitration

Secondly, the proposed Command-First Algorithm allows the storage controller to arbitrate immediately after the command is read to send. The controller checks the type of the frame to send. If the frame is a command, the controller switches to unfair access mode, and starts arbitration. Otherwise, the controller remains fairness accessing. The arbitrated loop physical address (AL_PA) of the storage controller is set to lowest value so that the controller has the highest priority of wining the arbitration, in the case of other HDDs also arbitrate the loop.

The Command-First Algorithm differs to the pure unfair access. The FC standard does not compulsorily regulate that all loop device must implement the Fairness Access Algorithm. The Fairness Access Algorithm is optional for loop devices. The standard actually recommends that the storage controller to be implemented with unfair. However, with pure unfair access implementation, the storage controller will always have higher priority to send frames, regardless of the different frame types of data or command. Thus the write performance will tend to be superior to the read since the writing data are sent by the storage controller. The overall effect is likely to be negative. In contrast, the Command-First Algorithm only transfers commands in unfair mode and therefore the data transferring is not affected. Because the command frame is transmitted in a single frame sequence and the size of the command frame is small, the duration of holding the loop is short. Thus the performance penalty of delaying other devices access will be very small.

3.4.3 Preemptive Transferring Command

Since data transfers are always either sent to or from the storage controller in a storage system environment, it is possible for the storage controller to send CLS signal with the intention of closing current communication with a particular HDD. When the controller is opened for the communication, such as the I/O read in half duplex mode, the hard disk drive might continue sending frames to complete the data-transferring even after it has received the CLS signal from the storage controller. The performance gains therefore may be only marginal for the case that the controller is in opened state. However, it would be significant if the storage controller were in open state (the state of a port that has sent OPN signal). In unfair mode, an L_Port that owns the loop has the privilege of sending OPN signal to another port without arbitration, after sending CLS to close current communication.

3.5 Summary

The FC-AL based storage system architecture is discussed in this chapter. The I/O operation from the storage controller to the initiator HBA is described in detail. Then Command-First Algorithm has been proposed with three different levels of Command-First schedule. The Command-First FIFO is to place the command ahead of data so that the command can be sent earlier. The Command-First Arbitration is to force the storage controller to operate in unfair mode for command frame transferring. The preemptive command transferring is to further enforce the storage controller to send the command.

Chapter 4

SANSim and Network Storage System Simulation Modeling

4.1 Introduction

The network storage system simulation model is a part of the SANSim developed in Data Storage Institute. The related publication can be seen in [2],[3] and [4]. The overall structure of the SANSim simulator is briefly introduced at the beginning of this chapter and then the detail implementation of the FC-AL storage system is presented.

4.2 SANSim Overview

SANSim is an event-driven simulation tool for SAN that includes four main modules: an I/O workload module, a host module, a storage network module, and a storage system module, as shown in Figure 4.1.

The I/O workload module generates I/O request streams according to the workload distribution characteristics and sends them to the host modules. The host module encapsulates the I/O workload to the SCSI commands and sends them to the Host Bus Adaptor (HBA) sub-modules. The storage network module simulates the network connectivity, topology and communication mechanism. The FC network module includes three sub-modules: an FC controller module, an FC switch module and an FC communication module. The storage module maps I/O data to the storage devices.

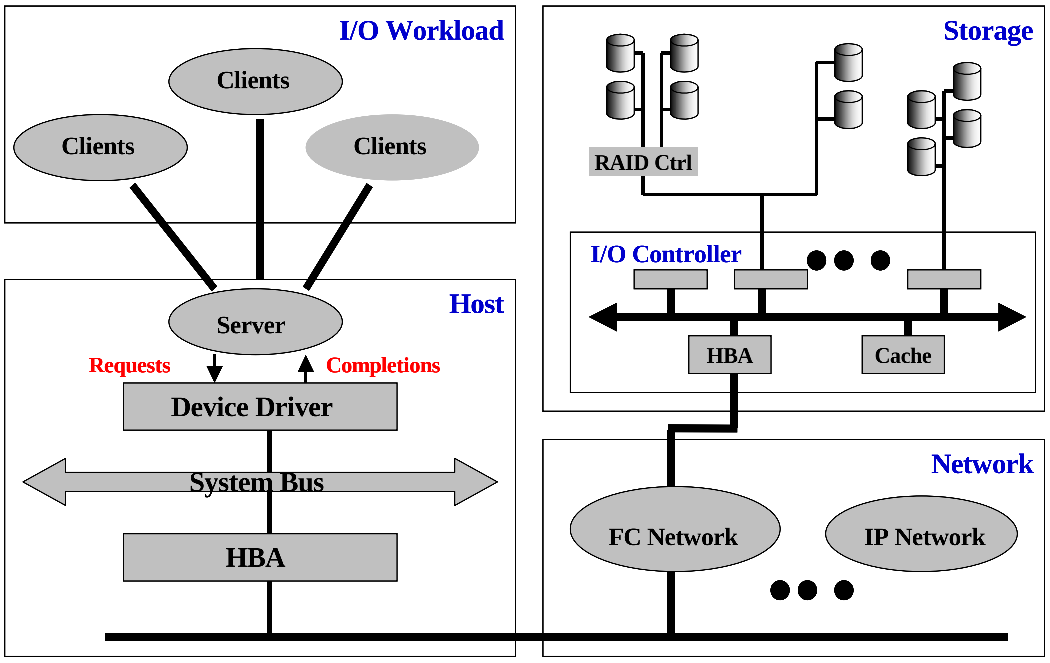


Figure 4.1 SANSim Internal Structure

SANSim is developed in pure standard C. It has been compiled successfully both in Windows and Linux platform. The simulator reads configuration parameters from a user specified input file to establish the simulative SAN configuration, plots measurement data, and stores the results in an output file after the simulation is completed. The simulation duration and the warm-up period can be specified in the input file as well, to control and eliminate the transient bias during the simulation. The configuration parameters for each of the four modules are arranged orderly in the input file. Some other constants are hard-coded in the source code.

4.2.1 I/O Workload Module

The key function of the I/O workload module is to generate I/O request streams according to the workload distribution characteristics and send them to the host modules. It supports both system-traced I/O workload and synthetic (artificial) I/O workload. The synthetic I/O workload is used in this thesis.

An I/O request is determined through five dimensions: the requested pattern, the size distribution, the repeatability, the location distribution and the I/O operations. The workload module is able to generate several basic different arrival patterns such as Poisson arrivals, deterministic time intervals arrival, normal distribution arrivals and so on. It can also generate arrival time to describe the situation where the requests arrive in different rates following different patterns and combinations. Another capability of the workload module is to generate repeatable requests. This scenario is used to define a workload in which some files are more popular than others and consequently accessed more frequently.

4.2.2 Host Module

The host module includes a device driver, a SCSI layer, a system bus, as well as DMA module. The main function of the host module is to encapsulate the I/O workload into the SCSI commands and sends them to the Host HBA sub-modules.

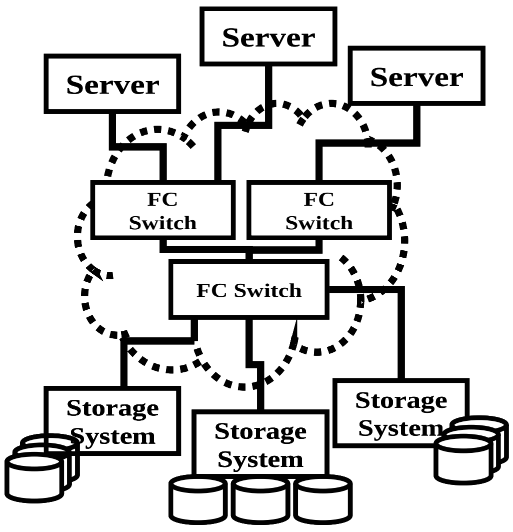
The host modules schedule the I/O requests generated by the workload model, based on various schedule policies. These I/O requests are traced in a circular queue and maintained as outstanding requests. The schedule policy and the number of allowable outstanding requests are configurable during the simulation. When the outstanding requests exceed the maximum allowable number, new arrived I/O requests wait in the device specific queues. These waiting I/O requests may be combined together to form a larger IO access. Performance parameters such as the access time of each request are collected and the statistical results are printed to a file at the end of the simulation.

The host modules maintain a separate queue for each storage device. The I/O requests accessing a particular device will be placed in the corresponding queue. These I/O requests in the queue are marked as either “Waiting” or “Outstanding”. The outstanding requests refer to the I/O request having been issued to the storage device. The waiting requests are those requests that have not been scheduled. The maximum number of outstanding requests depends on the maximum number of the concurrent I/O requests supported by the storage device.

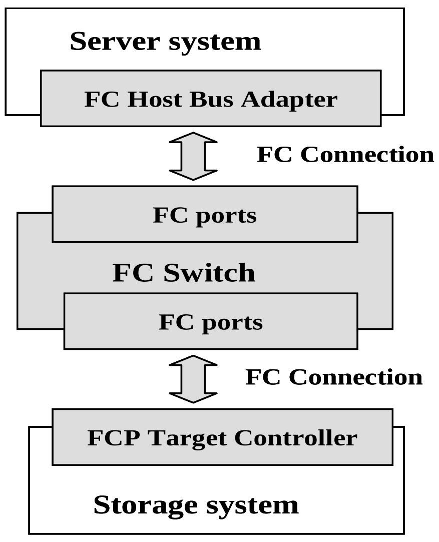
The host module supports a multiple-host configuration. Each host can have a separate I/O workload module to generate I/O independent streams. There is a specific mechanism to identify the I/O requests coming from different hosts. The storage device being accessed respond to corresponding hosts through an identifying bit implemented in the I/O requests data structure.

4.2.3 FC Network Module

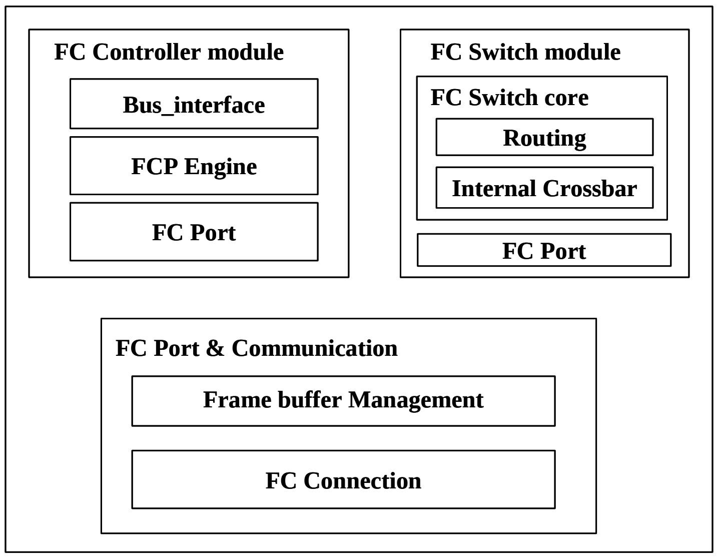
The key function of the FC network module is to simulate the FC connectivity, topology and communication protocol. The FC network module includes three sub-modules: the FC controller module, the FC switch module and the FC port & communication module, as shown in Figure 4.2. The FC controller module simulates the FC controller behaviors of generation FC command or data frames. The FC switch module models all the FC ports, switch architecture, and as well as the routing and flow control. The FC port & communication module transfers FC frames between the FC ports.



(a) An example of FC SAN



(b) Abstracted view



(c) FC network module

Figure 4.2 Fibre Channel Network Modeling in SANSim

4.2.3.1 FC Controller Module

The FC controller module models both initiator and target modes of the FC HBA. The module includes three sub-modules: a bus interface, an FCP engine and a FC port. The bus interface sub-module handles the communication between the device driver and the controller such as DMA and interruption. The FCP engine has the responsibility of constructing different FC frames corresponding to each sequence of

FCP exchange. The FC port is responsible for delivering an FC frame to the destination port on behalf of the communication module.

4.2.3.2 FC Switch Module

SANSim's FC switch module has two sub modules: FC port, and FC switch core. FC port supports F_Ports/FL_Ports and E_Ports. F_Ports/FL_Ports are for Host-Switch and Device-Switch connections, and E_Ports are for Switch-Switch interconnection. The FC_Port's address_ID is unique and is well confined to the FC-SW-2 standard. FC switch core is the switch's control center for frames routing and forwarding. It contains routing and internal cross-bar. If the destination port of requested FC frames is busy, the incoming frames are held in the incoming buffer until they are successfully routed. SANSim uses Dijkstra's algorithm to compute the shortest routing path. The routing table remains constant unless the network connectivity is changed during the simulation. When the network configuration is changed, the switch module re-computes the shortest path.

4.2.3.3 FC Port & Communication Module

The FC port & communication module includes a frame buffer management and FC connection sub modules. The frame buffer management sub module handles all management of incoming and outgoing frame buffers. The FC connection sub module establishes a FC connection between two FC ports for frame transferring. The FC port & communication module supports FC-AL topology that will described in detailed in the following sections.

4.2.4 Storage Module

The main function of the storage module is to map I/O data to the storage devices. Storage modules include modeling of the RAID array, cache management, disk drive, and RAM disk. In the event of disk failure, in a RAID system, the degraded mode and rebuild behavior can also be modeled.

The storage module can be configured with various network interfaces through an interface controller. The I/O requests is extracted from the received frame by the interface controller and passed to the storage controller for accessing the attached storage devices. The RAID controller converts a logical access to multiple device accesses based on the RAID algorithm. The RAID can be configured to support multiple dimension RAID arrays.

4.3 Simulation Modeling of FC-AL Storage System

The simulation model is based on discrete-event driven simulation technique and written in C language. With the discrete-event driven technique [37], a data structure that holds event's timestamp, type as well as other information is used to indicate a particular event happening at certain times. A double precision variable is used to record a high-resolution timestamp in the simulation model so that the FC transmission can be accurately modeled up to nanosecond level. An object or system being simulated is represented by a set of data structures that holds the current status of the system, and a series of actions (or functions) that will be triggered, in accordance with the current status, by various types of event at the event occurring time. An action being triggered by an event may change the system's status or generate a new future event and cause the evolution of the system.

In the remainder of this section, the implementation of FC-AL operation model is first discussed in detail, and then the HBA model is introduced. A system level model is necessary for conducting I/O performance simulation. Since this thesis is focusing on the FC-AL interconnection, other system components are described in rather simplified and abstracted way. This thesis puts more attention to the integration of the FC-AL and HBA model with the system model.

4.3.1 FC-AL Module

Although the FC-AL appears as a separate layer in FC standard, this thesis includes related FC-0 to FC-2 functions into the FC-AL model with the aim of providing a completed model that can be used directly by the HBA model that enables the FCP operation. Also, since this thesis is focusing on the performance of a steady state system, the initialization and error recovery process of the FC-AL operation are omitted from the model.

For the purpose of description, this thesis views the FC-AL model as consisting of four basic units: (1) the signal transmission that simulates the ordered set or frame transmission delay over various types of physical link, (2) the loop port state machine that adopts a simplified version of loop port state machine defined in the FC-AL standard, which omits the initialization and error recovery process, (3) the FC-2

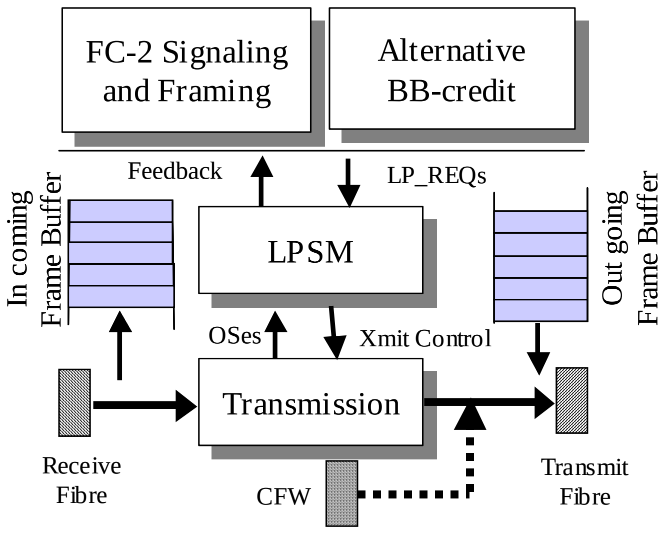


Figure 4.3 FC-AL Simulation Model Structure

signaling and framing that governs the frame transmission, and (4) the alternative BB credit that handles the flow control. The relationships between them are illustrated in Figure 4.3. This thesis discusses the implementation detail of each in following subsections.

4.3.1.1 Signal Transmission

The minimum data unit transmitted in FC is the transmission word with a size of four bytes. Some special transmission words are used as control signals, named by Ordered Set in FC terms, such as IDLE, ARB, OPN, R_RDY, SOF, EOF and others. The SOF and EOF are used to mark the beginning and the ending of a frame. The size of a frame including the payload must be transmission word aligned, i.e., it is four-byte dividable. If the user data to be packitized into a frame is not transmission word aligned, it must be padded-up with dummy data. Based on this characteristic, the simulation model is developed up to transmission word resolution, since the finer level of transmission modeling does not give higher accuracy.

In the first place, the signal transmission model unit takes charge of modeling the delay of Ordered Set (OS) and frame delivery from one port to its next. A transmission word is transmitted as 40 serial bits after the 8bit/10bit coding. With a given parameter of clock rate, the time to complete the transmission is determined. For an example, for 1 G FC, the clock rate is 1.0625 GHz, and thus the transmission time for a single transmission word is about 37.65 nanoseconds (40 divided by 1.0625 G). After the bit stream is transmitted, it takes time to travel along the fibre to reach the next port. For fibre optic cable, this propagation delay is about 5 nanoseconds per meter. The distance between the ports is given in the configuration parameters, and thus the propagation delay is determined. Finally, when a port retransmits a transmission word, an additional delay will be imposed due to the elastic buffering and it is often called

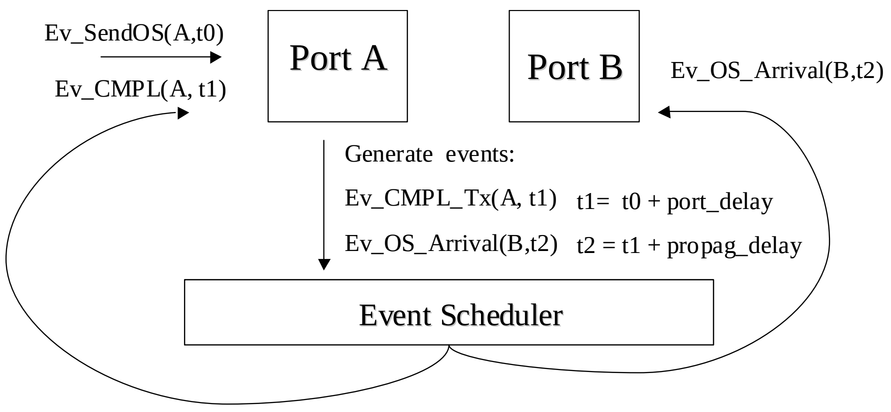


Figure 4.4 Signal Transmission Model

per-port delay. The FC standard specifies that the per-port delay must not greater than six transmission-words transmission time. This delay time actually gives the port a time window to analyze the signal it received and may vary from port to port. In this simulation model, the delay is set to be fixed of about 240 nanoseconds for 1 G media. To sum up, the delay time of a transmission word from one port to the other can be calculated at the beginning with the given configuration parameters.

Based on the timing model described above, a discrete event module is designed as illustrated in Figure 4.4. When simulation time reaches t0, the Ev_SendOS(A,t0) occurs and the Port A is triggered to call the transmission model to generate Ev_CMP_Tx(A,t1) and Ev_OS_Arrival(B,t2) events, which indicates when the data transmission is completed and when the Port B receives the transmitted OS. These two events will be scheduled by the event scheduler according to their time of t1 and t2, and are fed to the corresponding port.

The above model without optimization would have serious problem regarding the computing efficiency. The computing power required for the discrete event simulation is directly proportion to the number of events being processed. In above model, every transmission of one word would require at least three events. For a 2 KB frame, it would require 1536 events (2048 Bytes / 4 Bytes * 3). Moreover, there are always transmission words traveling on the loop to maintain the FC link. For every one second, it would require about 26.6 million transmissions (1.0625 x 10^9 / 40), or about 80 million events. This is too time-consuming and computing expensive. A method is used to solve this problem by implementing "edge-change" simulation, to greatly reduce the number of transaction without losing accuracy.

With the "edge-change" method, every port on the loop records the type and the time of the word being transmitted, marked as CTW (current transmitting word) and CTT(current transmitting time). When a new transmission is issued, the type of new transmission word is checked with the CTW type. If they are same, the transmission is discarded but the CTT is updated. Otherwise, both CTW and CTT are updated and the transmission word is delivered to the next port. For a frame, the model does not distinguish the transmission words between the SOF and EOF and consider from SOF to EOF an unchanged signal. Thus, a frame transmission requires only two "edge-changes", one from CFW (current fill words) to SOF, the other is EOF to CFW. With this method, no other transmission words other than the OSes are required. Additionally, since the state transition of the Loop Port State Machine (LPSM) are

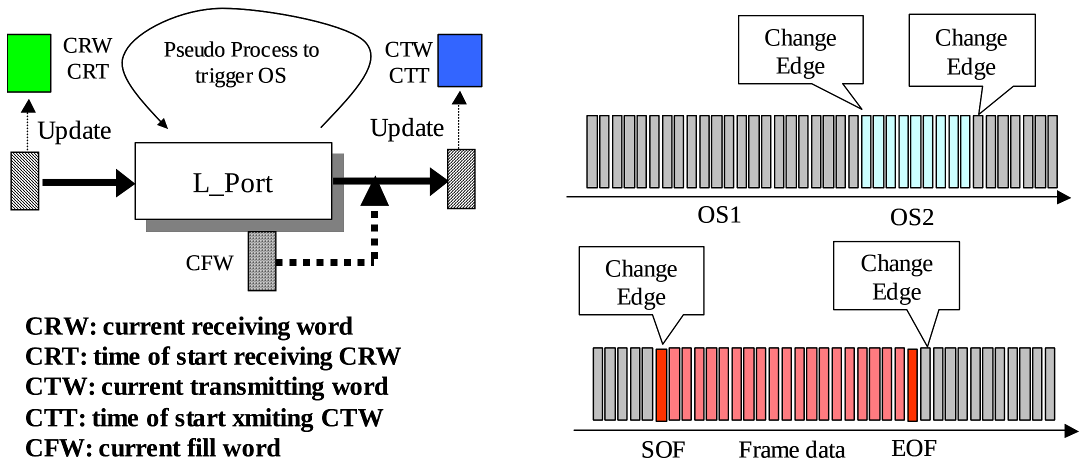

CRW
CRT
Pseudo Process to trigger OS
CTW
CTT
Update
Update
L_Port
CFW
CRW: current receiving word
CRT: time of start receiving CRW
CTW: current transmitting word
CTT: time of start xmiting CTW
CFW: current fill word
Change Edge
Change Edge
OS1
OS2
Change Edge
Change Edge
SOF
Frame data
EOF

Figure 4.5 "Edge-Change" Simulation Techniques often driven by the receiving OSes, it is also critical to maintain a pair of CRW (current receiving word) and CRT(current receiving time). A pseudo process is designed to generate the receiving signal based on the CRW and CRT when it is needed. The “Edge-Change” technique discussed is illustrated in Figure 4.5.

