Survey Report for

High-Speed Network Impact on DDBMS

Ning Zhang
Department of Computer Science
University of Waterloo

February 11, 2001

Abstract

DBMS as an I/O-intensive software has been focused on how to efficiently make use of cache, main memory, and disk storage based on the observation of their bandwidths in the memory hierarchy. In the past decade, Distributed Database Management Systems (DDBMS) has introduced computer network as another kind of data I/O media. Traditionally, computer networks were put on a lower layer than disk storage in the hierarchy. With the advent of broadband high-speed network and lightweight (low overhead) protocols, computer networks could have higher bandwidth than hard disk. How to reflect this fundamental change on the DDBMS or even traditional centralized DBMS architecture remains open. In this report, I shall survey the current status of high-speed networks especially gigabit/gigabyte networks, and the low overhead protocols and architecture. Introductions to Fibre Channel and Infiniband hardware architectures are included as the two major parts. Different approaches for incorporating high-speed network infrastructure to operating systems and high level network protocols are also introduced. At last, we give an analysis how we can incorporation application level protocols to DDBMS.

1. Introduction

1.1.Background and Motivation

The DBMS, as an I/O-intensive software, has largely been affected by the performance of its underlying hardware and operating system. The optimization of centralized database systems has been based on the observation that the speed of cache is about an order of magnitude faster than main memory and main memory is about an

order of magnitude faster than the disk storage [19]. Many cache-conscious main memory based algorithms and main memory conscious disk-based algorithms are developed to get better performance by getting data more close to the CPU. This has been proved to be a very effective way to manage data [20]. In Distributed Database Management Systems (DDBMS), our previous assumption was that the network was the bottleneck of the throughputs, so all the methodologies for query optimization and transaction management are to minimize the network communication, i.e. localize the computation as far as possible. But this assumption may not still hold for current hardware platforms.

The fundamental architecture of database systems has been around for nearly twenty years. In the last two decades, Moore's law operates on the hardware development -- CPU is getting faster, disk is getting bigger, and there are breakthroughs in long-dormant communication speeds. All of those have been changed and will change the architecture of DBMS, as well as DDBMS. In particular, the broadband high-speed networks will invalidate some of the previous assumptions of DDBMS, thus makes it necessary to redesign the architecture of next generation DDBMS. With current success of gigabit/gigabyte networks, network communication may not be the bottleneck any longer. Moreover high performance networks could be even faster than high performance local disk (for comparison, the Ultra-3 SCSI interface has a maximum of 160 MB/sec throughput, while Myrinet can achieve more than 200 MB/sec throughput with TCP/IP protocols). This implies that transferring data from the main memory of a remote host may be more efficient that loading data from local disks. If this is the case, what data structures, algorithms, and utilities of distributed database system have to be modified to adapt to the new changes? To answer this question, we have to see what have been done in the broadband and high-speed network infrastructure and what high performance communication services have available.

In the operating system, all the software components (network interface card drivers, implementation of protocols stacks, and etc.) related to network communication are called network subsystem. When two application programs want to communicate to each other, it is more complicated than simply sending a message from one host to another. To hide the complexity of underlying networks, the network subsystem of an operating

system usually provides some common services to the upper layer application programs. Since the needs of the application programs vary depending on their specific requirements, the common services may not be best suitable for the application program. For example, for Distributed Database Systems, the basic operations consist of distribute query processing, distribute transaction management, and distributed storage management. To be more specific, a DDBMS may have a distributed sorting algorithm that relies on the underlying network subsystem. The effectiveness and performance of this distributed sorting algorithm depends largely on what kind of common services and what the service quality the network subsystem provides. Another example is that the two-phase-commit distributed concurrency control protocol requires asynchronous message sending service (one of the most common services provided by the network subsystem). It is proved that two-phase-commit protocol cannot guarantee the global "all-or-nothing" property of transaction using asynchronous message passing. Hence, the network subsystem not only affects the performance of the upper level applications but also determines their functions and architecture. Based on this observation, it is crucial to understand what new network architecture and services have been provided to the high level application programs, so that we have the right baseline when we try to design new algorithms and architectures for the upper level DDBMS.

1.2.Scope of This Survey

In this report, I shall concentrate on the broadband, high-speed network infrastructure and the consequential changes on the operating systems and protocols to incorporate this infrastructure. However, I shall not dig too deep into physical level. All that detailed information can be found in referenced papers or specifications. What covered in this report are the architecture of the network or I/O system, and the corresponding lightweight protocols making the high-performance networks a reality. To be specific, we want to answer the following questions:

- What are the state-of-art broadband and high-speed networks?
- What components constitute these networks?