The above discussion provides evidence that high accuracy may be achieved with limited computing power by using the proper modeling method. Other techniques such as clock alignment handling method are omitted from this discussion.

4.3.1.2 Loop Port State Machine

The Loop Port State Machine (LPSM) defines the behavior of an L_Port, and is used to gain access to other L_Ports. Since the model is focusing on the steady state performance, for the purpose of simplicity, only a sub set of the state transition logic is modeled. The states associated with the initialization and error recovery process, namely the Initializing, Old-Port and Open-Init states, are omitted from the model. As show in Figure 4.6, the LPSM module models the remaining eight possible states that an L_Port might possibly transit to and from during normal steady state operation.

With no exception from the basic method of discrete-event driven modeling, the model defines a set of necessary control and status variables in the fields of the L_Port data structure to hold the current status of the L_Port with regard to the LPSM operation. In addition to that, a set of functions is developed to process input events based on the current status and the given event type, according to the LPSM logic defined in the standard. Corresponding to the real world’s input of the LPSM, two basic types of events are defined to trigger the state transitions: the loop port request arrival events and the ordered set arrival events. The loop port requests are issued by the FC-2 signaling and framing model to command the L_Port to perform certain activity, while the OS arrival events are fed by the signal transmission module upon reception of an OS.

It is worth mentioning that the Fairness Access Algorithm is naturally included in the LPSM logic. As specified in the standard, each fairness-access L_Port uses several history variables and controls to determine the start and the end of a fairness access window. Within a same fairness access window, the LPSM of an L_Port will not transit to Arbitrating state again until all conditions are satisfied (corresponding to the fairness access window's being reset). The comprehensive description of the LPSM model is omitted in this thesis. Rather, a simple example of three L_Ports (labeled as port A, port B and port C) configuration is used to illustrate the LPSM operation in the remainder of this subsection.

Assume that all three ports start from Monitoring state and a loop port request of Req(arb_own) is issued to port A. Since port A has not arbitrated before, the Fairness Access Algorithm does not prevent port A to transit from Monitoring to Arbitrating

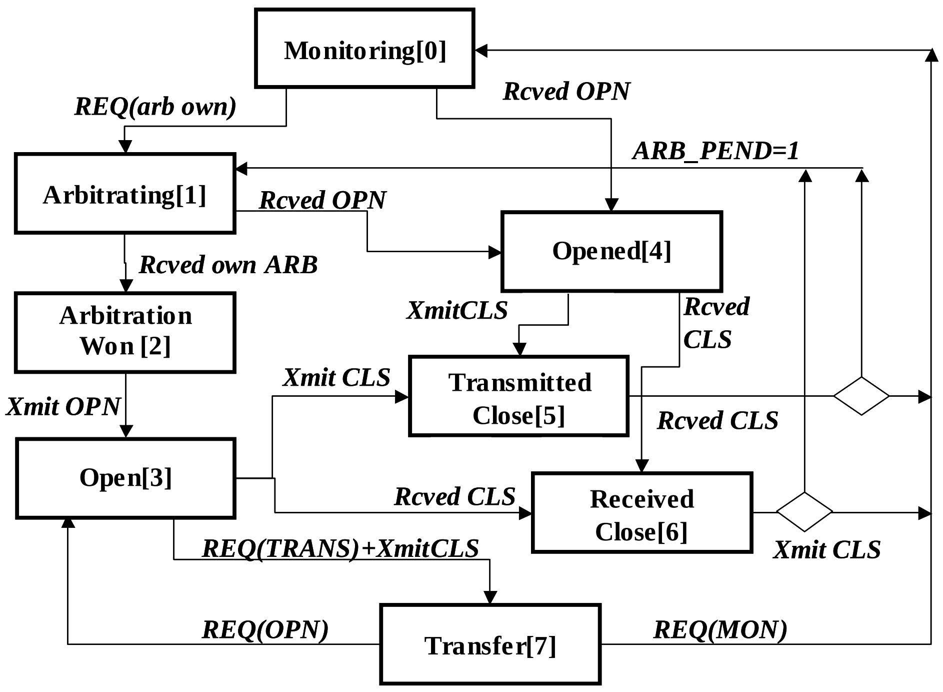


Figure 4.6 Loop Port State Machine

state. Port A then keeps transmitting the ARB(A) when it is in Arbitrating state. Since port B and C are in the Monitoring state, they retransmit the ARB(A) signal. Port A hence receives its own ARB(A), and transits to Arbitration_Won state. The FC-2 signaling and framing model is notified of the arbitration winning and Req(OPN) is issued for frame transferring. An OPN with parameter of destination port address, say the address of port C, is transmitted by port A and the port transits to Open state. While port A is in Open state, it will not retransmit any OS received from its receive fibre over to its transmit fibre. The CFW or other control OSes are used to fill the gap of frame transmission. The control of the frame transmission is then passed to the FC-2 signaling and framing model to determine when is the proper time to send frames. On the other hand, port B retransmits the OPN signal and remains in Monitoring state, while port C recognizes its address's match with the destination address carried by the OPN signal and transits to Opened state from Monitoring. Similar to port A in Open state, port C does not retransmit OS received from receive fibre over to transmit fibre and the control is passed to FC-2 signaling and framing model. Thus, the communication channel is established between port A and port C. If the communication is in full duplex mode, which is determined by the OPN signal, port C may send frames to port A while it receives frames. After all frames are transferred or due to other reasons, either the Open port (A) or the Opened port (C) may receive loop port request of REQ(Close). The LPSM then triggers the port to transmit CLS signal and transit to Transmitted Close state upon completion of sending the CLS. When a port is in Transmitted Close state, it is not allowed to transmit frames again although it may continue to receive frames. Port B forwards the CLS to the other party of the communication, being the Open port or the Opened port. Upon reception of this CLS signal, the port transits to Received Close state. While in the Received Close state, the port may continue to send frames until all frames are transmitted or up to the buffer-credit’s limit. It is noted that the opposite port does not transmit any more buffer-credits after it is in Transmitted Close state. The Received Close port then transmits CLS and transits back to Monitoring state and the Transmitted Close port consequently follows when it receives the CLS if there is no arbitration request pending. Otherwise, it transits to Arbitrating state directly to perform the arbitration.

Finally, the Transfer state is discussed to mark the end of this example. It is assumed that port A has an additional task to send a frame to port B and its communication with port C remains unchanged as in previous description. Upon completion of the task with port C, port A receives REQ(Transfer) from FC-2 signaling and framing model and thus it transits to Transfer state after the CLS is sent. Port A, in Transfer state, will eventually receive CLS signal from port C and the FC-2 signaling and framing model is informed. A REQ(OPN) is then issued since it has a remaining task to deal with port B, other than a REQ(MON) is issued which causes transition from Transfer to Monitoring. As the REQ(OPN) is received, port A sends OPN with destination address of port B and transits to Open state. And thus, the arbitration is saved for the second task. It is noted that the Transfer state is only available in the unfair access mode.

4.3.1.3 FC-2 Signaling and Framing

The FC-2 signaling and framing module take over the control of an L_Port for frame transferring after the communication channel has been established between the Open and Opened port pair. It is assumed that the outgoing frames are constructed on-the-fly by other functions of the HBA model and they are placed in the outgoing frame buffer and ready for transferring. It is further assumed that the conditional checking and other computing overhead associated with the FC-2 Signaling and

Framing are sufficiently small and negligible. The HBA overheads such as command execution are defined separately in the HBA model unit.

Since the frames can only be sent in Open state, or possibly in Opened or Received Close states for full duplex cases, or Received Close state transited from Open state for half duplex, three different functions are developed to simulate the control behavior. Nevertheless, the three functions have similar control logic. They must determine if there are waiting frames in the outgoing frame buffer that are heading to the connected port, if the six ordered-sets frame gap is satisfied, if the credit is available, and other conditions before sending a frame. If it is ready to send a frame, the control instructs the signal transmission unit to send the SOF signal and then determines when to send EOF signal based on the frame size and the processes shall not send any other Signal during the frame transmission. In between the frame, they must also determine when to send exactly one R_RDY for a newly available incoming frame buffer slot and separate the consecutive two R_RDYs signal with at least two CFWs. When there is no more frame to send, the control shall issue loop port request of REQ(Close) to the LPSM for closing current connection. From the above description, it can be seen that the control actions are dependent on the complex combination of conditions and the conditions are change rapidly as the simulation progresses. To increase the efficiency, the control function is developed as a virtual process, which may be activated from sleep state by certain events, e.g. a frame is newly en-queued to the outgoing frame buffer, or a received frame is de-queued from the incoming frame buffer to make the buffer slot available; or transit from active to sleep state when frame sending conditions are unforeseeable.

4.3.1.4 Alternative Buffer-to-Buffer Flow Control

The alternative buffer-to-buffer is used on a FC-AL loop to ensure that the source port does not send frames more than the number of available frame buffer slots that the destination port has. The alternative buffer-to-buffer flow control differs to the standard buffer-to-buffer flow control. With standard buffer-to-buffer flow control in point-to-point or Fabric topology, the two communication parties establish BB_Credit over each other after initialization login process. The partnership between the communication pair in this case remains fairly static at “one-to-one”. It is therefore safe for the standard buffer-to-buffer flow control to make use of BB_Credit to check if the port is in credit for sending frame. When a frame is sent, the BB_Credit decrements by one and when an R_RDY is received, the BB_Credit increments by one. As long as the BB_Credit is greater than zero, the frame can be safely sent. In contrast, the BB_Creidt is not directly applicable to a FC-AL loop. An L_Port may be opened by various other L_Ports over time and the BB_Credit can no longer reflect the latest available buffer space of the destination port that has just been opened by another L_Port.

Differing to the standard BB Credit, FC-AL uses an “alternative” buffer-to-buffer flow control. During loop initialization, each L_Port advertises its own LoginBBCredit that guarantees the number of buffer slots available, and records other ports’ LoginBBCredit. After that, when an L_Port opens or is opened for communication, it looks up its record for the opposite port’s LoginBBCredit. If the LoginBBCredit is greater than zero, the port can start sending frames up to that number at the beginning phase. During this phase, the R_RDYs received are discarded up to the equal number of LoginBBCredit to make the credit balance. When more R_RDYs are received, the port records the additional credits. After the LoginBBCredit is fully used and equal

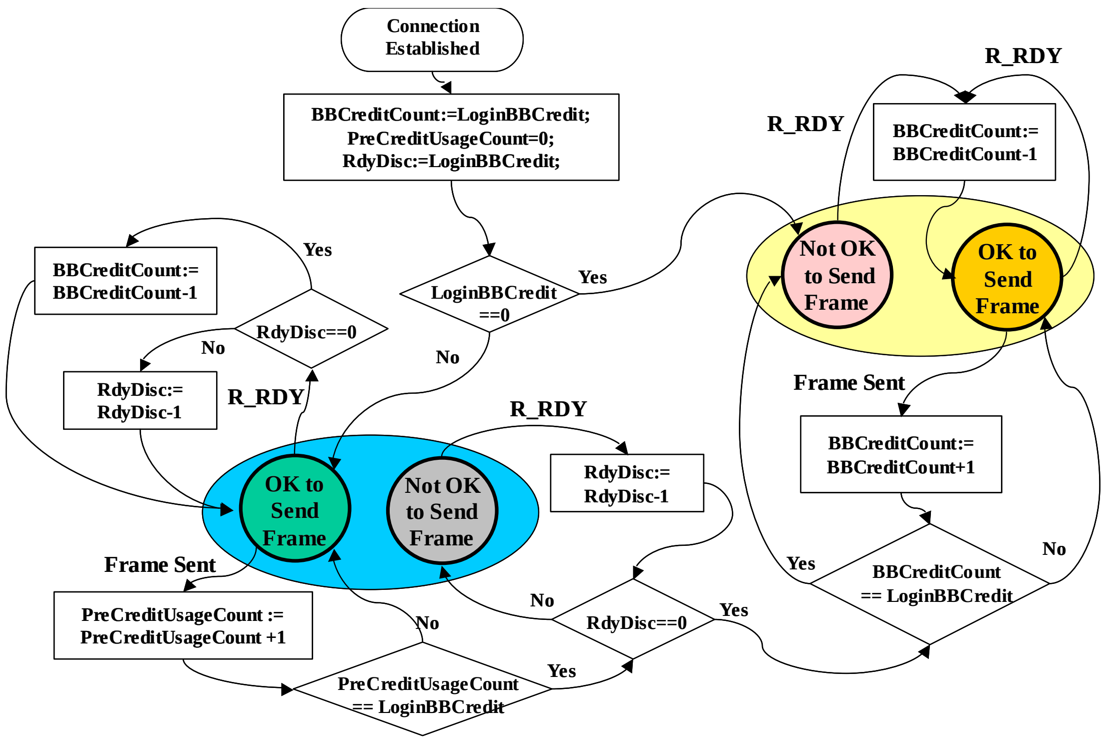
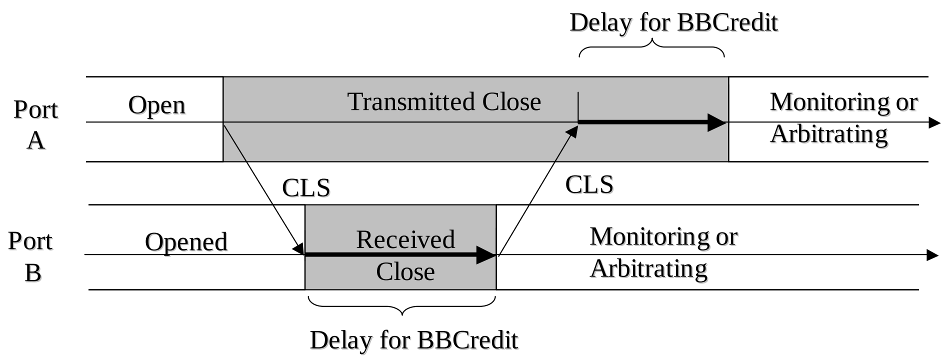

Connection Established
BBCreditCount:=LoginBBCredit; PreCreditUsageCount=0; RdyDisc:=LoginBBCredit;
LoginBBCredit ==0
Yes
No
RdyDisc==0
Yes
No
BBCreditCount:= BBCreditCount-1
RdyDisc:= RdyDisc-1
R_RDY
OK to Send Frame
Not OK to Send Frame
R_RDY
RdyDisc:= RdyDisc-1
Frame Sent
PreCreditUsageCount := PreCreditUsageCount +1
PreCreditUsageCount == LoginBBCredit
No
Yes
RdyDisc==0
No
Yes
R_RDY
R_RDY
BBCreditCount:= BBCreditCount-1
Not OK to Send Frame
OK to Send Frame
Frame Sent
BBCreditCount:= BBCreditCount+1
BBCreditCount == LoginBBCredit
Yes
No

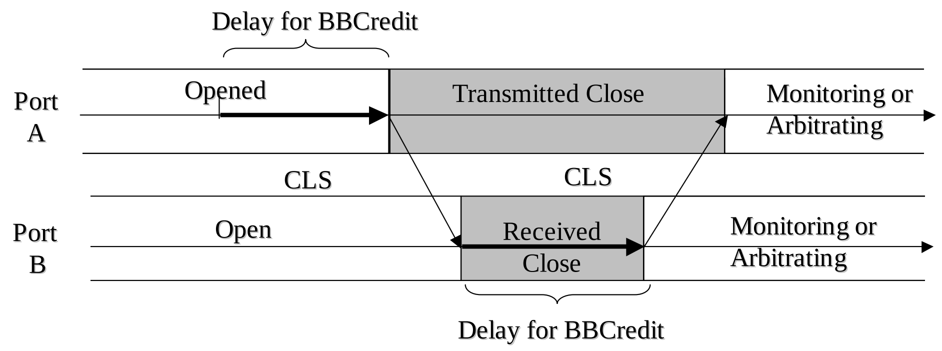
Figure 4.7 Alternative Buffer-To-Buffer Flow Control

number of R_RDY has been discarded, the second phase begins. By then, the port maintains the credit in the way similar to the standard buffer-to-buffer flow control.

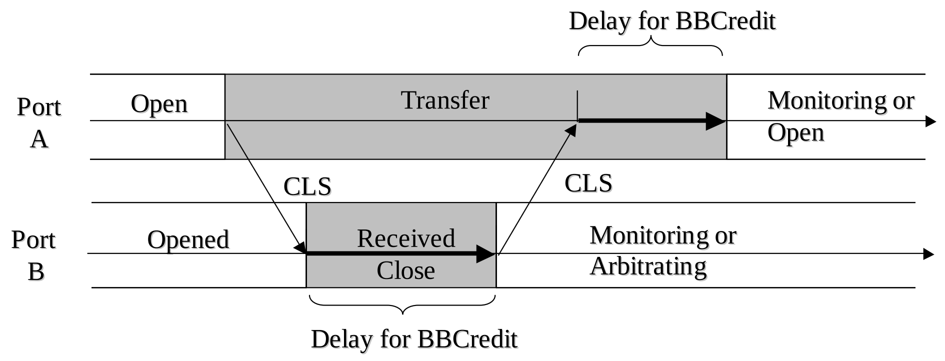
The Alternative Buffer-to-Buffer Flow Control logic is implemented as in Figure 4.7. Two types of event are used in the diagram, the "R_RDY" indicating the event that a R_RDY is received and the "Frame Sent" indicating the event that a frame has just been sent. The control could be in either "OK to Send Frame" or "Not OK to Send Frame" states and transits from left-hand side of the beginning phase to the right hand side of the continue phase. Three control variables are initialized when a communication channel is established: the BBCreditCount is assigned with the LoginBBCredit; the PreCreditUsedCount that is used to count the number LoginBBCredit that has been used is set to zero; and the RdyDisc that indicates the remaining number of R_RDY to be discarded is assigned with the LoginBBCredit. From then on, the control reacts to the event for changing the value of those control variables or transits to different states.



(a) Open port transmits CLS to transit to Transmitted Close state



(b) Opened port transmits CLS to transit to Transmitted Close state



(c) Opened port transmits CLS to transit to Transfer state

Figure 4.8 State Transition Delay for Alternative BB Credit

To guarantee a non-zero LoginBBCredit number of available space in its incoming frame buffer, an L_Port may need to delay some of its LPSM state transition. If the L_Port is the Open port and has transited to Transmitted Close state after sending CLS, it shall make sure it has enough free space in the buffer before it transits to Monitoring or Arbitrating state upon receiving CLS from the opposite port. If the free buffer space is not sufficient, the port waits until enough frames are processed and removed from the incoming frame buffer. Similarly, for an Open port that has transited to Transfer state, it should ensure enough free buffer space before it transits to Open state again or to Monitoring state upon receiving CLS. For an L_Port that is in Received Close state, it also needs to ensure the free buffer space before it transmits CLS signal, regardless if it is the Open port or the Opened port. The last scenario is that an L_Port in the Opened state wishes to close the communication. The L_Port needs to ensure the guaranteed free buffer space as well but it needs to consider the credits that have been given to the opposite port for the reason that the opposite port may continue transferring frame upon those credits. It may choose to inform the FC-2 signaling and framing unit to stop sending R_RDY, otherwise it may never get enough space which is being guaranteed for the next communication. Figure 4.8 illustrates the timing diagram for the above three different situations.

4.3.2 FC HBA Module

The FC HBA module simulates the behavior of the FC host adapter used in a storage system, as the FCP Initiator, and the HDD interface, as the FCP Target. For simplicity, the FCP Target is assumed to have identical internal structure to the FCP

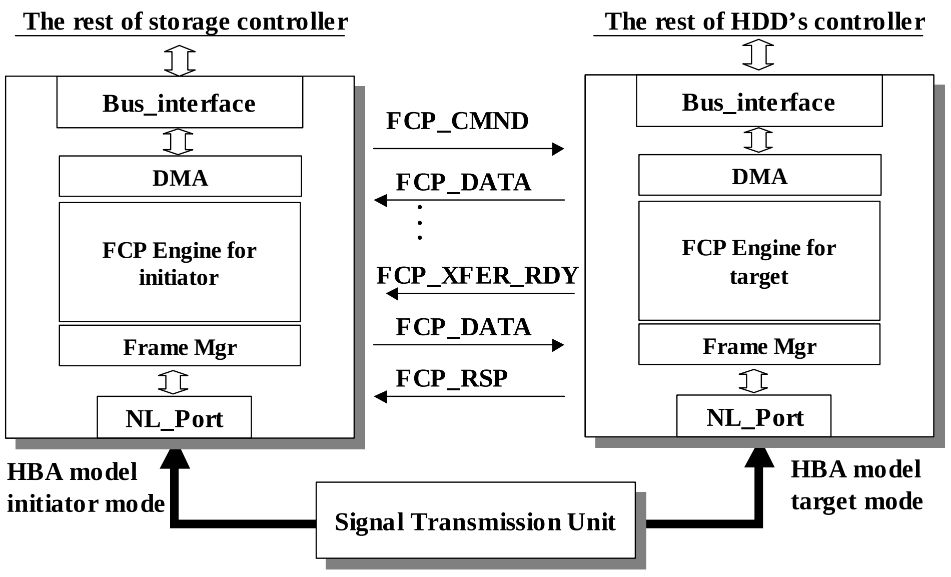


Figure 4.9 FC HBA Model Structure

Initiator except the FCP processing engine, as shown in Figure 4.9. Overall, the HBA module consists of several function sets such as bus interface, DMA, FCP engine, frame manager and the L_Port model developed in the previous section, corresponding to the HBA's internal basic operation. The signal transmission unit models the connection between the FCP Initiator and the FCP Target. The internal structure of the model in FCP Initiator mode and FCP Target mode are described as following.

4.3.2.1 FCP Operation Protocol

The FCP protocol maps each SCSI I/O into one FCP I/O Exchange that consists of several FC sequences corresponding to each SCSI command and data transferring phase. Four types of FCP Information Unit (FCP IU) are defined for SCSI read and write transactions, namely FCP_CMND, FCP_DATA, FCP_XFER_RDY and FCP_RSP. These FCP IUs are packed together with the directive frame headers into FC frames and transferred in sequences. As their names imply, the FCP_CMND carries a SCSI I/O

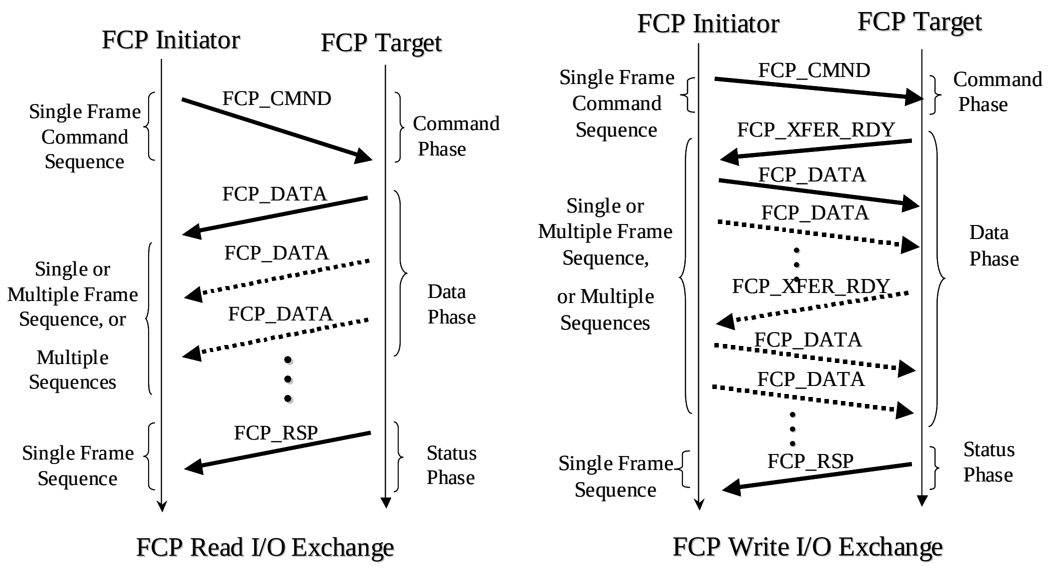


Figure 4.10 FCP I/O Operation Protocol

Command; the FCP_DATA is used to transfer data; the FCP_XFER_RDY indicates that the target is ready to receive a certain amount of data; and the FCP_RSP reports the completion status of the SCSI I/O. The FCP_CMND is issued from an FCP Initiator to an FCP Target.

The FCP protocol for SCSI read or write transaction is illustrated in Figure 4.10. As shown in the diagram, both read or write operations can be divided into in three phases: the command phase, the data phase and the status phase. During the command phase, the FCP Initiator initiates the I/O by sending FCP_CMND in a single frame sequence to the FCP target. Upon reception of FCP_CMND, the FCP Target prepares the requested data for read, or allocates data buffer to receive requested data for write. After the read data, or the data buffer for write is ready, the data phase starts. For read, the FCP target sends the requested data in one or more FCP_DATAs. Depending on various implementations, the FCP Target may prepare the requested read data in multiple parts, one after the other, and having a time space in between. In such cases, the FCP_DATAs may be sent in multiple data sequences. For write, the FCP Target sends FCP_XFER_RDY with a parameter of FCP_BURST_LEN specifying the size of the writing data buffer. The FCP Initiator then sends one or more FCP_DATAs up to the size of FCP_BURST_LEN given in the FCP_XFER_RDY received. If the write I/O size is greater than the FCP_BURST_LEN, additional FCP_XFER_RDYs is required after the FCP Initiator fully transmits all FCP_DATA requested by previous FCP_XFER_RDY. After all requested data are transferred, the transaction comes to the status phase. The FCP Target finally generates the FCP_RSP and sends it to the FCP Initiator in a Single Frame Sequence.

4.3.2.2 FCP Initiator Mode

Figure 4.11 shows the functional block diagram of the HBA module in FCP Initiator mode. The FCP Main Processor receives HBA commands or other short control messages through the Interrupt & Messenger from the Bus Interface. After interpretation, the FCP Main Processor recognizes that an I/O command is issued. It then takes some time to allocate necessary resource. After completion, it issues a DMA_REQ into DMA Queue to fetch the I/O information from the host system (not shown in the diagram) through the Bus Interface by the DMA Transfer. The DMA Scheduler de-queues one entry from DMA Queue based on various disciplines, such as Round Robin, First-Come-First-Serve (FCFS) or other. The DMA Transfer is then instructed to DMA the requested I/O information from the host system. The DMA Transfer supports bidirectional transfer, i.e., both from and to the host system. Once the complete I/O information is retrieved, the FCP Main Processor is informed and the Outgoing Frame Construction is commanded to build the FCP_CMND frame that

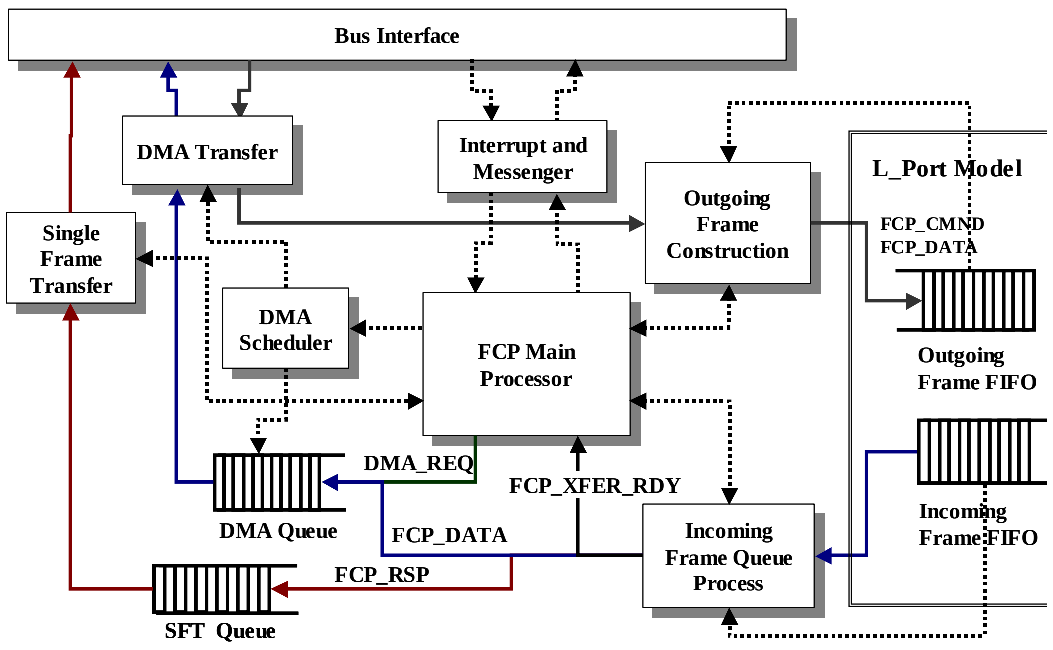


Figure 4.11 FCP Initiator Mode HBA Model Structure

carries the SCSI I/O command and places it into the Outgoing Frame FIFO through the interface provided by the L_Port model.

When an FC frame is received in the Incoming Frame FIFO, the Incoming Frame Queue Process is activated, if it is in sleep, to process the incoming frames. Once activated, the Incoming Frame Queue Process continues processing until the FIFO is empty. It analyzes the frame retrieved from the Incoming Frame FIFO through the L_Port model interface. The FCP_DATA frames are de-encapsulated and placed in DMA Queue for DMA transfer, while frames with other types are directed to the SFT Queue to be transferred to the designated host system memory location through Single Frame Transfer. Since this thesis is focusing on steady-state modeling, it is assumed that only FCP_RSP frames are transferred in this mode. In the case of In-Order-Delivery, the FCP_RSP shall be received after all requested data has been transferred. The Single Frame Transfer handles the SFT Queue in FCFS order and informs the FCP Main Processor upon completion. An interrupt is then issued through the Interrupt & Messenger to inform the host system to process the FCP_RSP that indicates the completion of an I/O. For FCP write, the FCP_XFER_RDY received is passed to the FCP Main Processor for handling data transfer. DMA requests are generated to transfer the writing data from the host system memory. The Outgoing Frame Construction then, with assistance of FCP Main Process, encapsulates the data fetched by the DMA Transfer into FCP_DATA frames that are placed into the Outgoing Frame FIFO.

4.3.2.3 FCP Target Mode

The functional block diagram of the HBA module in FCP Target mode is shown in Figure 4.12. It can be seen that the FCP Target Mode HBA module has identical internal bocks to the FCP Initiator mode. It is noted that the FC interface may be tightly connected to the disk controller through other means in an actual FC hard disk drive. However, since the model supports simulation-time configuration, this means that the model can be configured with various parameter when simulation starts, such as using zero overhead of command execution to eliminate some non-actual-exist function, thus it still be able to model the HDD's interface function.

In FCP Target Mode, when FCP_CMND frames are received in the Incoming Frame FIFO, the Incoming Frame Queue places them into the SFT Queue to be transferred to the HDD controller (not shown in the diagram) through Single Frame Transfer. The HDD controller then executes the SCSI command embeded in FCP_CMND. If it is a read, the requested data is read from the media (HDA) and placed in read buffer. Otherwise, enough free buffer space is allocated for writing data. After completion, the HDD controller informs the FCP Main Processor through the Interrupt & Messenger. For read, the FCP Main Processor then issues DMA requests and places the data in DMA queue to be handled by the DMA Scheduler to transfer

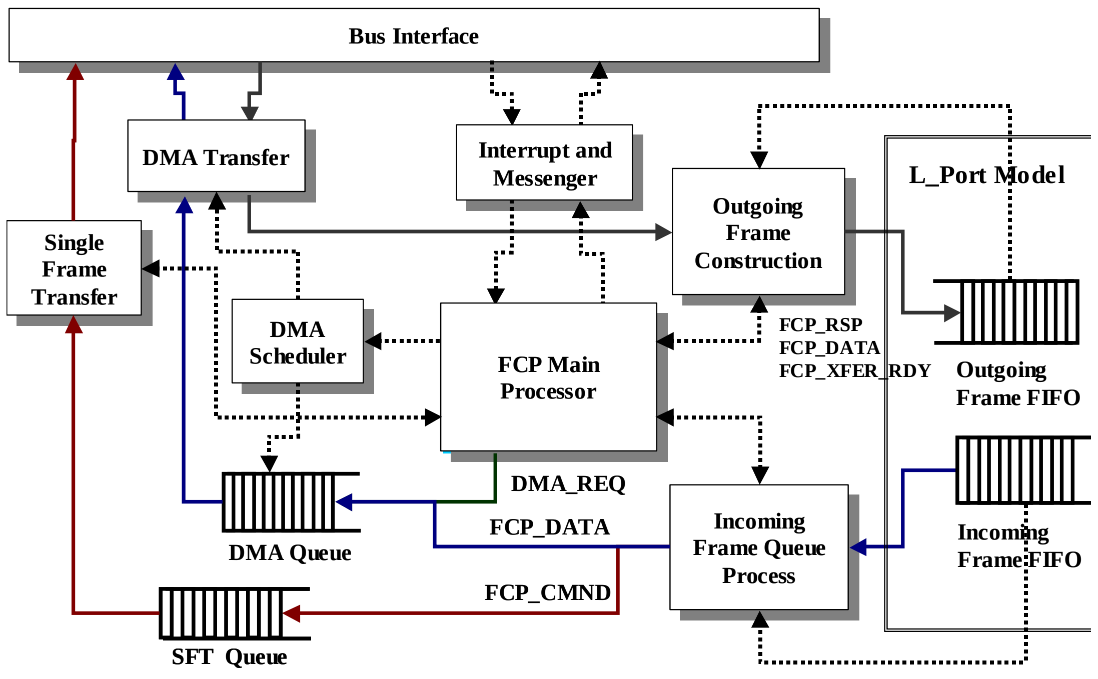


Figure 4.12 FCP Target Mode HBA Model Structure

read data from the read buffer. The read data is then en-capsulated into FCP_DATA frames to be placed into the Outgoing Frame FIFO by the Outgoing Frame Construction. When data transfer is completed, an FCP_RSP frame is generated and placed into the Outgoing Frame FIFO indicating the I/O has been completed. For write, the FCP Main Proessor assists the Outgoing Frame Construciton to build the FCP_XFER_RDY frame based on the informaiton received from the HDD controller, and places it into the Outgoing Frame FIFO. When FCP_DATA is received, it is DMAed to the write buffer that has been allocated. If the last FCP_DATA requested by a FCP_XFER_RDY is received but more data is writing, the HDD controller is informed to allocate additional write buffer. A new FCP_XFER_RDY is then contructed and placed in the Outgoing Frame FIFO. Otherwise, if all data has been received, an FCP_RSP is generated and en-queued to the Outgoing Frame FIFO to indicate the completion of the write.

4.3.3 HBA Device Driver Module

Before a system model can be established, two more components module, the device driver for the Initiator HBA and the hard disk drive firmware function are developed to simulate the I/O behavior of the rest of the systems that communicate with the HBAs through system bus. After these additional two components are presented, all corresponding component modules can then integrated into the system model.

4.3.3.1 FC HBA Initiator Device Driver

For simplification, the FC HBA Initiator device driver is modeled as shown in Figure 4.13. I/O requests are submitted to the device driver model through the arrival queue. The Arrival Process takes some time to establish all required data structure, such as the I/O request package for holding the information of the I/O request. After completion, the request package is then en-queued in the I/O Request Information Queue that physically holds the I/O information in host system memory. The information of the I/O arrival is sent through the Message Passing to the HBA model. The HBA retrieves the I/O request package through DMA from the I/O Request Info Queue, and processes the I/O request. During data phase, the HBA model may fetch data from the Write Buffer, or send data to the Read Buffer. After completion of data transfer, the HBA model notifies the HBA device driver through interrupt indicating

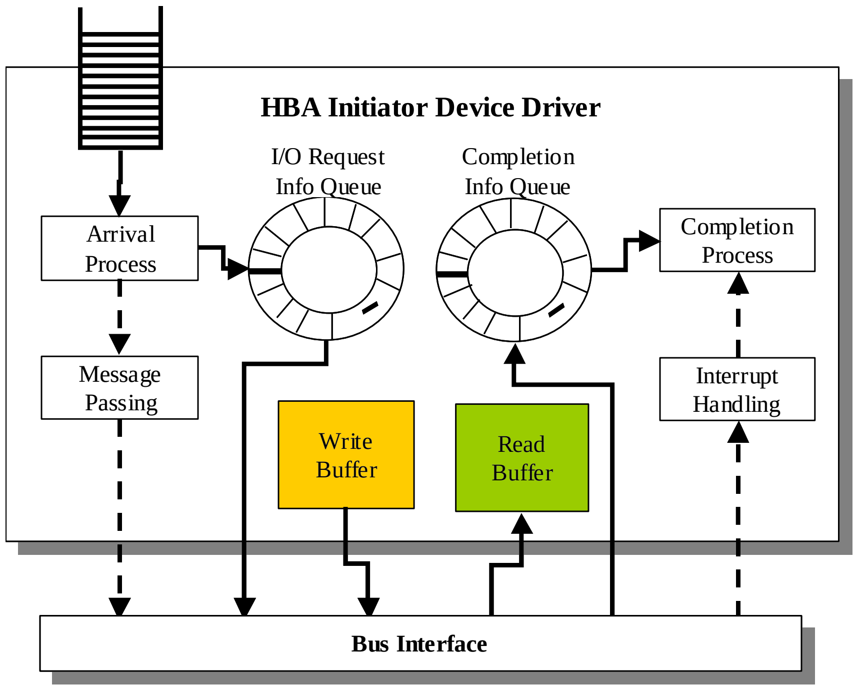


Figure 4.13 FC HBA Device Driver Model

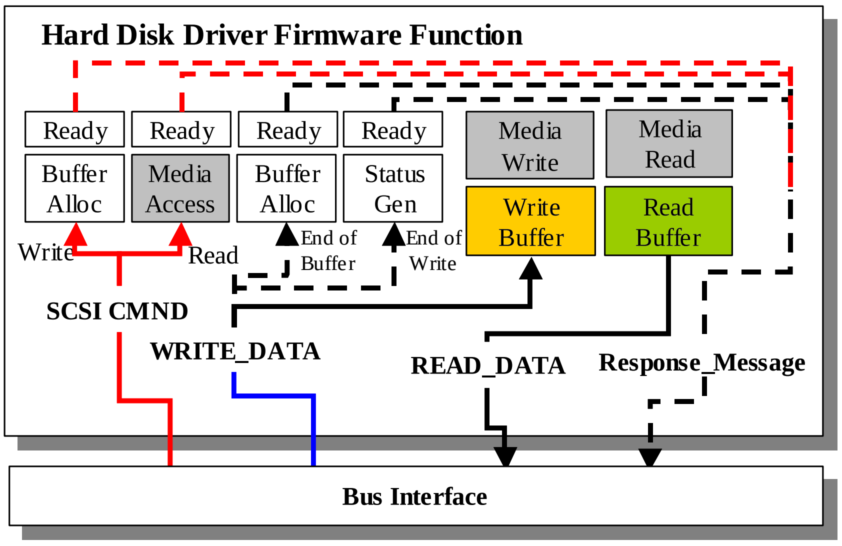


Figure 4.14 HDD Firmware Function Model

that some I/Os have been completed, and the completion messages are available in the Completion Info Queue. The Completion Process examines the I/O completion status in the Completion Info Queue and reports the completion.

4.3.3.2 Hard Disk Drive Firmware for FC Interface

The HDD firmware function with regard to the FC target interface is modeled as shown in Figure 4.14. Detailed description of the model is omitted in this thesis.

4.3.4 Model Integration

Since this thesis is aiming at the system level I/O performance, it is required to integrate the component models into a system model. Figure 4.15 shows an example of the system model consisting of a workload Generator to generate certain pattern I/O workload, a storage controller hosting an FCP Initiator HBA Device Driver, an FCP Initiator HBA model attached to the HBA device driver through the I/O bus, and three FCP Targets interfacing to each attached HDD. I/O requests generated by the workload generator arrive at the storage controller for execution. The storage controller schedules and submits corresponding internal I/O requests to the HBA Device Driver.

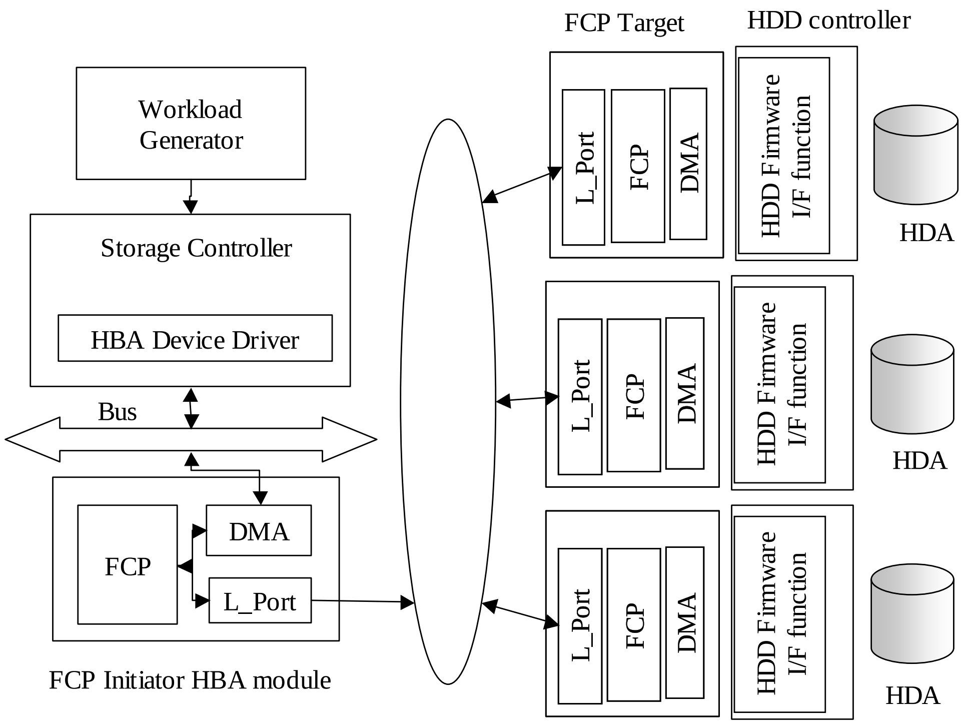


Figure 4.15 System Level Integration

The FCP Initiator HBA then initiates FCP I/O operations for these internal I/O requests. The FCP Target, with support from HDD controller, response to those FCP I/O operations.

4.4 Summary

This chapter has implemented a set of simulation models for FC-AL based storage system in the way of “bottom-up” developing approach. The transmission level modeling is first introduced, and then the L_Port’s functionalities are simulated. After the L_Port model is ready, the FCP HBA model is developed. Lastly, the system level model is established and ready for validation. .

Chapter 5

Calibration and Validation

5.1 Transmission Calibrations

In order to calibrate the FC-AL model, the Finisar GTX-P1000 Fibre Channel Analyzer has been used to track the actual communication between a host system and the storage target. As shown in Figure 5.1, the FC analyzer logically sets two monitors to watch different directional traffics on the two transmission links. The traffic monitors are so designed that they do not interfere with the original communication, in the way of “signal coupling”, instead of copy. The transmissions can be recorded and stored in memory buffer, up to the pre-defined size. When the tracking end time is reached, the recording is stopped. The analyzer software then generates a time-stamped event list according to the raw data and presents them in various formats. A number of traces under different I/O workload pattern have been produced using the analyzer.

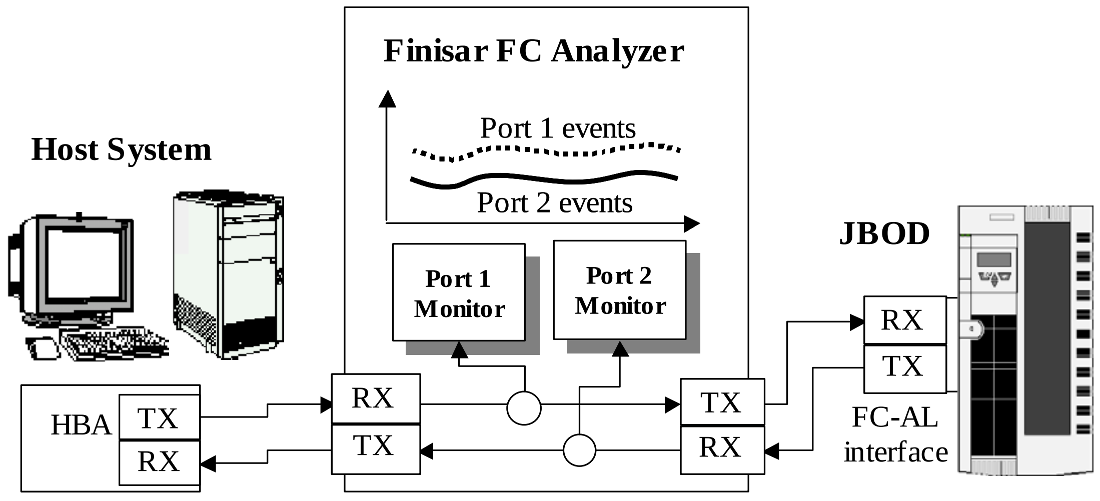


Figure 5.1 Finisar GTX-P1000 Analyzer Logical Configuration

These traces are in the format of readable plain text. Each event in the trace starts with a nanosecond-resolution time stamp that indicates the beginning time of the transmission, followed by a record of multiple same ordered sets or a frame transmitted during the past period of time. The time difference between previous event and current event with the same port (port 1 or port 2) is also given. If the transmission are Ordered Sets, the OS type with parameter such as OPN (x,y), and the OS counts are presented. If it is a frame, the FCP type and frame size as well as other frame header information are abstracted. Figure 5.2 shows a segment of a trace in such format.

hhh:mm.ss.ms_us_ns	DT/Port	Port	Count	OS	Size	RCtl
000:00:00.998_306_903	0.217	FC_Port 2	1	CLS	4	
000:00:00.998_306_941	0.038	FC_Port 2	1021	Idle	4	
000:00:00.998_307_533	40.208	FC_Port 1	1	CLS	4	
000:00:00.998_307_571	0.038	FC_Port 1	1013	Idle	4	
000:00:00.998_345_378	38.438	FC_Port 2	39	ARB(E1,E1)	4	
000:00:00.998_345_708	38.138	FC_Port 1	41	ARB(E1,E1)	4	
000:00:00.998_346_856	1.478	FC_Port 2	1	OPN(E2,E1)	4	
000:00:00.998_346_894	0.038	FC_Port 2	3	ARB(F0,F0)	4	
000:00:00.998_347_006	0.112	FC_Port 2	1	R_Rdy	4	
000:00:00.998_347_044	0.038	FC_Port 2	2	ARB(F0,F0)	4	
000:00:00.998_347_119	0.075	FC_Port 2	1	R_Rdy	4	
000:00:00.998_347_157	0.038	FC_Port 2	2	ARB(F0,F0)	4	
000:00:00.998_347_232	0.075	FC_Port 2	1	R_Rdy	4	
000:00:00.998_347_261	1.553	FC_Port 1	2	ARB(F0,F0)	4	
000:00:00.998_347_269	0.038	FC_Port 2	2	ARB(F0,F0)	4	
000:00:00.998_347_336	0.075	FC_Port 1	1	R_Rdy	4	
000:00:00.998_347_345	0.075	FC_Port 2	1	R_Rdy	4	
000:00:00.998_347_374	0.038	FC_Port 1	38	ARB(F0,F0)	4	
000:00:00.998_347_382	0.038	FC_Port 2	28	ARB(F0,F0)	4	
000:00:00.998_348_408	1.026	FC_Port 2	5	Idle	4	
000:00:00.998_348_626	0.218	FC_Port 2	1	SOFi3	60	FC4Status
000:00:00.998_348_776	1.402	FC_Port 1	28	Idle	4	
000:00:00.998_349_191	0.565	FC_Port 2	6	Idle	4	
000:00:00.998_349_406	0.215	FC_Port 2	1	CLS	4	
000:00:00.998_349_444	0.038	FC_Port 2	3235	Idle	4	
000:00:00.998_349_848	1.073	FC_Port 1	1	R_Rdy	4	
000:00:00.998_349_886	0.038	FC_Port 1	6	Idle	4	
000:00:00.998_350_104	0.217	FC_Port 1	1	CLS	4	
000:00:00.998_350_141	0.038	FC_Port 1	3186	Idle	4	
000:00:00.998_470_105	119.964	FC_Port 1	39	ARB(E2,E2)	4	
000:00:00.998_471_215	121.771	FC_Port 2	41	ARB(E2,E2)	4	

Figure 5.2 Fibre Channel Analyzer Trace Format

During the model-debugging period, these traces are used from time to time to calibrate the timing. Various kinds of software event watchers are developed in the models and can be turn on during compile-time to produce a simulative event trace. Figure 5.3 shows an example of port watcher's result. It has similar content to an actual trace as shown in Figure 5.2. The port watcher also prints the LPSM state of the L_Port at event occurring time. Through examining these events, the correctness of the protocol implemented in the simulation models is validated. For some examples, the six-ordered-sets-gaps requirement between frames is fulfilled; the R_RDY, OPN, CLS and other signals are sent only once; the R_RDYs are transmitted with the rule of at least two fill words prior-to and following-by; the logic of LPSM state transition is verified; the Alternative BB credit flow control logic is tested; and the FCP transaction protocol is evidenced.