- What protocols and OS level modifications have been made to facilitate the new architecture?
- What end-to-end communication protocols have been adaptive to the changes?
- What the DDBMS can do with these changes in the author's point of view?

Since the hardware and network communities have been evolving very fast in the last decade, we may not include all the up-to-date technologies in this survey. I apologies for not being comprehensive, but because of the time bound, it is the best I have known.

Although, high-speed network seems to be a right breakthrough for distributed database systems, one must keeps in mind that some theoretical results are still valid. Due to the autonomy property of distributed database systems, the fundamental distributed assumption has remained unchanged, that is the limitation of local knowledge in a distributed system. Some distributed problems remain intractable no matter how fast the underlying network transmits data. For example, the distributed commit problem cannot be solved in asynchronous setting regardless the speed of network [22]. How to solve this kind of problems in synchronous and semi-synchronous networks is not in the scope of this report.

Another thought of distributed database systems on top of high-speed network is that it has much similarity with *shared-nothing* parallel database systems. It is natural to adopt the parallel algorithms used in parallel database to distributed database systems built on top of high performance networks. To some extent, the new generation distributed database system incorporating high-speed network is some kind of combination of parallel database system and traditional distributed database system, in that it manages data across high-speed network using parallel algorithms in order to get high degree of parallelism, and manages data on slow networks using traditional distributed algorithms. The parallel databases are beyond the scope of this paper. DeWitt and Gray gave an excellent overview on parallel database systems in [23].

1.3.Organization of this survey

The organization of this report is as follows:

- Section 2 presents the current achievements on the hardware of broadband and high-speed networks, focusing on gigabit local area networks. These architectures include Fibre Channel and Infiniband.
- Section 3 investigates the software changes in order to exploit high-speed network hardware. This includes the operating system changes, lightweight communication protocols, and higher layer protocols that provide high-speed communication services to the application level.
- Section 4 discusses (in the point of view of the author's) the possible impacts on Distributed Database Systems due to the broadband and high-speed network infrastructure.
- Section 5 contains conclusion and summary of this report, and proposes interesting problems for future research.

2. Broadband and High-Speed Network

The network subsystems can be categorized into three major areas: the hardware architecture of the host, the host software system, and the network interface. Some of the issues related to the components are:

1. Hardware component: DMA vs. Programmable I/O (PIO) for moving data between host memory to network interface. This will be discussed in section 2.2.
2. Host software: OS, API, high level protocol processes, and the device driver for the network interface. This will be discussed in section 3.
3. Network interface: programmable network interface; high performance protocol support; on-board protocol processing to improve communication performance. This will be discussed in section 2.2 and section 3.

In section 2.1, we first clarify some concepts related to broadband and high-speed networks.

2.1.Introduction to Broadband and High-Speed Network

Network performance is determined by two parameters: bandwidth (a.k.a. throughput) and latency (a.k.a. delay). The bandwidth of a network is defined by the number of bits that can be transmitted over the network in a certain period of time. So if a bandwidth of a network is higher, the time needed to transfer one bit is shorter. The second performance metric, latency, corresponds to how long it takes a message (or a bit) to travel from one end of a network to the other. Latency is measured strictly by time (usually seconds). We often think of latency as having three components:

1) The speed-of-light propagation delay. This delay cannot be changed for any kind of network, although the speed of light may be slightly different in different transmission media.
2) The amount of time it takes to transfer a unit of data. This is a function of the network bandwidth and the size of the packet in which the data is carried.
3) There may be queue delays inside the network, since packet switches generally need to store packets for some time before forwarding them on an outbound link.

Based on the three kinds of time consumption in data transferring in a switched network, we would define the total latency as:

- $T_l = T_p + T_t + T_o$ where T_l stands for latency, T_p stands for propagation time, T_t stands for transmission time, and T_o stands for protocol overhead in transmission.
- $T_p = \frac{d}{c}$ where *d* stands for distance between the source and destination, *c* stands for the speed of light.
- $T_t = \frac{s}{w}$ where *s* stands for size of message, *w* stands for bandwidth.

The bandwidths available on today's networks are increasing at a dramatic rate, but the speed of light does not change. So "high-speed" does not mean that latency improves

at the same rate as bandwidth; the transcontinental "round trip time" (RTT) of a 1-Gbps link is the same 100 ms as it is for a 1-Mbps link. Here, the meaning of "high-speed" is twofold. Firstly, high-speed network is usually broadband network. When you want to transfer large amount of data, the transmission time T_t dominates the latency. So, broader bandwidth networks have less transmission time, hence the total latency is reduced. The second implication of a high-speed network is that its physical media is usually optical fiber, which enables long distance transmission of light without being reinforced by repeaters. This will eliminate significant overhead when the source and destination hosts are far apart.