These traces are also analyzed to achieve some necessary configuration information that is used in later system level validation, such as the frame buffer count

```
time:4.881436,portID:1,state:3,cnt:70  ::ARB(01,01)::=>::OPN(00,01)::
time:4.881474,portID:1,state:3,cnt:1   ::OPN(00,01)::=>::ARB(F0,F0)::
time:4.881664,portID:1,state:3,cnt:5   ::ARB(F0,F0)::=>::R_RDY:::::::
time:4.881702,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::SOFi3:DATA::
time:4.901462,portID:1,state:3,cnt:520 ::SOFi3:DATA::=>::EOFn3:::::::
time:4.901500,portID:1,state:3,cnt:1   ::EOFn3:::::::=>::IDLE::::::::
time:4.901576,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.901614,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::IDLE::::::::
time:4.901690,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.901728,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::SOFi3:DATA::
time:4.921488,portID:1,state:3,cnt:520 ::SOFi3:DATA::=>::EOFn3:::::::
time:4.921526,portID:1,state:3,cnt:1   ::EOFn3:::::::=>::IDLE::::::::
time:4.921602,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.921640,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::IDLE::::::::
time:4.921716,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.921754,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::SOFi3:DATA::
time:4.941514,portID:1,state:3,cnt:520 ::SOFi3:DATA::=>::EOFn3:::::::
time:4.941552,portID:1,state:3,cnt:1   ::EOFn3:::::::=>::IDLE::::::::
time:4.941628,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.941666,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::IDLE::::::::
time:4.941742,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.941780,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::SOFi3:CMND::
time:4.942350,portID:1,state:3,cnt:15  ::SOFi3:CMND::=>::EOFn3:::::::
time:4.942388,portID:1,state:3,cnt:1   ::EOFn3:::::::=>::IDLE::::::::
time:4.942464,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.942502,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::IDLE::::::::
time:4.942578,portID:1,state:3,cnt:2   ::IDLE::::::::=>::R_RDY:::::::
time:4.942616,portID:1,state:3,cnt:1   ::R_RDY:::::::=>::SOFi3:CMND::
```

Figure 5.3 Simulative L_Port Event Trace

of the initiator HBA or the target FC hard disk, the command execution overhead of the initiator or the host, the status generation time. These parameters in many cases may become some performance bottleneck, and may dominate the I/O performance. For example, the initiator HBA command execution time may determine the maximum I/O throughput for small access.

5.2 Trends Confirmation

In order to confirm and verify the simulation model's overall correctness, a series of simulations have been conducted, with some basic performance factors changing while others keep constant, so that the overall trend of the FC-AL performance can be achieved. High accuracy of most hardware overheads in these simulations is not required, since this thesis is focusing on the changing trends.

The close-system I/O workload is used in these and succeeding simulations. As shown in Figure 4.19, the workload generator initially generates number "n" of I/Os that will simultaneously enter the I/O subsystem under test. The execution order of

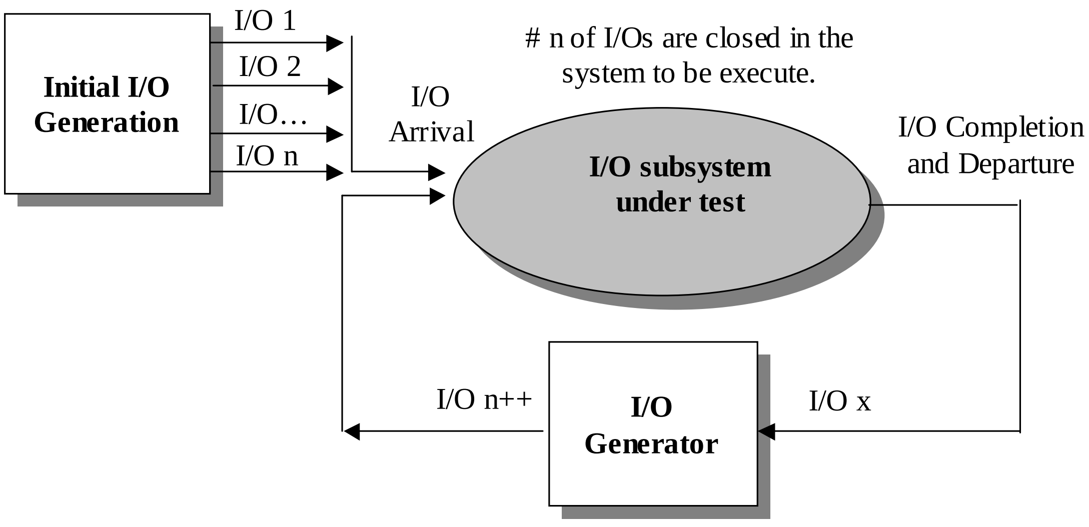


Figure 5.4 Close-system I/O Workload

these I/Os is up to the I/O subsystem, either concurrently or sequentially or in-between. When any one I/O completes, it departs from the I/O subsystem and triggers the I/O generator to generate a new I/O that will enter the subsystem again quickly. Subsequently, there are always number “n” of I/Os existing in the I/O subsystem. With greater number “n”, the subsystem will be more deeply explored for parallelism. This is probably the reason why the “n” is often instead called “queue-depth” in I/O performance measurements.

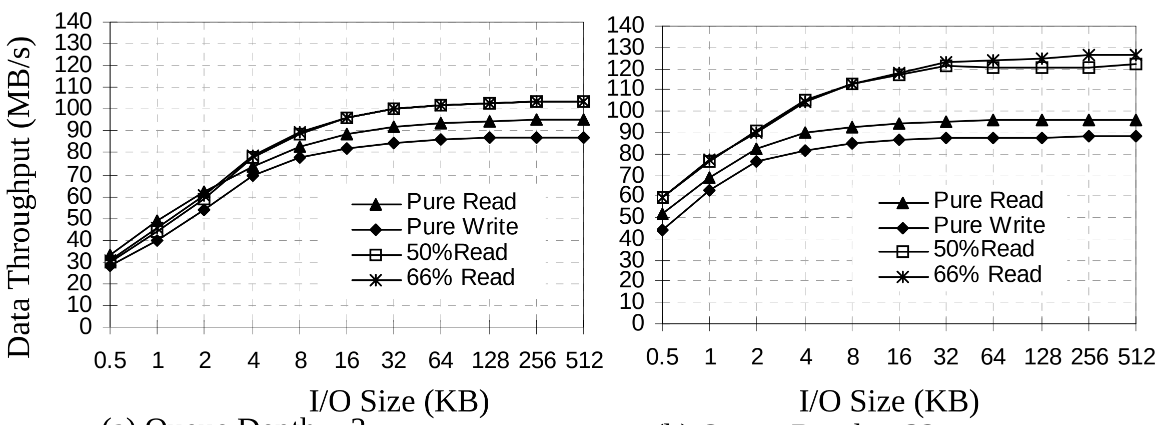
The remainder of this section presents the simulation results for a number of cases. Firstly, the simulation result of two nodes configuration (one initiator and one target) with short physical distance are presented. The factors of read or write, queue-depth and the I/O sizes are considered. Secondly, the effect of number of nodes in the loop is discussed. With more nodes attached, the overall round trip time becomes longer and the performance shall degrade in some level. Thirdly, the effect of large physical distance is investigated.

5.2.1 Performance of One-to-one Configuration

This subsection evaluates the simulation model for handling general performance simulation on a one-to-one simple configuration (one initiator to one target and no other passive port). The HBA overheads for command execution are set to small to simplify the analysis. The FC link rate is set to one gigabit per second and the alternative buffer to buffer credit flow control with one LoginBBCredit is supported. The full duplex function is turned on. The HDD access time is configured to small so that the data can be transferred quickly, while the maximum concurrent I/O requests supported by the HDD is set to four. The closed-system I/O workload is used with various queue-depths.

When queue depth is one, the simulated throughput grows and approaches to the maximum bandwidth of 100 MB/s as the I/O size increases, since the weight of the system overhead per I/O request over the data transfer time significantly decreases when I/O size changes from small to large. The system or the FC-AL protocol overhead causes some idle periods before and after the frame transfer and the fraction of idle periods to the frame transferring determines a lower than 100% bandwidth utilization. For example, when I/O size is 2 KB, one single data frame is used for data transfer and every I/O, subsequently every data frame, will impose certain idle periods. When I/O size is 1 MB, hundreds of frames (512) may be transferred in a continuous chunk. The system overhead imposes only certain idle before and after this chunk of frames transfer. The overhead are shared by 512 frames. Therefore, the bandwidth utilization increases and the throughputs surges up, approaching to the maximum bandwidth. The maximum achievable throughput however are always smaller than the nominate 100 MB/s, regardless read, write or the mixtures I/O types, since there is only one I/O in processing in any time for one queue depth and there is no chance to transfer both read and write data in the two directions. In other word, the full duplex potential is not explored. It is also noted that the write throughput is generally lower than the read in the simulation, because the write requires the additional phase of FCP_XFER_RDY. The mixture of read and write (50% read and 66% read in the simulations) has the combination effect of the read and writes, which is worse than the read but better than the write.

When queue-depth increases, multiple I/Os are executed in parallel and so the bandwidth utilization increases. The throughput improvement is expected since the system overhead is shared among these I/Os. Meanwhile, for the I/O workload with combination of read and write, concurrent I/Os can achieve higher than 100 MB/s



(a) Queue Depth = 2 (b) Queue Depth = 32

Figure 5.5 FC-AL Throughputs with Two Nodes Configuration

throughput with full duplex when I/O is large. Figure 5.5 shows the simulation results of the data throughput when the queue depth is 2 and 32, for four types of I/O workload with I/O size changing from 0.5 KB to 512 KB. It can be seen that the small read and write mixture I/O does not achieve much throughput gain when queue depth is two. The possible reason is that the probability of the small read I/O overlapping with the small write I/O is small and the full duplexing is hardly possible. However, when the queue depth changes to 32, the overlapping probability significantly increases and thus mixture type of I/O can achieve significant improvement, as seen in (b) of Figure 5.5. Meanwhile, when I/O size is large, it is always possible to find the overlapping period, since the devices can support multiple request concurrently. The mixture I/O thus achieves better performance than pure read and pure write for queue depth 2 (Figure 5.5 (a)), and even better for queue depth 32 (Figure 5.5 (b)) It is also noted that the 66% read I/O achieves slightly higher throughput than 50% read I/O when queue depth is 32. This is probably due to the slightly longer write execution time and more read I/Os are required to get better overlapping.

The effect of queue depth can be seen clearly in Figure 5.6 for all four types of I/O workload. For pure read (a) and pure write (b), with deeper queue-depth up to four,

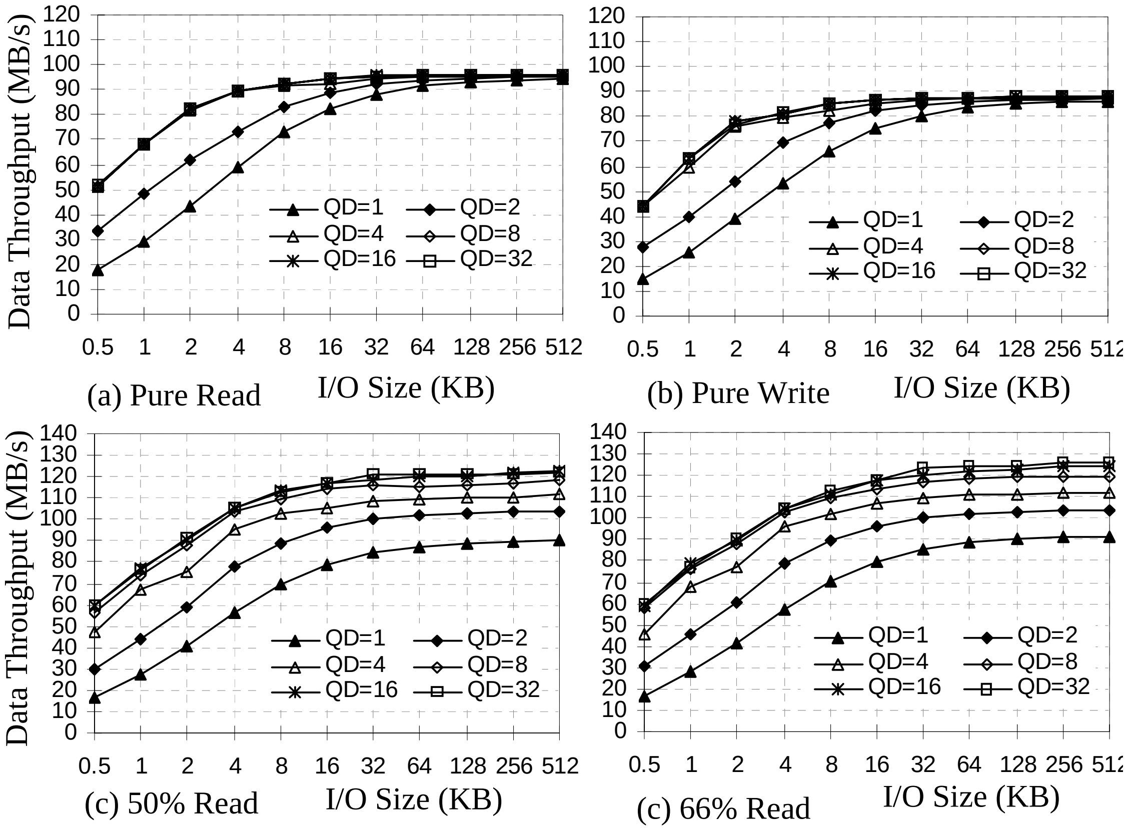


Figure 5.6 Queue Depth Effect with Two Nodes Configuration

the small I/Os achieve significant throughput improvements, since the system overhead dominates the small I/O performance, while the large I/Os hardly gain better performance because the link utilization is already high. In contrast, the 50% read (c) and 66% read (d) achieve significant performance improvement for both small and large I/Os because of the concurrency and the full duplexing. However, the throughputs do not further increase after queue depth four, since the HDD used supports only four maximum concurrent I/Os.

To sum up, the results obtained from the simulation model for this example follow the expectation in general, for various workload types with different queue depths, from small I/O to large I/O. The proper modeling on the full duplex feature is proved and the I/O execution process is tested.

5.2.2 Effect of Number of Node

According to the standard, each port in a FC-AL loop is allowed to delay signal retransmission for six transmission words. This delay is referred as “per-port delay”. With 1 gigabit per second link rate, the per-port delay may up to 226 nanoseconds (6*40/1.0625). As node number increases, the overall delay increases. The simulation model for this delay effect is verified in this sub section.

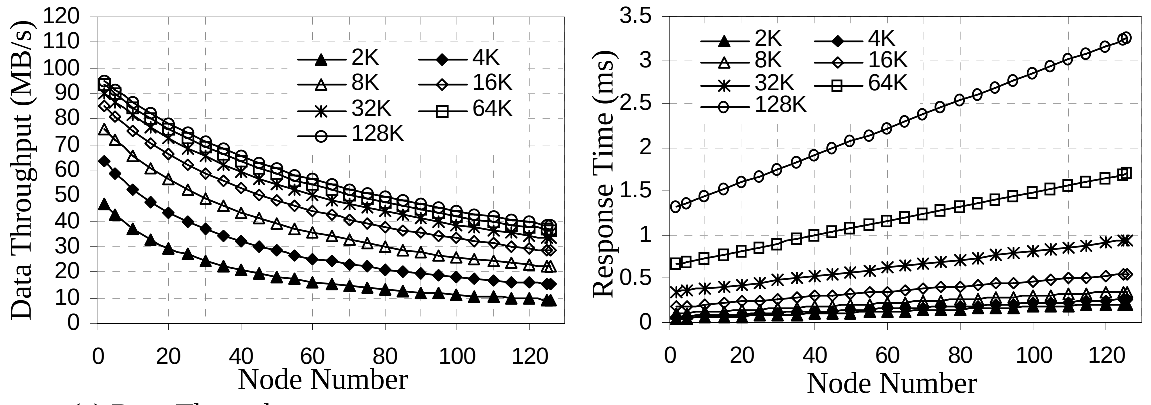
It is assumed that there is only one initiator and one target in the loop, while the remaining nodes participate only in signal retransmission. The other I/O overheads are set small for better focus. The closed-system I/O workload is used with queue-depth equals to one. The total node number in the loop changes from 2 to 126. The response time growing trend as node number increase is investigated first, followed by a discussion on the performance improvement by increasing incoming frame buffer size and implementing non-zero LoginBBCredit.

As described in earlier chapter, the I/O requests are executed in multiple phases according to the FCAL protocol. Table 5.1 presents an analysis on read I/O phases and corresponding loop latencies required for the case of one incoming frame buffer and

Table 5.1 Read Transaction Loop Latency   IncomingBuffer=1, LoginBBCredit=0

I/O Phases	2k	4k	8k	16k	32k	64k	128k
Arb1	1	1	1	1	1	1	1
CMD Credit	1	1	1	1	1	1	1
CMD Deliver	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Arb2	1	1	1	1	1	1	1
Data Credit	1	2	4	8	16	32	64
RSP Credit	1	1	1	1	1	1	1
RSP deliver	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Total	**6**	**7**	**9**	**13**	**21**	**37**	**69**
$\Delta T/\Delta N$ (ns)	**1355**	**1581**	**2033**	**2936**	**4744**	**8358**	**15586**
Simulation	**1355.28**	**1581.16**	**2032.91**	**2936.48**	**4743.54**	**8357.61**	**15585.97**

zero LoginBBCredit. To start a read I/O, the initiator first arbitrates and wins the loop access (Arb1), requiring one round trip signal transferring (from sending its own ARB to receiving the same ARB). After winning the loop, the initiator sends OPN but can not send the FCP_CMND frame immediately because of zero LoginBBCredit. It takes half round of loop traveling for the OPN to reach the target (assuming that the target is located in the middle of the loop trip but it is not essentially necessary), and the target send a credit back to the initiator after receiving OPN. Therefore, the initiator spends one whole round trip to gain the credit to send the I/O command. The command frame takes half round trip to reach the target. After the target receive the command, it spends some time to execute the command but the time is not related to the loop latency and is not affected by the node number. It is therefore excluded from the analysis. This is also applicable to other overheads. When the data is ready, the target has to arbitrate and win the loop (Arb2) taking one whole loop latency. It is noted that there are no other arbitrators in the example, and that the target can hold the loop for all subsequent frame transferring. In other word, no additional arbitration is required fro the data transferring. Moreover, with assumption that the status frame is generated quickly enough, the status frame is appended to the data frame sequence. However, there are crediting loop latencies for each frame transferring. Due to zero LoginBBCredit, the target takes one loop-latency to obtain the credit for sending the first data frame. Since the incoming buffer is one, there is only one credit sent by the initiator. The target therefore has to wait for a new credit that is only received after the initiator receives the first frame (half loop latency since the target complete sending the first frame) and sends back the credit in another half loop latency, before it can send another frame. Therefore, when I/O size is 128 KB, there are total 64 round-trip loop latencies for the crediting (assuming frame size is 2KB). Last but not least, the status frame (RSP)



(a) Data Throughput (b) Response Time

Figure 5.7 Effect of Number of Nodes

requires additional round to obtain the credit and another half round to reach the initiator to mark the completion of the I/O. The total rounds of loop latency required for various I/O sizes therefore can be obtained as in the table. Since each increment of node number cost additional 226 nanoseconds to the loop latency, the total increase of the I/O response time can be determined.

The corresponding simulations are conducted to confirm the simulation model with the above analysis. Figure 5.7 shows the simulation result of data throughput and response time when node number increase, for read I/O with different size from 2KB to 128KB. Since the queue depth is one, the data throughput equals to the I/O size

Equation 5.1 Throughput declines slowly as node number increases:

$$\mathrm{ResponseTime}(\mathrm{nodeNum}) = \mathrm{BaseIOtime} + \mathrm{nodeNum} \times \mathrm{PerNodeOverHead}$$

$$\mathrm{ThroughPut}(\mathrm{nodeNum}) = \frac{\mathrm{RequestSize}}{\mathrm{ResponseTime}(\mathrm{nodeNum})}$$

$$= \frac{\mathrm{RequestSize}}{\mathrm{BaseIOtime} + \mathrm{nodeNum} \times \mathrm{PerNodeOverHead}}$$

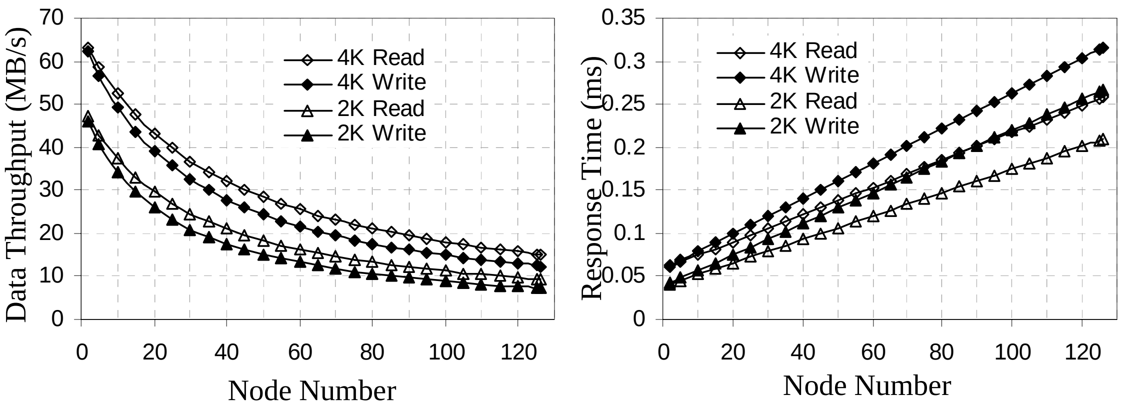
$$\frac{\Delta[\mathrm{ThroughPut}]}{\Delta[\mathrm{nodeNum}]} = -\frac{\mathrm{PerNodeOverHead} \times \mathrm{RequestSize}}{(\mathrm{BaseIOtime} + \mathrm{nodeNum} \times \mathrm{PerNodeOverHead})^2}$$

divided by the I/O response time. The data throughput therefore declines when node number increase, since the response time linearly grows. Equation 5.1 gives the data throughput degradation function as the node number increases. It can be seen that the throughput declines more slowly as the node number become larger. It is also note that the response time increase faster when I/O size is large. The response times for each I/O size and corresponding node number are measured and the growing speeds per node number increment are calculated. As shown in last row of Table 5.1, the simulation accurately agrees with the analysis results.

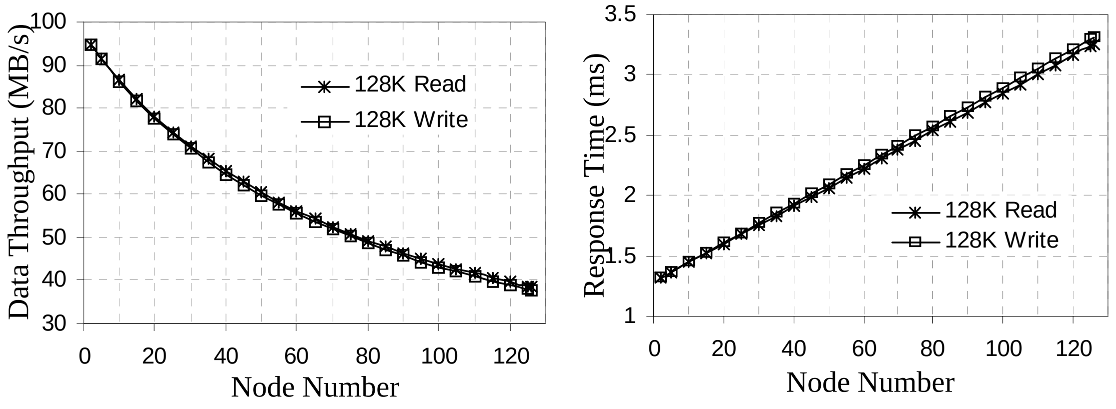
For the write I/O, the I/O transaction phases differ to the read. The target arbitrates the loop and sends the XFER_RDY frame to the initiator after receiving I/O command. The initiator then arbitrates the loop and transfers data with crediting overhead for each data frame. The last data frame takes half loop latency to reach the target. The target then arbitrates the loop to send the status and the I/O is complete when the RSP is received by the initiator with half loop latency. Table 5.2 summaries the loop latency analysis for the write with same assumption of one incoming buffer and zero

Table 5.2 Write Transaction Loop Latency IncomingBuffer=1, LoginBBCredit=0

I/O Phases	2k	4k	8k	16k	32k	64k	128k
Arb1	1	1	1	1	1	1	1
CMD Credit	1	1	1	1	1	1	1
CMD Deliver	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Arb2	1	1	1	1	1	1	1
XFER Credit	1	1	1	1	1	1	1
XFER Deliver	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Arb3	1	1	1	1	1	1	1
DATA Credit	1	2	4	8	16	32	64
Data Deliver	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Arb4	0	0	0	0	0	0	0
RSP Credit	1	1	1	1	1	1	1
RSP deliver	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Total	**8**	**9**	**11**	**15**	**23**	**39**	**71**
$\Delta T/\Delta N$ (ns)	**1807**	**2033**	**2485**	**3388**	**5195**	**8809**	**16038**
Simulation	**1807.06**	**2032.94**	**2484.71**	**3388.24**	**5195.29**	**8809.41**	**16037.65**



(a) Small I/O Performance comparison for Read and Write



(b) Large I/O Performance comparison for Read and Write

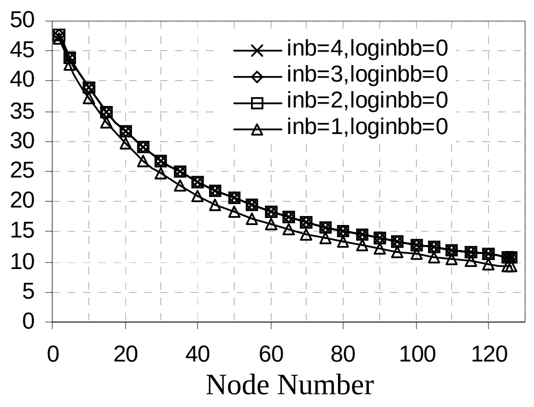
Figure 5.8 Small I/O Read/Write Comparisons for Node Number Effect

LoginBBCredit. A set of simulations are conducted and the response time growing speed per node number increment are presented in the last row of the table, which are accurately matching the analysis result.

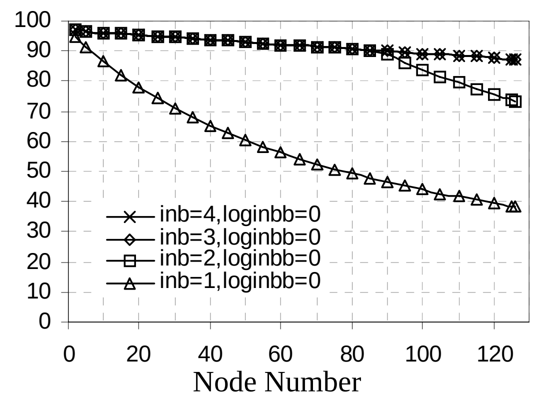
Apparently, the node number increase has greater effect on the write than that on the read. Figure 5.8 shows the simulation result of data throughput and I/O response time for the comparison of the read and the write when I/O size is small in (a) and large in (b), with node number increasing from 2 to 126. As shown, the small write performance decline much noticeably from read comparing to the large I/O. This is because the weigh of the loop latency overhead to the data transferring is much smaller for large I/O. For 128K request, the total data transfer time is 1.25 milliseconds (128KB divided by 100MB/s), while the total loop latency difference between the read and the write is about 0.5 microseconds only. Therefore, the large write is only marginally worse than the large read for the performance degradation caused by node number increment.

From the above loop latency analysis, it is clear that the performance can be improved by eliminating the loop latencies required for crediting. Two options are considered. One way is to equip more incoming frame buffers to the port, so that the frame can be sent continuously. The other supplemental way is to make the loginBBCredit non-zero, so that the port can transfer frame immediately after winning the arbitration.

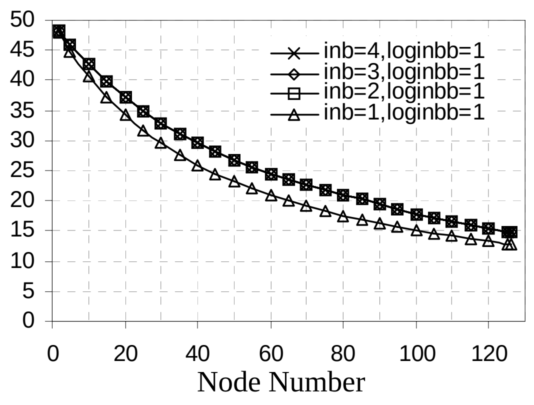
Figure 5.9 presents the simulation results of data throughput for small (2KB read) and large (12KB read) I/Os, for the node number effect when various incoming frame buffer number and non-zero loginBBCredit are applied. As shown in (a), where I/O is small and loginBBCredit remains zero, the performance is noticeably improved when incoming buffer changes from one to two. No further improvment is achieved when incoming buffer increase further to three or four, since there is at maximum only two frames (one data frame and one status frame for 2KB I/O) to send and only one crediting loop latency is saved for the second frame. By contrast, when the I/O is large, two incoming frame buffers improve throughput tremendously as shown in figure (b). At 60 nodes, the throughput degradation from 2 nodes for two incoming frame buffer is only about one tenth of that for one incoming frame buffer. The reason is that the two incoming frame buffer make it possible to transfer all data frames continuously. In such case, the first frame is sent after the first credit is received and by the time of transferring completion, the second credit shall already be received and the second frame can be sent. On the other hand, the first frame takes half loop latency to reach destination port and a new credit is sent back in half loop latency. The new credit



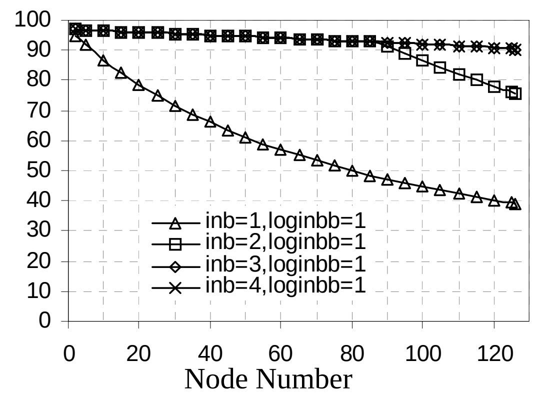
(a) 2K Read with zero LoginBBCredit



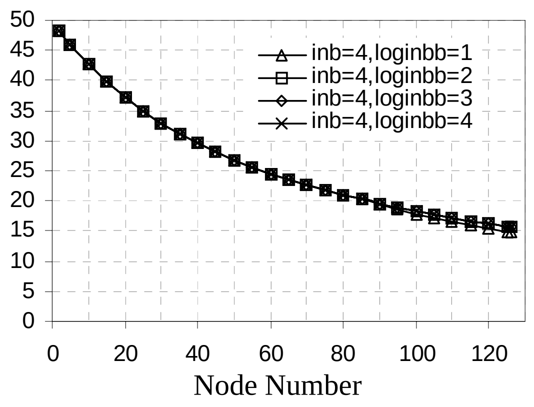
(b) 128K Read with zero LoginBBCredit



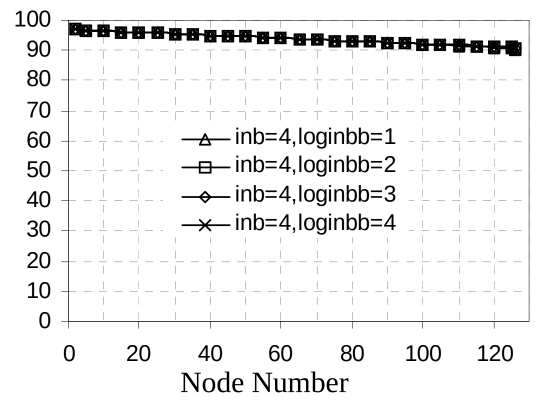
(c) 2K Read with one LoginBBCredit



(d) 128K Read with one LoginBBCredit



(e) 2K Read with more LoginBBCredit



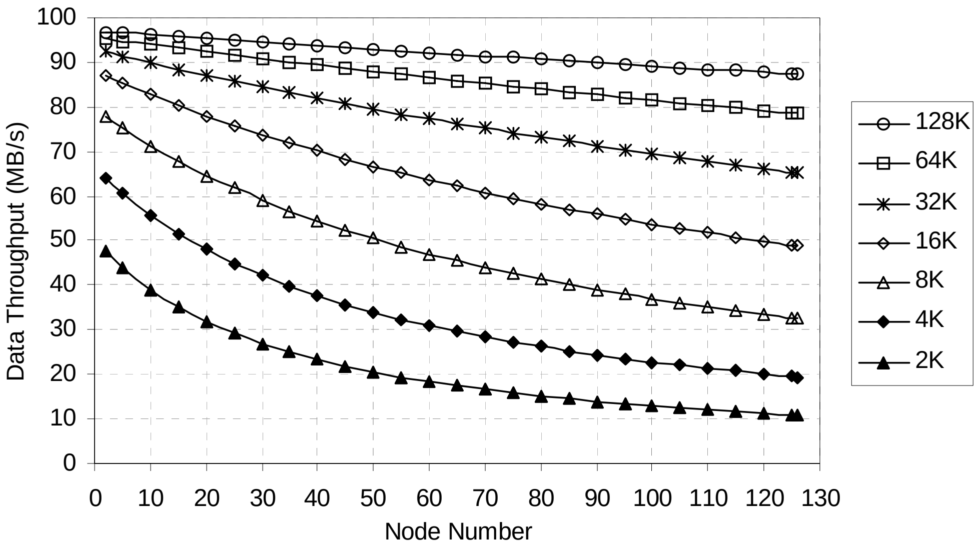
(f) 128K Read with more LoginBBCredit

Figure 5.9 Sufficient Buffering to Improve Performance
(All vertical axis measure the data throughput in MB/s)

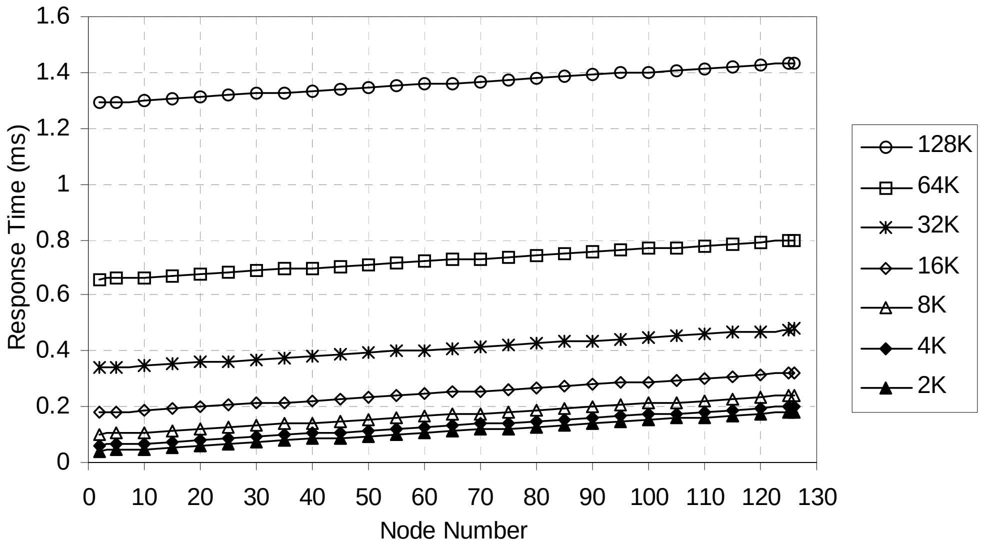
therefore will reach the frame sender in one loop latency and if and only if it arrives earlier than the completion of the second frame transferring, the third frame can be sent continuously. That is the reason why the data throughput falls at about 90 nodes, where the loop latency is about 20.34 microseconds (90*224) exceeding 2KB frame transferring time (19.58 microseconds). When incoming buffer is three, the new credit will always arrive earlier than two frames transferring for all possible node number (2-127), and thus data frames can be sent continuously. Four incoming buffer serves no better than three. Comparing (c) to (a) and (d) to (b), if non-zero LoginBBCredit is implemented, the performance is further improved for both small and large I/Os, since the port can send frame without the initial crediting latency. If incoming frame buffer is sufficient, one LoginBBCredit can eliminate the initial crediting latency when node number is smaller than 90. After that, a small waiting time is required to receive the new credit after the first frame is sent and this time has only minor effect on the overall throughput and even hardly noticeable when I/O is large, as shown in (e) and (f).

From above analysis, it is save to conclude that the configuration with three incoming frame buffers and one LoginBBCredit can sufficiently support large loop up to the maximum 127 nodes. However, the loop latency still increases as more nodes are added and cause performance degradation since the protocol require some minimum round-trip communications. Figure 5.10 plots the throughput and I/O response time for the case of three incoming frame buffers and one LoginBBCredit, as a comparison to Figure 5.7. It can be seen that the degradation are greatly reduced and all I/O sizes has same response time growing speed in Figure 5.10, which means no additional overhead are required for large I/O.

To sum up, the simulation model are accurately developed in term of port delay, the I/O transaction protocol is properly modeled and the alternative buffer to buffer flow control is correctly implemented.



(a) Data Throughput Effect of Number of Nodes



(b) Response Time Effect of Number of Nodes

Figure 5.10 Effect of Number of Node with Optimal Buffering

5.2.3 Effect of Physical Distance

The signal propagation delay for long physical distance may become significant enough to greatly degrade the I/O performance. Figure 5.11 shows the simulation results obtained from the simulation model for the effect of long distance transfer.

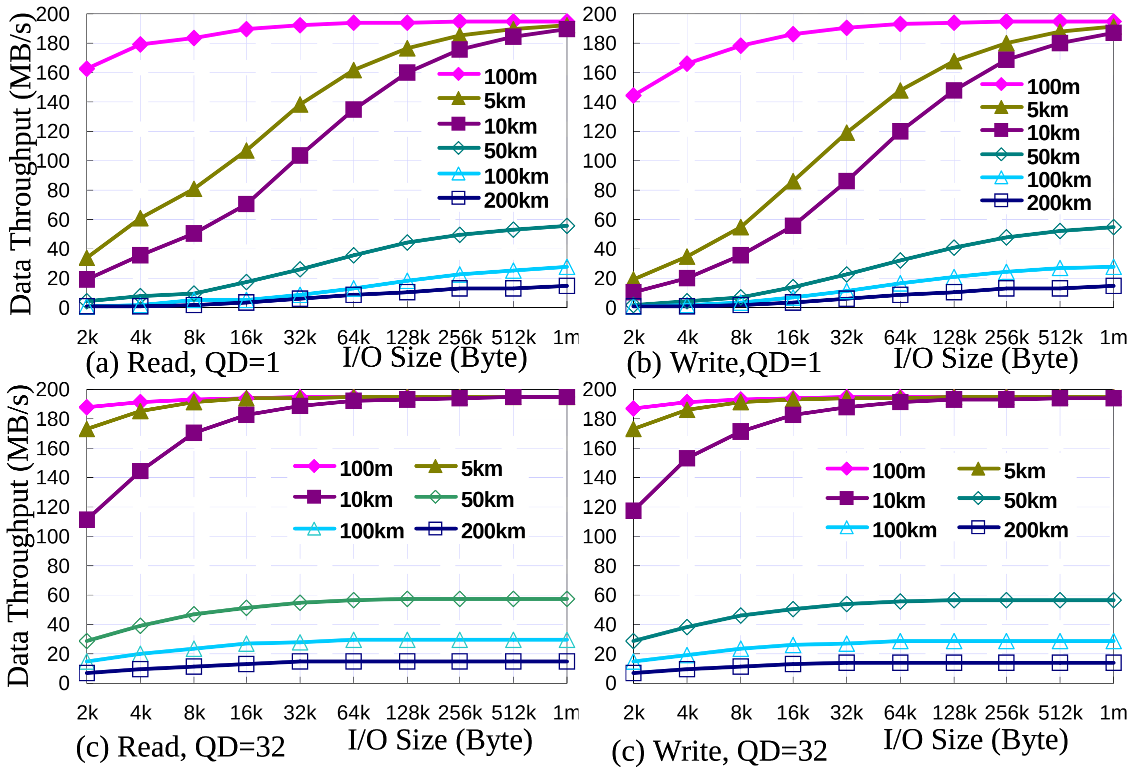


Figure 5.11 Effect of Physical Distance

As can be seen, the throughput degrades tremendously when the physical distance extends from 100 meters to 100 kilometers. This is probably due to two problems. Firstly, the long distance causes a large round trip time for an L_Port to send and received signals for loop communication. For optical cables, one single round trip time reaches about 0.5 milliseconds for 100 kilometers loop distance. Since each I/O requires minimum three loop round trips (arbitrating, send OPN and receive first R_RDY, send and receive CLS), the minimum time require is 1.5 milliseconds, which results in less than 700 IOPS and 1.4 MB/s for 2 KB I/O when queue depth is one. Secondly, the performance degradation for large I/O may be probably also due to insufficient frame buffer for the BB credit flow control. With 2 gigabit per second FC link rate, one 2K FC frame is transmitted in 10 microseconds. When I/O size is 2 M, the total transmission time is about 10 milliseconds (1024 frames * 10 microseconds per frames). With sufficient incoming frame buffer, the bandwidth utilization can be calculated as 10/(10+1.5) and results in over 85%. The achievable throughput therefore shall still be about 170 MB/s. This does not happen because the port being simulated does not have enough incoming frame buffer. The frame sending port has to wait for R_RDY to come when the credits are used up.

With the same 100 kilometers distance example, it will take 10 microseconds to transmit one 2 KB frame from the first bit to the last bit, and 0.25 milliseconds (assuming the destination port is in the middle of the round trip) for the last bit of the frame to reach the destination. Assuming that the destination port takes little time to clear the buffer, it sends back an R_RDY immediately when it receives the frame. The R_RDY then reaches the source port in 0.25 milliseconds. The total time from the beginning of frame sending to the time when the source port receives the corresponding R_RDY is thus about 0.51 milliseconds. During this period, the source port may receive several other R_RDYs, together with the remaining credits after the one used for sending the frame. If and only if all these add-up is greater than 51 (0.51/0.01), the source port is possibly continuously transmitting frames. Hence, total 52 incoming frame buffers are required for fully utilizing the transmission bandwidth. However, the port being simulated has only four incoming frame buffers in this example, which is far less than the required. After every four transmission, the port has to wait for 0.47 milliseconds (0.51-0.04). The bandwidth utilization can thus go low to about 8.5%, which results in about 18 MB/s data throughput. The simulation results shown in Figure 5.10 agree with the above analysis. Therefore, the degree of the model validity is further improved.

5.3 Actual Testing and Simulation Comparison

The simulation model so far has been verified in transmission levels by using the FC analyzer and confirmed on the general performance trends, by comparing simulation result with general analysis. In order to further validate the model, an actual system experiment is conducted in this section to compare the actual measurement with the simulation result.

5.3.1 Experimental Environment

The experiments are conducted on a FC-AL configuration with one window's initiator and one FC RAM disk target. Two Qlogic 2300 FC HBAs are used to support the 2 Gigabit FC, one for the initiator and the other for the target. Microsoft Window

Table 5.3 Experimental System Configuration

	Initiator	Target
Hardware	CPU: AMD AthonMP 1600+ FC HBA: Qlogic 2300 RAM: 2x256MB DDR SDRAM Mainboard: 64 bit PCI Tyan Tiger MP2466N	CPU: Intel PIII 1GHz FC HBA: Qlogic 2300 RAM: 4 x 1GB Kingston ECC Reg. PC133 Mainboard: 64bit PCI, Supermicro 370
Software	OS: Windows XP Professional SP1 Driver: Qlogic Driver Version 8.1.5.12 Tool: Intel IOMeter Version2003.02.15	OS: RedHat 8.0 Kernel: 2.4.18 Driver: In-house 2300 target driver Ver 1.0, In-house Linux FC RAM Disk Ver 2.0

XP is installed on the initiator system together with the HBA initiator device driver to drive the HBA. The target HBA is installed on a Linux system that is configured to be a virtual HDD using DSI’s FC RAM Disk software. The software maps all storage I/O to the memory rather than to an actual magnetic disk. Since this thesis is focusing more on the FC-AL connection, using RAM disk as a target helps to isolate problems from modeling of actual hard disk drive. Table 5.3 gives the detail hardware and software configurations in the experiments.

The I/O Meter [40], widely used in the industry, is installed on the initiator system to conduct the experimental test and measurement. After the system is boot-up, the virtual hard disk appears in the window system as a “physical raw disk” and is ready to be tested by the I/O Meter. The “physical raw disk” test bypasses the file system overhead for better focusing. The I/O Meter typically supports the closed-system workload, with specifiable parameters of queue-depth, I/O size, fraction of read and others. The software also allows specifying a “warm-up” time to eliminate the transient period of the test, as well as the testing duration in which the I/O statistics are collected. The simulation sets 10 minutes for the warm-up and another 30 minutes for the data collection in each test. Since the RAM DISK responses very fast and it will complete 1,800 I/Os for 2 MB request in 30 minutes and many times more for smaller I/O size. This will give sufficient sample space for the confidence of the measurement.

The simulation measures the system’s data throughput (MB/s) and I/O throughput (IOPS) with I/O size changing logarithmically from 2 KB to 512 KB, queue-depth linearly from 1 to 12 for both read and write. The obtained measurements are recorded and compared with the corresponding simulation results.

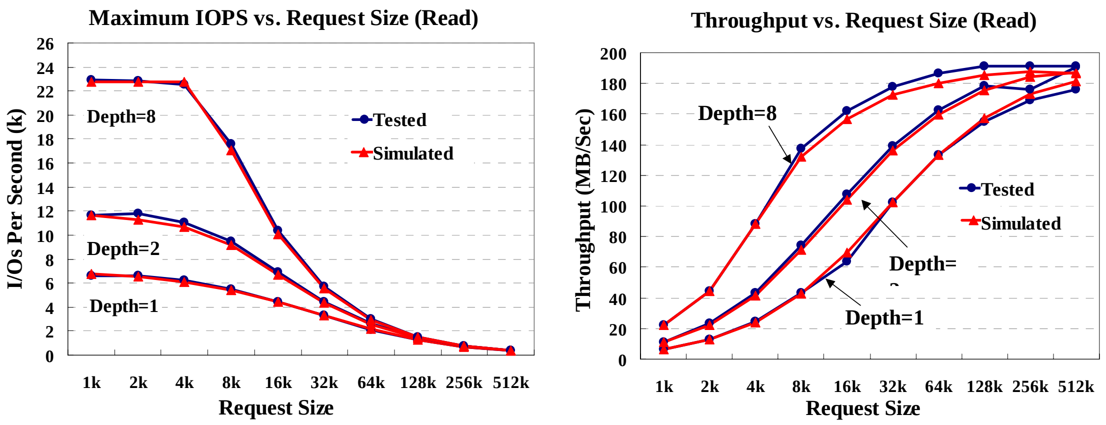


Figure 5.12 Read Experiments Comparisons

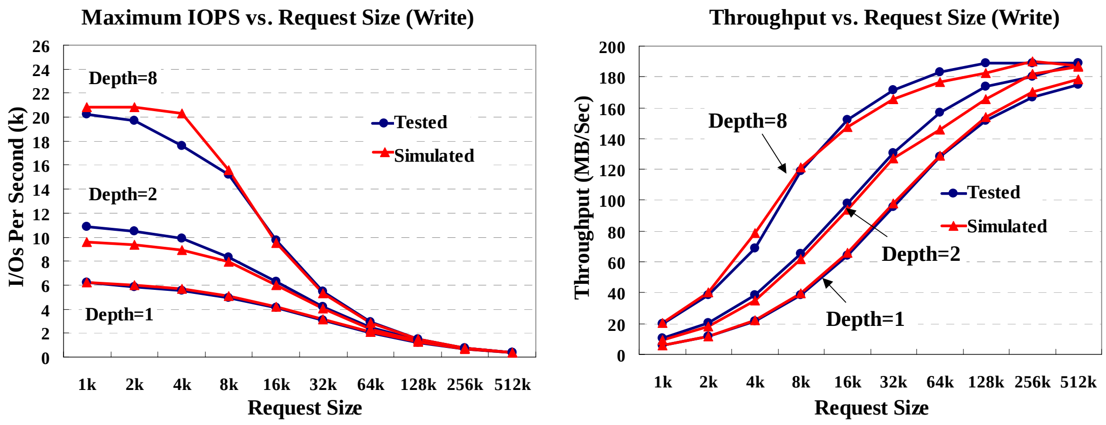


Figure 5.13 Write Experiments Comparisons

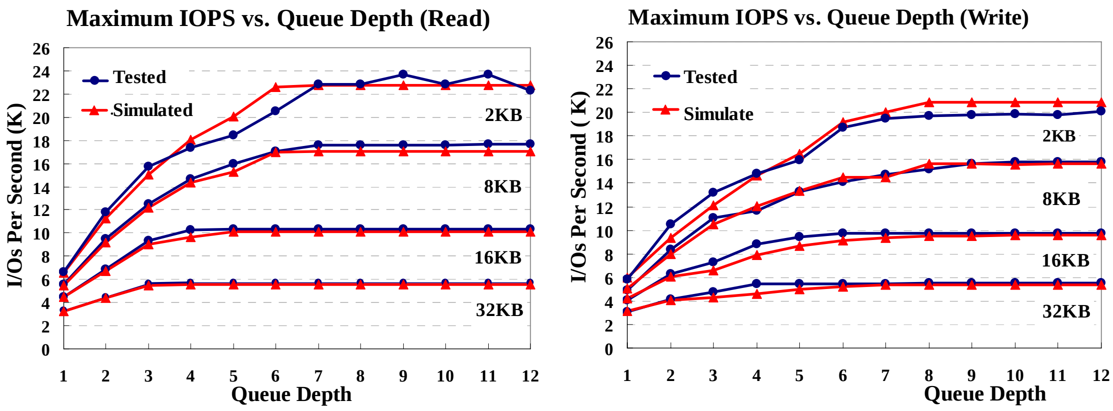


Figure 5.14 Queue Depth Effect Experiment Comparisons

5.3.2 Result Comparisons

Figure 5.12-5.14 plot simulated and tested results together for comparison. The tested results are drawn in round-dot line and the simulated data are in triangle-mark line. It can be seen that the simulation results match well with the actual measurements for all cases. Further numerical comparison shows that the mismatching are mostly less than 10%. For read operation simulation, the mismatching is less than 3%. Thus, the correctness and accuracy of the simulation model is demonstrated.

5.4 Summary

The simulation model has been calibrated and validated from three different prospects. From the FC signal transmission level angle of view, the model has been verified by checking signal transmission events against the actual FC analyzer's traces. From the general I/O performance trends point of view, it has been proven that the simulation model outcome agrees well with the expectation. Lastly, and also more importantly, the simulation model has demonstrated its accuracy by comparing its result with actual experimental measurement.

Chapter 6

Command-First Algorithm Performance

The Command-First Algorithm (CMDF) is evaluated by simulation in this chapter. The overall method for the performance comparison between normal schedule and CMDF is first presented. The configuration parameters and overhead constants are stated. The performance improvement of CMDF on a sixteen HDDs storage system is presented and the effects of CMDF are further investigated when number of HDDs varies, the HDD access time changes and workload increases.

6.1 Overall Method

The effect of CMDF is evaluated by comparing I/O performances of an identical simulative storage system with or without the algorithm. As shown in Figure 6.1, a simulative storage system under test can be determined with given configuration system parameters and FC-AL schedule methods. The simulative storage system use FC-AL as

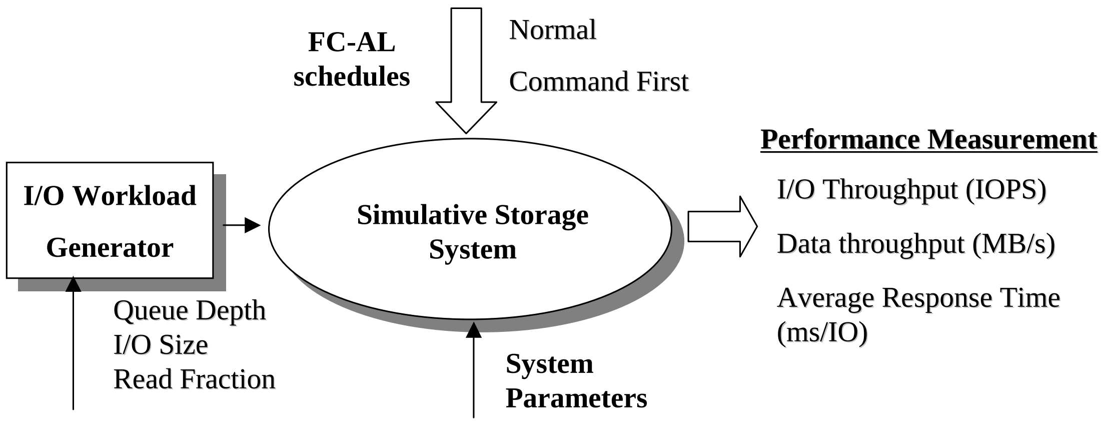


Figure 6.1 Performance Evaluation Method for CMDF

the backend interconnection. The arbitration schedule used on the storage controller of the storage system can be configured as either normal or CMDF. Once the system under test is determined, the I/O workload generator injects I/O requests based on the given parameters such as queue depth, I/O size and read fraction. The simulation model processes these I/Os and reports their completion.

The performance matrixes such as I/O transaction throughput (IOPS), data transfer throughput (MB/s) and average I/O response time (ms) are monitored during each simulation. The total number of I/O requests that have been completed is recorded. By dividing the total number of completed requests to the simulation duration, the IOPS can be computed. During the simulation, the payload of every data frame transferred from source to destination port is counted for the total data transferred. The total data transferred is divided by the simulation time to achieve the data throughput. Each I/O request is time-stamped for arrival and completion. The I/O response time is determined by subtracting the completion time to the arrival time. The sum of I/O response time of all I/O requests is computed by adding the I/O response time to a history variable that holds previous sum. The average response time can therefore be achieved by dividing this sum to the total number of I/O requests.

6.2 System Configuration

The simulative storage system consists of one imitator HBA (storage controller) and a number of HDDs connected by a shared FC-AL loop, as shown in Figure 6.2. The storage controller is equipped with the specified arbitration schedules: the normal

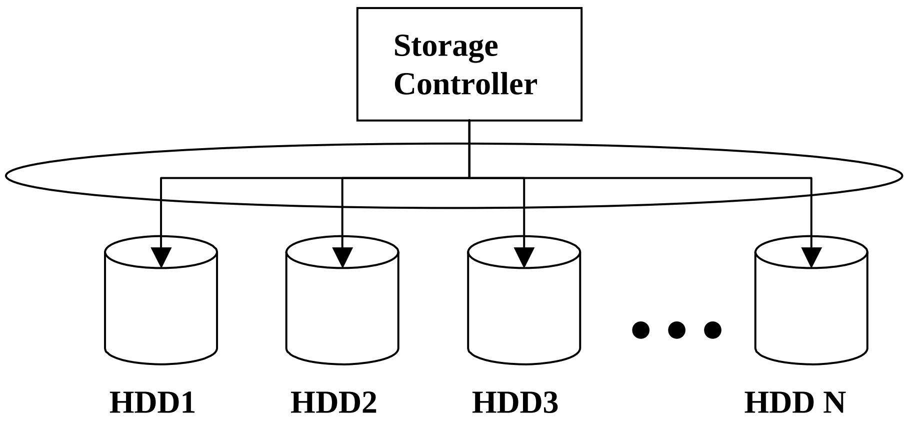


Figure 6.2 System Configurations

schedule or the CMDF, which can be determined by configuration inputs. By contrast, the HDDs attached use the normal arbitration schedule only. Other overhead constants and control parameters used in the model are specified as follows.

6.2.1 System Overhead Constant

It is initially assumed that the HDDs used are state-of-art fast disks with two milliseconds average access time and 50 MB/s data transfer rate. A single disk can therefore provide up to 500 I/Os per second for small access and 50 MB/s data throughput for large I/Os. It is noted that the seeking time and the rotational latency of the disk access are not distinguished in the model. They are included in the total delay from receiving command to beginning of data transferring. The effects of different HDD access speeds are further investigated.

On the other hand, for the purpose of problem isolation, it is further assumed that other overheads of the storage controller are small and negligible. This may be not true for a real storage system, but it is worthwhile to artificially configure these overheads to

Table 6.1 System Overhead Constant

HDD Access Time	2 milliseconds per I/O
HDD Transfer Bandwidth	50 Mbytes per second
FC Link Rate	2 Gigabits per second
Storage Controller Overhead (Close-system workload “think time”)	Small and negligible

small so that some performance bound problem of FC-AL loop can be investigated under saturated conditions. The system overhead constants are summarized in Table 6.1.

Based on the analysis of real FC analyzer traces obtained during the model validation, the simulation sets HBA’s overhead and control constant as shown in Table 6.2 and Table 6.3 for the initiator and the target respectively. It is noted that the HBA has command execution overhead of 43.9 microseconds, which theoretically results in 22K IOPS I/O processing capacity. Following most industry implementation, the login guaranteed buffer credit is set to zero. The DMA transfer bandwidth between the HBA

Table 6.2 Initiator HBA Overhead and Control Constant

Incoming Frame Buffer	3
Login Guaranteed Buffer	0
Maximum Frame Size	2048 Bytes
Full Duplex	No
Command Execution Overhead	43.9 microsecond
FCP_XFER_RDY Handling Overhead	11.8 microsecond
DMA Bandwidth	1064 Mbytes per second
DMA Scheduler Overhead	15.5 microsecond
DMA Round Robin	No
Incoming Frame Processing Overhead	400 nanosecond
Single Frame Transfer Overhead	20.2 microsecond

Table 6.3 FCP Target Overhead and Control Constant

Incoming Frame Buffer	2
Login Guaranteed Buffer	0
Maximum Frame Size	2048 Bytes
Full Duplex	No
Command Execution Overhead	47.4 microsecond
FCP_RSP Generation overhead	10.3 microsecond
DMA Bandwidth	1064 Mbytes per second
DMA Scheduler Overhead	15.5 microsecond
DMA Round Robin	No
Incoming Frame Processing Overhead	400 nanosecond
Single Frame Transfer Overhead	20.2 microsecond

and the rest of the system is set to be 1046 MB/s corresponding to the 133 MHz 64 bits optimal bus speed. However, the overhead of the DMA scheduler is set to about 15 microseconds. At the same time, the DMA scheduling policy use FCFS.

6.2.2 Control Variables and Result Collection

Other control variables are summarized in Table 6.4. The control variables include four types of parameters, namely the FC-AL schedule, the number of HDDs, the HDD access time and the I/O workload. The FC-AL schedule is set to either normal or CMDF. The HDD number is set to 16 initially. The effects on variety of HDD number are also simulated. The HDD access time is set to 2.0 milliseconds as the base. The effects of HDD access time increases or decreases are evaluated. The closed-system I/O workloads are used with different “queue depth” per HDDs, I/O size logarithmically increases from 2 KB to 1 MB and three request types: pure read, pure write and

Table 6.4 Configuration Variables

FC-AL Schedule	Normal CMDF
HDD Number	16, and 2 to 126
HDD Access Time	8.0 millisecond 4.0 millisecond 2.0 millisecond 1.0 millisecond 0.5 millisecond
I/O Workload	Queue Depth (I/Os per HDD): 1, and 2 to 16 Options: 100% Read 100% Write Read Write mixture (with various Read fraction) Size: 2KB to 1MB logarithmically increase

read/write mixtures. The simulations were done first on a configuration of 16 HDDs system with other variable changes. Once the target system is determined, different workloads were applied. Each of these workload configurations, e.g., “1 queue-depth 2 KB pure read”, is scheduled in one simulation running. The simulations were run on a Pentium IV 3.0 G CPU computer with Linux operating system.

Each simulation running lasts for a sufficient period so that enough I/Os can be processed to achieve stable measurements. The I/O statistics during simulation “warm-up” and “shut-down” periods are excluded in the measurement. The “warm-up” period is considered complete when first ten I/Os per HDD for one queue-depth are processed. If queue depth is n and HDD number is 16, the total number of “warm-up” I/Os is 160n. Similarly, the last 160n I/Os are not counted during the “shut-down” period. Thus, if the targeted measurement I/O number is N, the total I/O to process is (320n+N).

6.3 Result Analysis

The simulation results are presented and analyzed in this section. A FC-AL storage system with 16 HDDs is first simulated to compare the I/O performance between the normal schedule and the CMDF schedule. Event though FC-AL can support up to 127 devices in a single loop, it is often criticized, with no exception from other storage interfaces, for not being able to provide enough sustainable bandwidth for all high performance HDDs attached. For example, with 50 MB/s HDDs, it can only theoretically support up to four HDDs when the link speed is 2 gigabit per second (200MB/s). However, the actual real-life application fortunately does not request for such high transfer throughput to every disk at the same time. An FC-AL based storage system with 16 HDDs attached is studied as the base system, since it gives a fairy large attachment and receives less criticism for saturation. After evaluation on the base system, the simulation studies are conducted for other performance factors such as the effect of different read fraction, the effect of HDD speed, the effect of HDD number and the effect of queue depth.

6.3.1 Based Line System Performance Improvement

The simulation results of a 16 HDDs storage system with or without the CMDF algorithm are presented in Figure 6.3~6.5 for data throughput (MB/s), I/O throughput (IOPS) and average response time (millisecond). It can be seen in Figure 6.3 that the

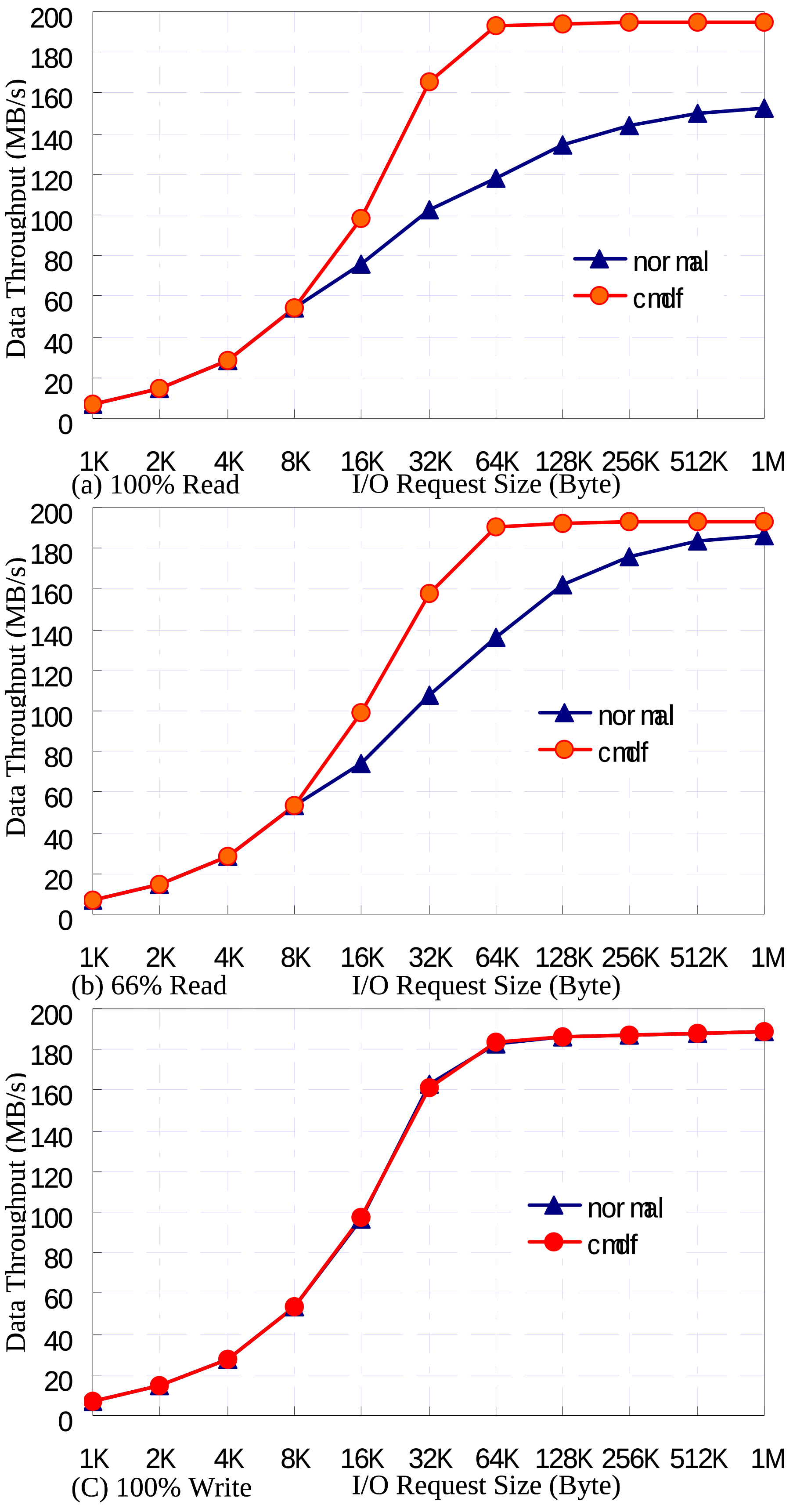


Figure 6.3 Based Line Storage System Data Throughput Comparison

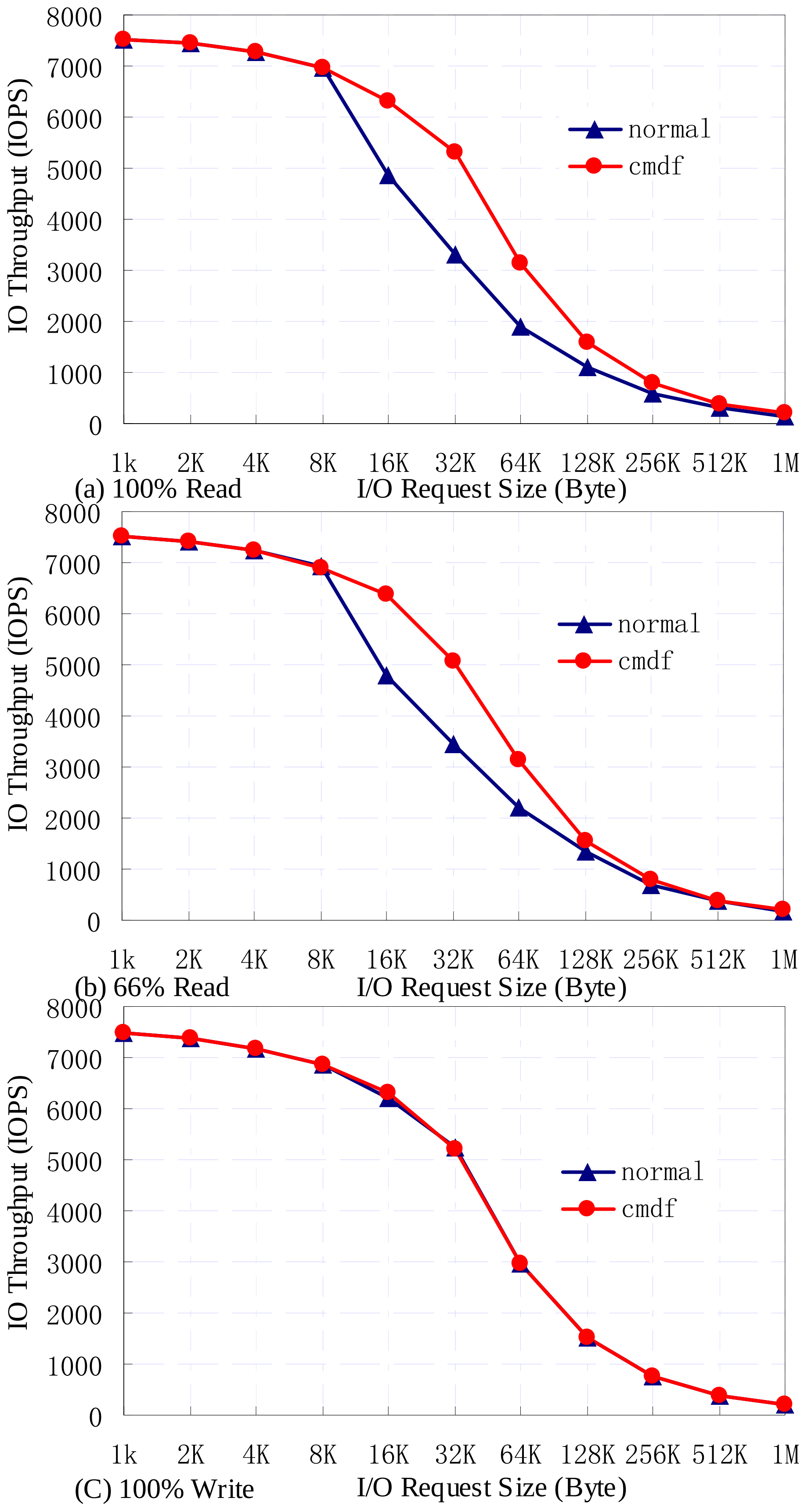
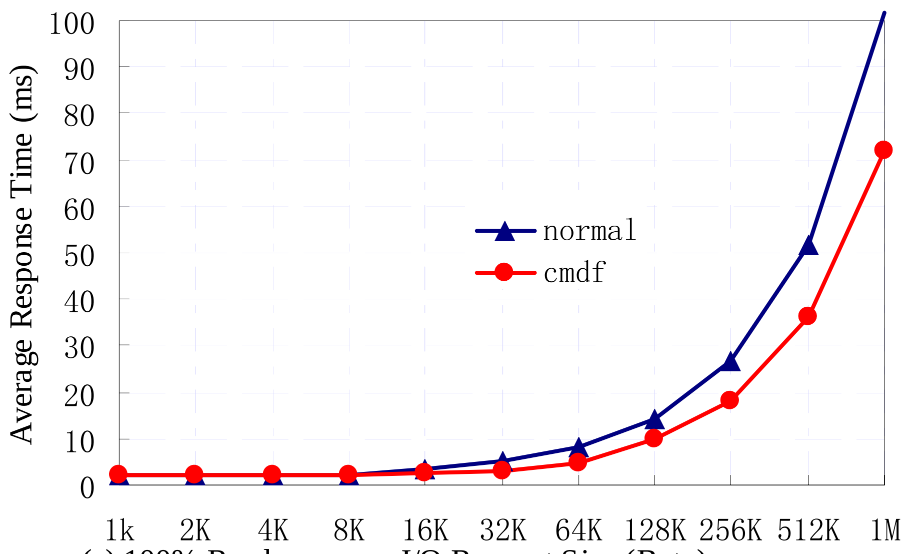
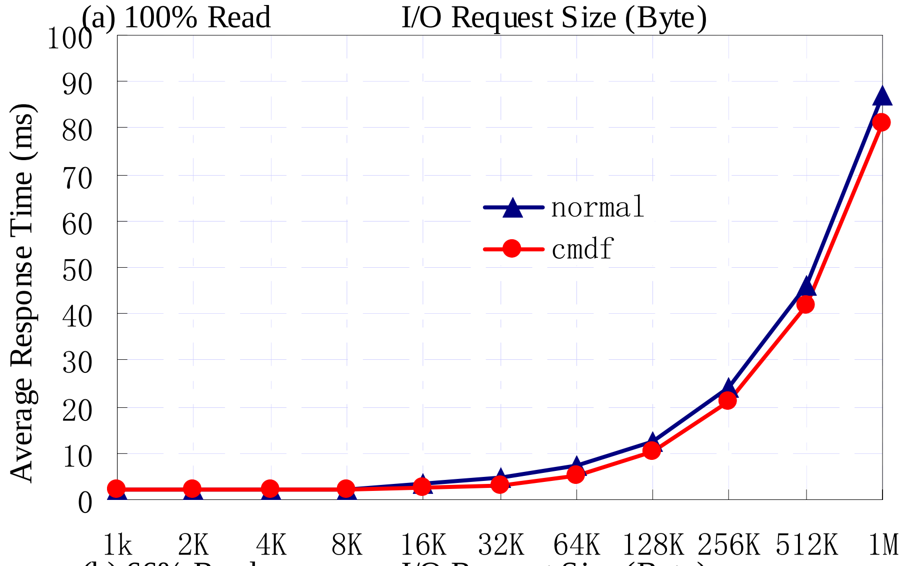


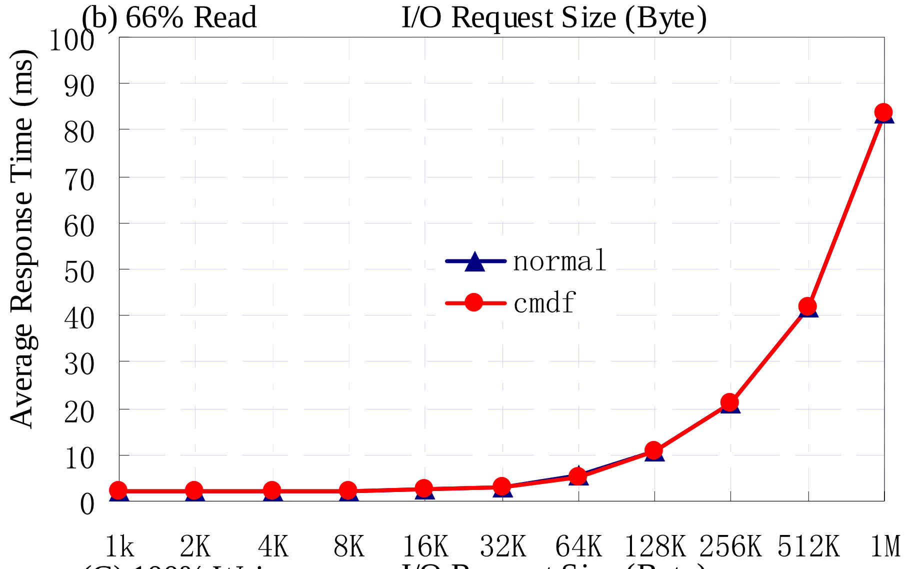
Figure 6.4 Based Line Storage System I/O Throughput Comparison



(a) 100% Read I/O Request Size (Byte)



(b) 66% Read I/O Request Size (Byte)



(C) 100% Write I/O Request Size (Byte)

Figure 6.5 Based Line Storage System Average Response Time

data throughput raises sharply as the I/O size increases, for all cases. The measured data throughput achieves about 7 MB/s at smallest I/O size (1KB) and grows when I/O size increases and approaches to its maximum throughput. By contrast, the I/O throughput (IOPS) declines as I/O size increase, as shown in Figure 6.4. It achieves about 7.5 K IOPS when I/O size is 1 KB and reduces to half when I/O size is 32 KB. I/O average response time shown in Figure 6.5 increases as the I/O size increase. With the close-system workload, these three performance metrics are not independent, although they are collected independently. The data throughput equals to the I/O throughput multiplying the requests size, while the average response time could be calculated by dividing the queue depth by the I/O throughput. This thesis is focusing on data throughput analysis and the interpretations are extendable to the other two.

For the purpose of comparison, the I/O performance measurements for normal schedule and for CMDF are plotted together in each figure. The triangle-mark blue line presents the performance of the normal schedule and the round-dot red line presents the CMDF. It can be seen from Figure 6.3 that the CMDF achieves significant performance improvement when I/O request size is greater than 8 KB for pure read (100% read) and read/write mixture (66% read) access. The CMDF however does not improve the performance for pure write. The data throughput relative improvements by CMDF are tabulated in Table 6.5. It is clear that the CMDF effectively improves the I/O performance for read I/O. The improvements are achieved only when the I/O size is larger than 8 KB. As the I/O size further increases, the data throughput achieved by the

Table 6.5 CMDF Data Throughput Relative Improvement

I/OSize	PureRead			PureWrite			ReadWrite		
	NORM	CMDF	Δ%	NORM	CMDF	Δ%	NORM	CMDF	Δ%
1KB	7.32	7.32	0.00%	7.27	7.27	0.00%	7.31	7.31	0.00%
2KB	14.47	14.47	0.00%	14.37	14.37	0.00%	14.44	14.44	0.00%
4KB	28.29	28.29	0.00%	27.97	27.97	0.00%	28.21	28.21	0.00%
8KB	54.13	54.13	0.00%	53.18	53.18	0.00%	53.86	53.86	0.00%
16KB	75.66	98.1	29.66%	96.62	97.81	1.23%	74.26	99.05	33.38%
32KB	102.56	165.09	60.97%	163.09	161.5	-0.97%	107.54	157.69	46.63%
64KB	117.72	193.14	64.07%	182.93	183.63	0.38%	146.05	190.66	30.54%
128KB	134.22	194.09	44.61%	186.07	186.43	0.19%	162.00	192.19	18.64%
256KB	144.33	194.57	34.81%	187.42	187.42	0.00%	176.00	192.81	9.55%
512KB	149.98	194.82	29.90%	188.23	188.23	0.00%	183.59	192.9	5.07%
1MB	152.97	194.94	27.44%	188.57	188.57	0.00%	186.37	193.03	3.57%

CMDF approaches to the maximum that is bounded by the loop nominate bandwidth of 200MB/s.

For read access, when I/O size is small (smaller than 16KB), the CMDF does not achieve better throughput because the overhead of the I/O commands dominates the response time for small I/O. When the request size is small, the loop occupying time by a target HDD to transfer data to the initiator is short. The target HDD releases the loop quickly after access completion. Consequently, the probability of multiple devices simultaneously arbitrating for loop access is small. In other words, the fairness access window is small and the storage controller can quickly get access on the next access window without the CMDF. When request sizes become bigger, the target HDDs take a longer time to transfer data and situations of multiple devices arbitrating for loop access become more frequent. In such environment, the storage controller has to wait for the next access window until all devices have finished their data transfer for normal schedule. By contrast, the CMDF does not need to wait for as long. Once the

FCP_CMND frame is ready, the storage controller can start arbitration for accessing the loop with the CMDF schedule and send commands earlier. The target HDDs thus can start to serve the request in advance and performance gain is achieved.

Two main reasons may account for no obvious benefits from the CMDF when access is writing. In the first hand, since the writing data are all transferred from the initiator to the writing target, the initiator shall be mostly holding the loop when a new command arrives. It can therefore send the command immediately without the CMDF. In the second hand, during a write transaction, a target holds the loop only for FCP_XFER_RDY and FCP_RSP sequence, both of which are small in size (48 bytes and 60 bytes) and transferred in a single frame sequences. The loop holding time is very short, and consequently the command would not be delayed for long owing the Fairness Access Algorithm if it ever happens.

For the case of read/write mixture with 66% read, the CMDF achieves relative smaller performance gain than pure read. It can be seen from Figure 6.3 (a) and (b) that the CMDF achieves identical data throughput between read and read/write mixture, although the read maximum throughput is slightly higher than read /write mixture due to the fact that the write require additional process of FCP_XFER_RDY. The normal schedule without CMDF however achieves a noticeable difference in data throughput between the pure read and the read/write mixture. The HDD’s data transferring for read request will be blocking the storage controller to send commands earlier. The effect of read fraction is investigated in the next subsection.

6.3.2 Other Performance Factor Analysis

The effect of other performance factors for the performance improvement by the CMDF is investigated in this subsection. The effect of read fraction is first studied. With read fraction linearly increasing, the performance improvements by the CMDF are monitored. The effect of HDD speed is examined next. It is aiming to evaluate the CMDF with 100% and 200% HDD speed increase or decrease. Following this, the effect of HDD number increasing is discussed. The number of HDD increases from 2 to 126 and the I/O performances achieved by the two schedules are compared. Lastly, the effect of queue depth per HDD is presented.

6.3.2.1 Effect of Read Fraction

Figure 6.6 shows the simulation results for the effect of read fraction for the CMDF compared to the normal schedule. Each diagram in the figure corresponds to one read fraction from 0.1 to 1.0. The round-dot red lines plot the data throughput achieved by the CMDF as the I/O size increases, while the triangle-mark blue lines show the data throughput of a normal schedule. It can be seen that the improvement of data throughput achieved by the CMDF compared to the normal schedule becomes more significant as the read fraction increase. It is also noted that the data throughput of the CMDF only slightly increases when the read fraction increases. The performance improvements are mainly due to the lower throughput of the normal schedule when read fractions become higher. Under the normal schedule, the command-blocking factor becomes more significant as read fraction increases and the data throughput consequently degrades.

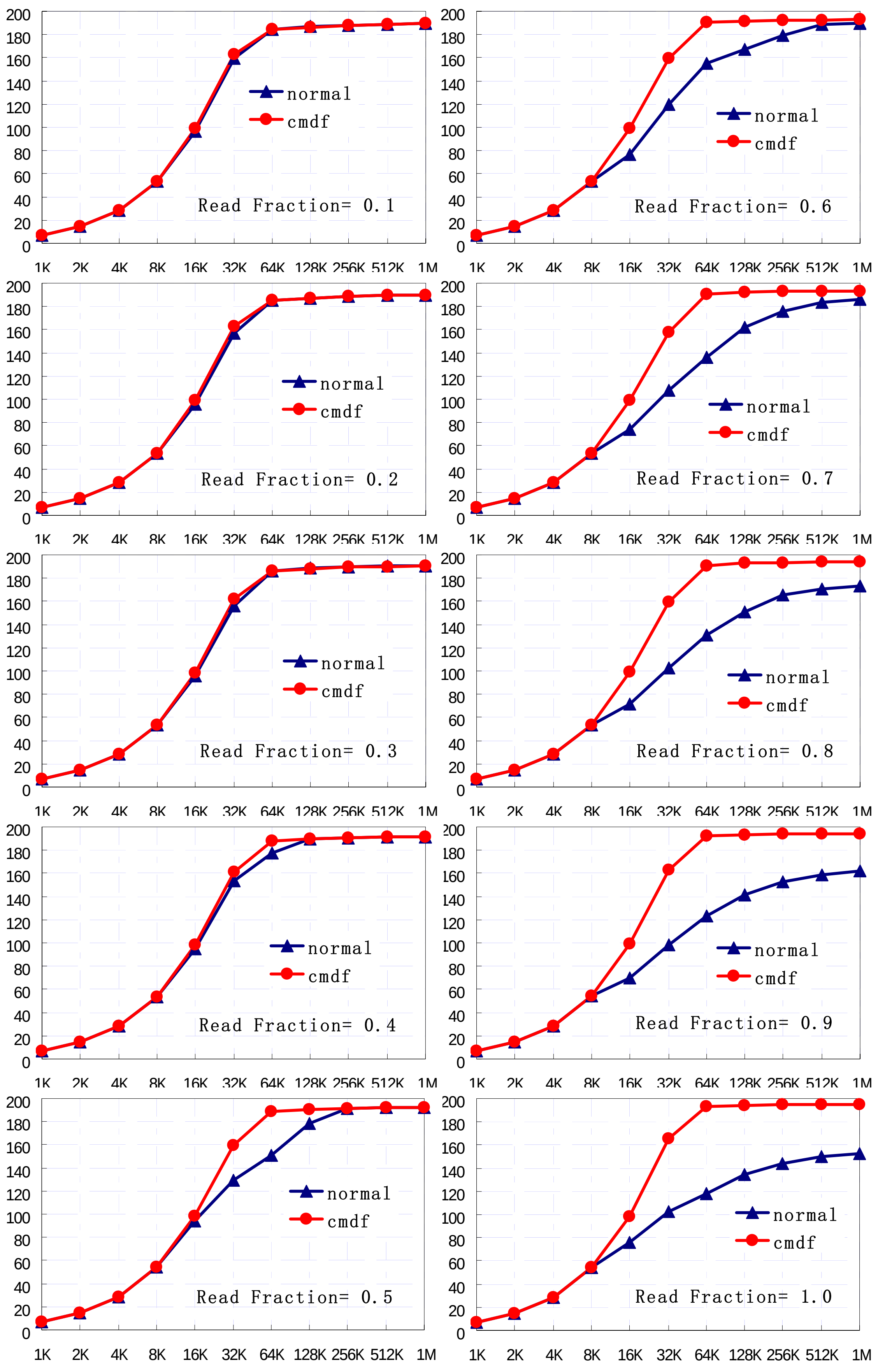


Figure 6.6 Effect of Read Fraction for CMDF

6.3.2.2 Effect of HDD Speed

Since the benefit brought by the CMDF is also contributed by the HDD for executing the I/O command in advance, the speed of the HDD shall therefore also play an important role for the overall performance improvement. The effect of the HDD speed is presented in this subsection. Based on the parameters used for HDD in the previous section, the speed of the HDD is adjusted for 100% and 200% increments to be 1.0 and 0.5 milliseconds access time, and 100% and 200% decrements to be 4.0 and 8.0 milliseconds. The internal data transfer rate remains 50MB/s.

Figure 6.7 (a), (b) and (c) show the simulation results of the data throughput achieved by the CMDF compared to the normal schedule when I/O size increases for three different access patterns of pure read, read/write mixture and pure write. In each diagram, the red dotted lines represent the throughput achieved by the CMDF, each of them corresponding to one access time as labeled in the legend block. The blue solid line represents the data throughput achieved by the normal schedule. It is clear that the CMDF achieved significant improvement compared to the normal schedule for all different HDD speeds when the I/O access is pure read. For faster HDD, the CMDF starts achieving performance improvement from smaller I/O sizes since the data transferring time is comparable to the HDD data preparing time (access time). It is also noted that with faster HDD, the storage system achieves higher aggregate throughput. Because of the ceiling effect of the nominate bandwidth (200MB/s), the throughput gain achieved by the CMDF is relatively reduced as the HDD speed becomes faster.

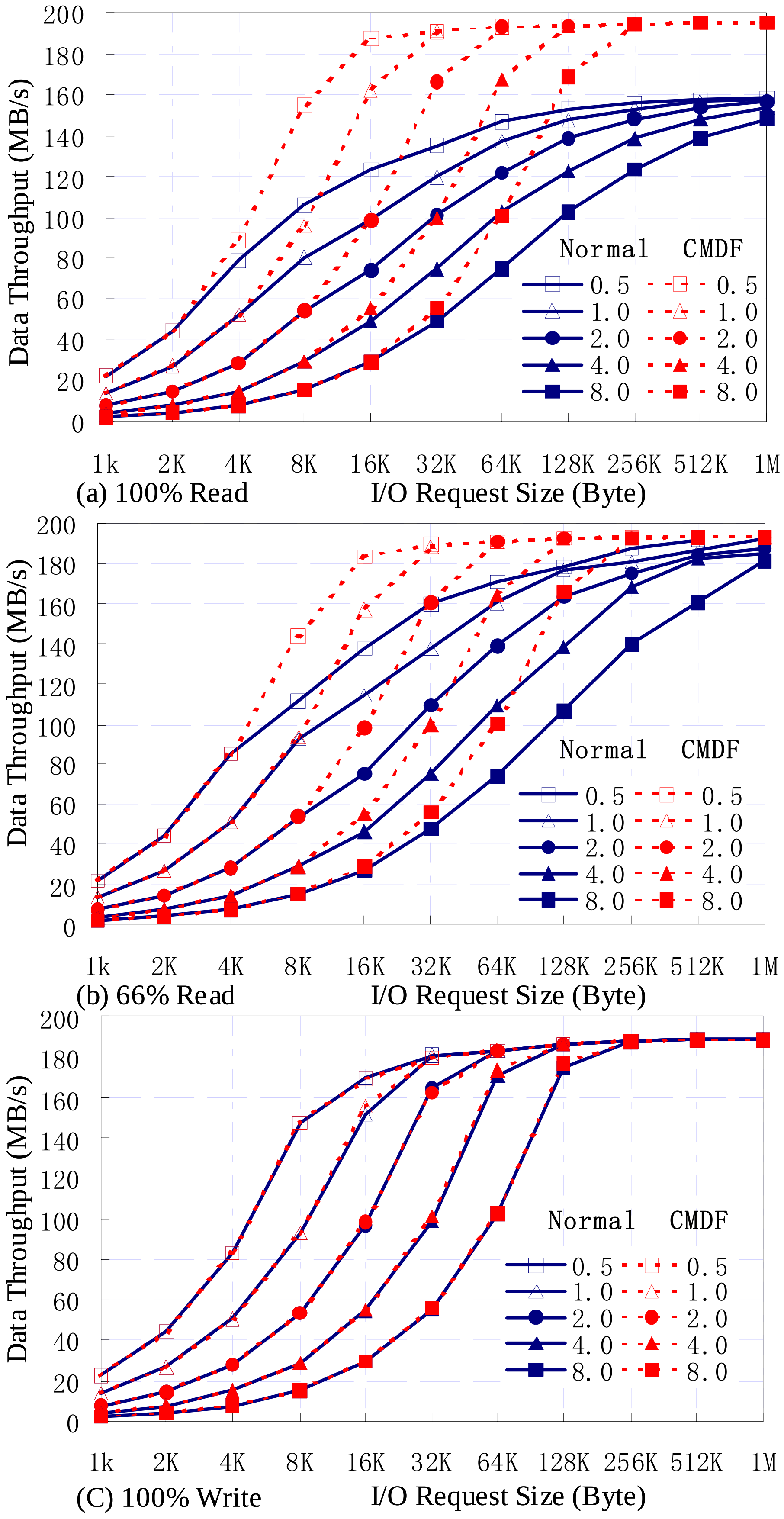


Figure 6.7 Effect of HDD Speed for CMDF

6.3.2.3 Effect of Number of HDD

Figure 6.8 shows the simulation results of aggregated data throughput with CMDF or normal schedule for 4KB, 32KB, 64KB and 256KB read when HDD number changes from 2 to 126. The round-dot red lines in the diagrams show the data throughput achieved by the CMDF while the triangle-mark blue lines represent the normal schedule throughput.

It can be seen in the top most diagram of Figure 6.8, the CMDF does not improve performance when the I/O size is 4KB. It is however interesting to see that the aggregated data throughput increases as more HDDs are attached to the loop and settles down to about 90 MB/s around 48 HDDs. After this, the loop becomes saturated and the throughput does not increase with additional HDD but rather slightly declines. The reason why the CMDF does not improve throughput has been discussed in the previous subsection for the case of 16 HDDs. The same reason is applicable here.

The second diagram in Figure 6.8 shows the simulation results of 32KB read with the two schedules. It can be seen that the CMDF does not improve the throughput when the HDD's number is less than eight, while it achieves significant improvement when more HDD's are attached. With given system parameters, the total HDD processing time is about 2.6 millisecond including 0.6 milliseconds HDD's transfer time at 50MB/s internal transfer rate for 32KB I/O request, while the ideal FC transfer time is about 0.16 milliseconds for 2 gigabit per second link rate (32KB/200MBps). It is possible to schedule all eight requests (one per HDD) in the 2.6 milliseconds time window for data transferring with idles in-between. With such idle periods, the storage

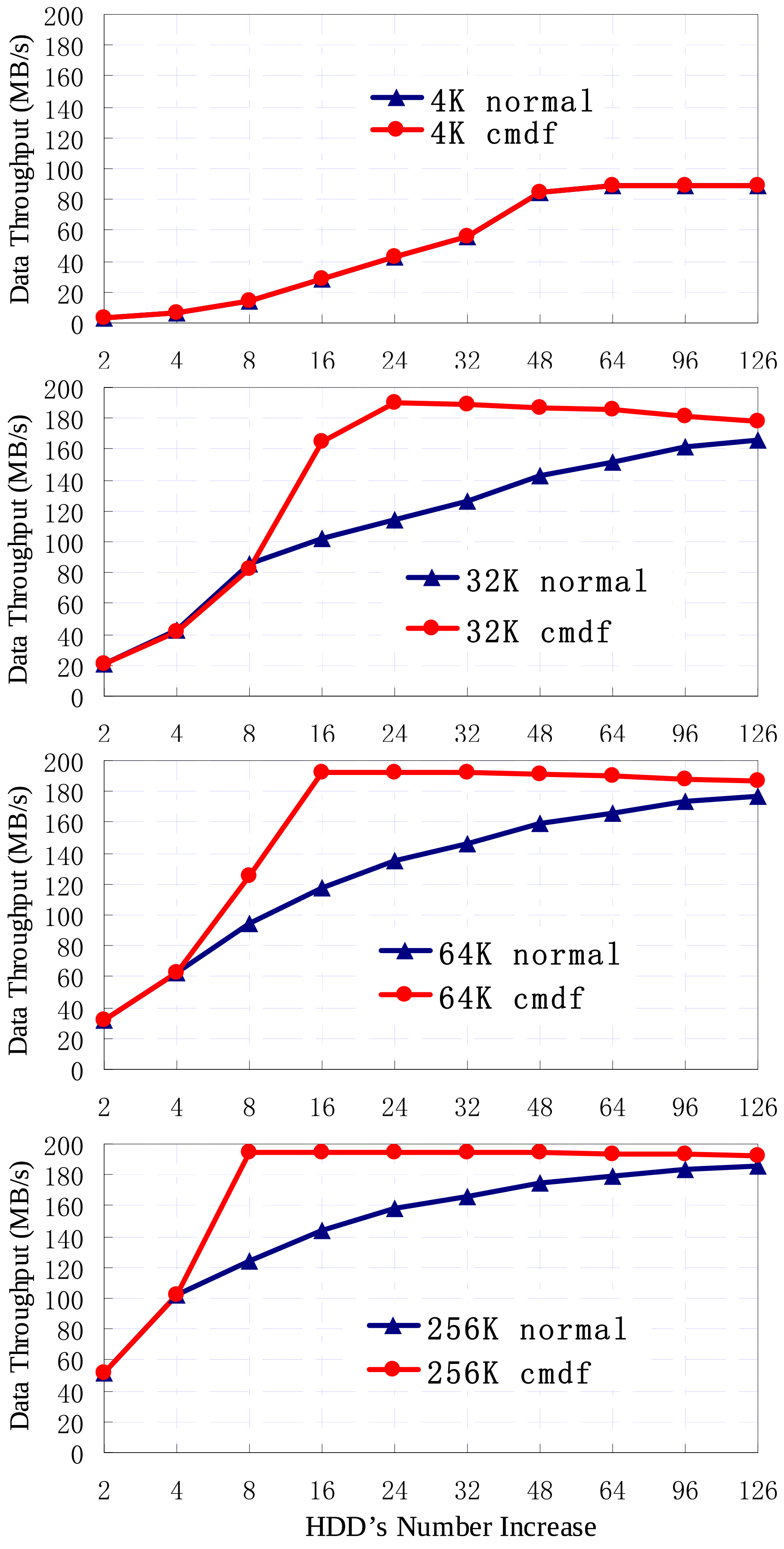


Figure 6.8 Effect of Number of HDD for CMDF

controller hence has chance to arbitrate the loop to send a new command. The CMDF therefore does not achieve improvement when the number of HDD is small. As the HDD number increases to sixteen and greater, it becomes very hard to schedule all 16 or more sections of data transferring in the same time-window, with idles in between, especially as the actual loop occupying time is considerably longer than 0.16 millisecondsl. In such situation, the HBA may often be blocked from sending new commands by multiple sections of data transfer. As more HDDs are attached to the loop, the normal schedule achieves higher aggregated throughput because more HDDs are ready to send data even though some of the commands are delayed. The aggregated throughput grows steadily as the HDD number increases with normal schedule. By contrast, the throughput achieved by the CMDF grows quickly to the maximum around 195 MB/s, and declines slightly as the HDD number increases further from 24, owing to the increment of per-port-delays.

The other two diagrams in Figure 6.8 show the data throughput of the two schedules for the bigger I/O size (64KB and 256KB) when the HDD number changes. It is clear that the CMDF achieves significant improvement when the HDD number is more than four. Under the CMDF, the aggregated data throughput quickly reaches its peak at 16 HDD for 64KB I/O, and at 8 HDD for 256KB I/O. After that, the throughput slightly degrades due to the additional delay of HDD's per-port-delay. For bigger I/O, it is also noted that the throughput grow rate under the normal schedule become smaller as more HDDs are attached.

6.3.2.4 The Effect of Queue Depth

Figure 6.9 shows the data throughputs achieved by the CMDF and the normal schedule on a 16 HDD’s storage system for 4KB, 32KB, 64KB and 256KB read when number of outstanding I/O requests per HDD increases from 1 to 16. The round-dot red lines in the diagrams show the data throughput achieved by the CMDF while the triangle-mark blue lines represent the throughput of the normal schedule.

When I/O size is small (4KB), the CMDF and the normal schedule achieve identical throughput regardless of the number of outstanding I/O per HDD (queue depth). As stated at the beginning of this chapter, the storage controller spends about 43.9 microseconds to execute each I/O, which results in maximum of 22K IOPS I/O processing capacity. As the queue depth increases, the storage controller I/O processing capacity limits the maximum aggregate throughput, that is, about 88 MB/s (22K times 4KB).

When I/O size is 32KB, the aggregated throughput achieved by the CMDF rises to the maximum when queue depth per HDD is two. When queue depth further increases, the throughput remains unchanged since there is no room for improvement due to the ceiling effect of the maximum bandwidth. By contrast, without the CMDF, the throughput is limited by the effect of command blocking by the data transferring. When queue depth increases, the storage controller can send multiple I/O commands to each HDD when the loop is held by the storage controller. The HDDs are therefore kept in busy, and the aggregated throughput increases as the queue depth grows until the loop is saturated. After saturation, the HDD are mostly busy for either preparing

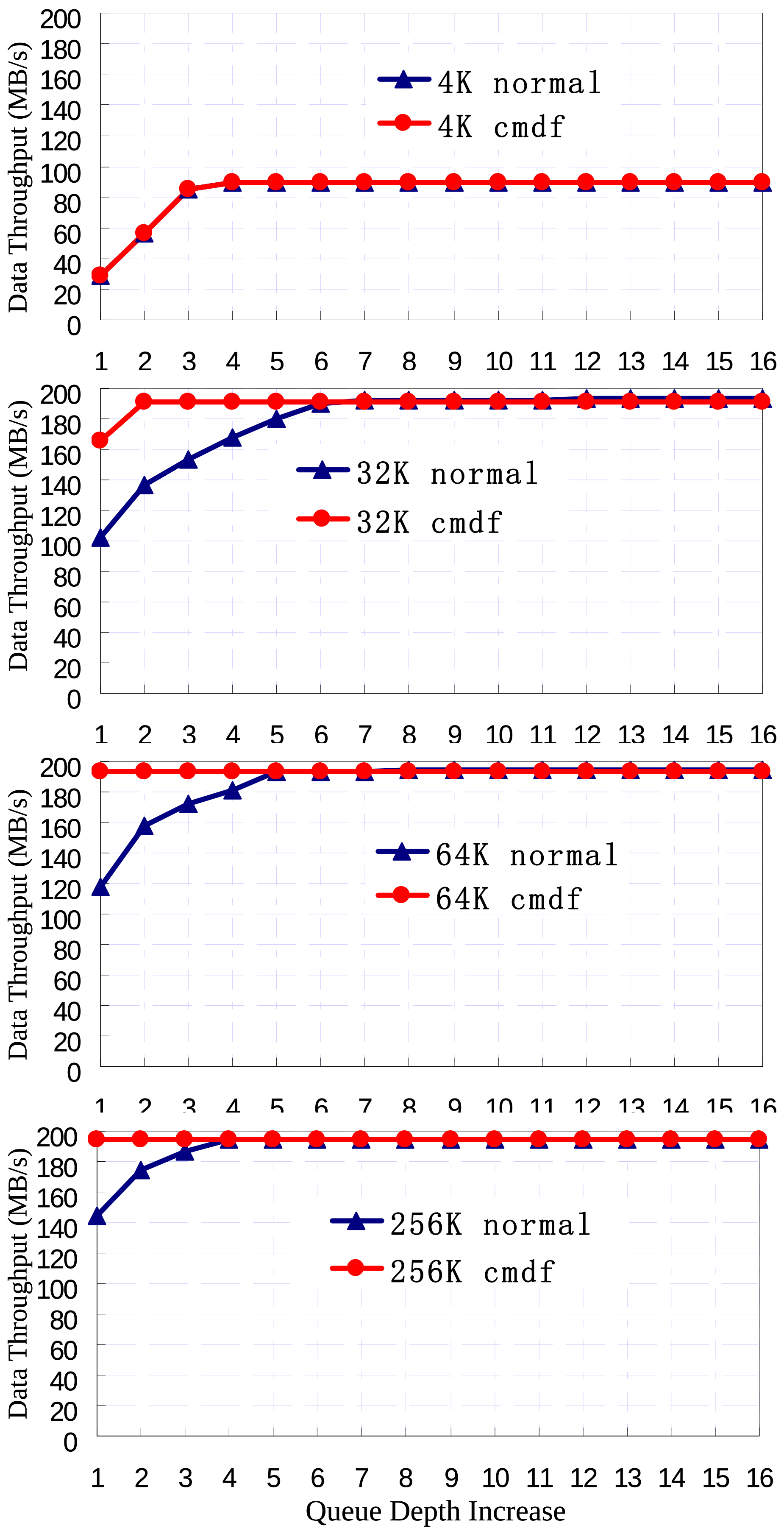


Figure 6.9 Effect of Queue Depth per HDD

the data or transferring data through the FC-AL loop when it receives a new command. Therefore no benefit can be seen for the CMDF when queue depth is deep enough (greater than 6 for the case of 32KB I/O).

For the cases of 64KB and 256K, the aggregated data throughput achieved by the CMDF reaches the maximum even for the case of one queue depth. The FC transfer time for 64KB data is about 0.3 milliseconds (64KB/50MBps). All 16 requests accessing the 16 HDDs would take about 4.8 milliseconds to complete the data transferring. With the CMDF algorithm, a new request command will be issued and sent to the corresponding HDD at about 0.3439 milliseconds (0.3 + 0.0439) after the data transferring starts, if the data requested by these 16 I/Os are assumed to be transferred continuously. Once the corresponding HDD receives the command, it can start to prepare the requested data. With the assumption of 2 milliseconds overhead and 50MB/s internal transfer rate, the HDD will be ready to transfer data in 3.125 milliseconds. Adding 0.3439 milliseconds, the HDD will be ready to transfer data at about 3.5 milliseconds that is before the completion time of 4.8 milliseconds. The requested data can therefore be continuously transferred, and the loop idle periods will be kept in minimum. The maximum throughput is therefore achieved. By contrast, without the CMDF algorithm, the command may be delayed by the data transferring and the HDD would not be able to prepare the requested data in advance. It would cause the loop to become idle after completion of previous batch of request and the throughput achieved by normal schedule is therefore degraded.

6.4 Summary

This chapter has conducted performance evaluations on the effect of Command-First Algorithm compared to the normal schedule on FC-AL storage system in this chapter. The overall method for the performance evaluation has been described at the beginning, followed by detailing the simulative storage system configuration. The I/O performances in terms of data throughput (MB/s), I/O throughput (IOPS) and the average I/O response time (millisecond) for the base system consisting 16 HDDs have been compared for the two schedules. The Command-First Algorithm has achieved up to 50% throughput improvement for medium size I/Os. The effects of Command-First Algorithm have been further evaluated in some extended environments, such as different number of HDDs, increasing workload (in the form of deeper queue-depth per HDD). In all situations, the Command-First Algorithm has almost no negative effect.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The goals of this thesis are to develop a detailed and accurate simulation model for high-end storage systems that employ the FC-AL as back-end connection for HDDs, and to evaluate the proposed Command-First Algorithm for an FC-AL based storage system through the simulation model. This thesis is summarized as following.

Firstly, a novel way of simulating FC-AL based storage system has been presented. A modular simulation model hierarchy for an FC-AL based storage system has been developed. The FC-AL transmission model was first introduced, and then the L_Port's functionalities including the LPSM and the Alternative Buffer-to-buffer flow control were modeled. On top of that, the FCP HBA model was developed to simulate the FCP SCSI transaction. With additional support of an HBA device driver module and HDD firmware functions module, the system level simulation tool integration has been delivered.

Secondly, the simulation model has been calibrated and validated. By checking signal transmission events against the actual FC analyzer's traces, the model has been verified in term of lowest level transmission. By examining the general I/O performance trends, the model has been proven to agree with the expectation. The actual experiments have been conducted and the experimental results have been compared to the simulated results. The results show that the FC-AL model is accurate with an error range of less than 3% for read operation.

Thirdly, the Command-First Algorithm has been proposed in three different levels. The fist level is to place the command in front of data so that the command can be sent earlier. The second level is the command first arbitration that forces the storage controller to operate in unfair mode for command frame transferring. The preemptive command transferring, the third level, is to further enforce the storage controller to send the command preemptively.

Finally, the evaluation of the proposed Command-First Algorithm have been conducted and compared to a normal FC schedule. The simulation measurements have shown that the performance gains achieved by the algorithm are up to 50% improvement compared to the normal schedule in certain conditions, and that there are no negative effects of Command-First Algorithm.

7.2 Future Work

The proposed Command-First Algorithm so far has been proven an effective schedule for FC-AL based storage systems. It is however worthwhile to note that the evaluation has not included the benefit that might be brought along with the algorithm when the optimal scheduling is enabled in HDD. Future work may involve a more detailed HDD model to evaluate this effect. On the other hand, the solid effect of the algorithm has yet been evaluated by actual implementation. The natural extension of the work is to carry a prototype that enables the algorithm. Furthermore, the realistic application environment whereby the algorithm can achieve its significance is another consideration for the future work.

Bibliography

[1] Yao-Long Zhu, Shun-Yu Zhu and Hui Xiong, "Performance Analysis and Testing of the Storage Area Network," *the 19th IEEE Symposium on Mass Storage Systems and Technologies*, April 2002.

[2] C.Y. Wang, F.Zhou, Y.L.Zhu, C.T. Chong, B. Hou, W.Y.Xi, "Simulation of Fibre Channel Storage Area Network Using SANSim," *the 11th IEEE International Conference on Network (ICON2003)*, October 2003.

[3] C.Y. Wang, F.Zhou, Y.L.Zhu, C.T. Chong, B.Hou, W.Y.Xi, "Simulation and Analysis of FC Network," *the 28th Annual IEEE Conference on Local Computer Networks (LCN2003)*, October 2003.

[4] Y.L. Zhu, C.Y. Wang, W.Y. Xi, F.Zhou, "SANSim - A Simulation And Design Platform of Storage Area Network," *the 12th NASA Goddard Conference on Mass Storage Systems and Technologies / the 21st IEEE Symposium on Mass Storage Systems*, April 2004.

[5] E. Grochowski and R. D. Halem, "Technological Impact of Magnetic Hard Disk Drives on Storage Systems," *IBM System Journal*, Vol. 42, No. 2, Pages:338-346, 2003.

[6] R. J. T. Morris and B. J. Truskowski, "The Evolution of Storage Systems," *IBM System Journal*, Vol. 42, No. 2, Pages: 205-217, 2003.

[7] David A. Patterson, Garth Gibson, and Randy H. Katz, “A Case for Redundant Arrays of Inexpensive Disks (RAID),” *International Conference on Management of Data (SIGMOD)*, Pages: 109-116, June 1988.

[8] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz and David A. Patterson, “RAID: High-Performance, Reliable Secondary Storage,” *ACM Computing Surveys (CSUR)*, Vol. 26, No. 2, Pages 145-185, June 1994.

[9] ANSI X3.272:1996, “Information Technology – Fibre Channel Arbitrated Loop (FCAL),” American National Standard Institute, Inc., 1996.

[10] ANSI X3.230:1994, “Fibre Channel Physical and Signaling Interface (FC-PH),” American National Standard Institute, Inc., 1994.

[11] ANSI X3.269:1996, Fibre Channel Protocol for SCSI (FCP), American National Standard Institute, Inc., 1996.

[12] WWW webpage for FC Projects on Technical Committee T11 homepage, http://www.t11.org/Index.html.

[13] Jeffrey D. Stai, “The Fibre Channel Bench Reference,” ENDL Publications, ISBN 1-879936-17-8, 1st Edition, May 1995.

[14] Elizabeth Shriver, Bruce K. Hillyer, and A vi Silberschatz, “Performance Analysis of Storage Systems,” Performance Evaluation, LNCS 1769, Pages: 33-50, 2000.

[15] Xavier Molero, Federico Silla, Vicente Santonja and José Duato, “Modeling and Simulation of Storage Area Networks,” *the 8th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOT2000)*, September 2000.

[16] Petra Berenbrink, André Brinkmann and Christian Scheideler, "SIMLAB - A Simulation Environment for Storage Area Networks," *the 9th Euromicro Workshop on Parallel and Distributed Processing (PDP)*, 2000.

[17] John S. Bucy, Gregory R. Ganger, "The DiskSim Simulation Environment Version 3.0 Reference Manual," *http://www.pdl.cmu.edu/PDL-FTP/ DriveChar/CMU-CS-03-102_abs.html*, January 2003.

[18] Gregory R. Ganger and Yale N. Patt., "Using System-Level Models to Evaluate I/O Subsystem Designs," *IEEE Transactions on Computers*, Vol. 47, Issues 6, Pages:667-678, 1998.

[19] John Wilkes, "The Pantheon storage-system simulator," HPL-SSP-95-14, Hewlett-Packard Laboratories technical report, May 1996.

[20] John R. Heath and Peter J. Yakutis, "High-Speed Storage Area networks Using Fibre Channel Arbitrated Loop Interconnect," *IEEE Network 2000*, Pages: 51-56, April 2000.

[21] David H.C. Du, Tai-Sheng Chang, Jenwei Hsieh, Yuewei Wang and Sangyup Shim, "Interface Comparisons: SSA versus FC-AL," *IEEE Concurrency*, Vol. 6, No. 2, April-June 1998.

[22] Shenze Che and Manu Thapar, "Fibre Channel Storage Interface for Video-on-Demand Servers," HPL-95-125, Hewlett-Packard Laboratories technical report, November 1995.

[23] Jae-Chang Namgoong and Chan-Ik Park, "Design and Implementation of a Fibre Channel Network Driver for SAN-Attached RAID Controllers," *the 8th*

International Conference on Parallel and Distributed Systems (ICPADS2001), June 2001.

[24] Vishal Sinha and David H. C. Du, “Switched FC-AL: An Arbitrated Loop Attachment for Fibre Channel Switches,” *the 17th IEEE Symposium on Mass Storage Systems*, March 2000.

[25] Zhang Hong, Koay Teong Beng, Venugopalan Pallayil, Zhang Yilu, John R Potter, and Lawrence Wong Wai Choong, “Fibre Channel Storage Area Network Design for an Acoustic Camera System with 1.6 Gbits/s Bandwidth,” *in Proc. of IEEE Region 10 International Conference on Electrical and Electronic Technology* (TENCON 2001), August 2001.

[26] Thomas M. Ruwart, “Performance Characterization of Large and Long Fibre Channel Arbitrated Loops,” *16th IEEE Symposium on Mass Storage Systems,* March 1999.

[27] Denise Colon, “SANs Demystified,” McGraw Hill, ISBN: 0071396586, October 2002.

[28] Ralph H. Thornburgh & Barry J. Schoenborn, “Storage Area Networks – Designing and Implementing a Mass Storage System,” 1st Edition, Prentice Hall PTR, ISBN: 0130279595, September 2000.

[29] Marc Farley, “Building Storage Networks,” 1st Edition, McGraw-Hill Osborne Media, ISBN: 0072130725, February 2000.

[30] Tom Clark, “Designing Storage Area Networks – A Practical Reference for Implementing Fibre Channel and IP SANs,” 2nd Edition, Addison-Wesley Professional, ISBN: 0321136500, April 2003.

[31] Bruce L. Worthington, Gregory R. Ganger and Yale N. Patt, “Scheduling Algorithms for Modern Disk Drives,” *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1994.

[32] Bruce L. Worthington, “Aggressive Centralized and Distributed Scheduling of Disk Requests,” PhD Thesis, Department of Computer Science and Engineering, University of Michigan, June 1995.

[33] Chris Ruemmler and John Wilkes, “An Introduction to Disk Drive Modeling,” *IEEE Computer*, Vol. 27, No.3, Pages:17-28, March 1994.

[34] Edward Kihyen Lee and Randy H. Katz, “An Analytic Performance Model Of Disk Arrays,” *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1993.

[35] Gregory R. Ganger and Yale N. Patt, “The Process-Flow Model: Examining I/O Performance from the System’s Point of View,” *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1993.

[36] Ehud Finkelstein and Shlomo Weiss, “A PCI Bus simulation framework and some simulation results on PCI standard 2.1 latency limitations,” Journal of Systems Architecture, Vol. 47, Pages: 807-819, 2002.

[37] M.H. MacDougall, “Computer System Simulation: An Introduction”, Computing Surveys, Vol 2, No. 3, September 1970.

[38] SCSI Standard Architecture, http://www.t10.org/.

[39] Serial Storage Architecture, http://www.t10.org/.

[40] IO Meter, http://www.iometer.org/.

[41] Raj Jain, “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling”, John Wiley & Sons, Inc., ISBN 0-471-50336-3, April 1991.