We can see that latency and bandwidth are two different metrics although they are somehow related. Some applications are latency dominated, while some are bandwidth dominated. For example, interactive programs (such as telnet) are latency dominated, but some programs involving large amount of data transfer are bandwidth dominated, such as ftp. So in Distributed Database Systems, which part will be affected by the high-speed, broadband network is to be determined. Basically, if a DDBMS uses query shipping, the size of data transfer is usually not large, so it should be latency dominated. On the other hand, in most cases data shipping DDBMS will be bandwidth dominated. So for query shipping DDBMS, broadband networks do not help much as to data shipping.

Another thing need to be clarified is that the bandwidth gotten from application level is much less ($\frac{1}{4} \sim \frac{1}{10}$) than that in the "raw" network level. The major cause of this poor performance was the interaction required between the host and the network operations. Optimizations could be done in every level of network subsystems. In section 2.2, we will introduce network adapter and the optimizations on this level. In section 3, software level optimizations are introduced.

2.2.Network Adapter

Another interesting part when it comes to high performance networks is network adapter or network interface card (NIC). A lot of research has been done to reduce the

overhead from the network adapter [12, 13, 14, 15, 17, 18]. A network adapter serves as an interface between the host and the network link, so it is on the lowest level of network subsystem in the operating system. The network adapter can be divided into two parts: the host half and the link half. The host half "talks" to the host via I/O bus, while the link half "talks" to the network link media using particular physical layer and data link layer protocols. Usually there is small amount of fast memory (such as SRAM) on the network adapter to synchronize the I/O bus and network link media since they typically (always) run in different frequencies. The diagram Fig. 1 shows the position of network adapter between host and network link.

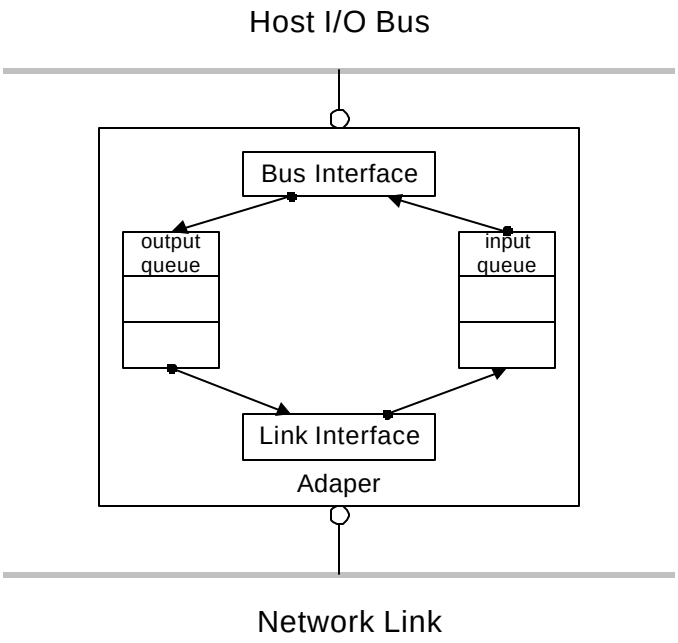

Figure 1 Typical Network Adapter

There are basically two mechanisms to transfer data between adapter and the host memory: direct memory access (DMA) and programmed I/O (PIO). With DMA, the adapter directly read/write the host's memory without any involvement by the host's CPU. The host simply gives the adapter a memory address and the adapter read/write data from/to that memory. There is a less powerful CPU in the adapter, so the work needed to be done by the adapter should be very simple and quick so that no performance penalties will be imposed on the network adapter. With PIO, the host's CPU is directly responsible for transferring data between the adapter and the host's memory. The adapter

must maintain a buffer (greater than or equal to the frame size of the network link) to allow host's CPU to copy to and from. In comparison, the adapter only needs a few bytes of buffer to stage between the I/O bus and network link with DMA. In this sense, DMA is faster than PIO along with its inheriting parallel executing with the host's CPU. But the downside of DMA is that it needs a longer startup time than PIO. So it may be less efficient than PIO when the frame is short.

A typical question facing OS designer is where to allocate a buffer for the data transferred. Typically the data is put in the system's space first and then copied to user's space. This incurs an OS system mode (user mode to system mode) switching and an additional memory copy. Both of these two operations can be very expensive when it comes to high-performance network systems. Thus the user level bandwidth is typically much less than the raw bandwidth of the underlying networks. One solution is to let the adapter directly read and write data in the user's memory. This requires the OS make changes not only in the upper level protocols, but also in the lower network adapter level as well. Detailed information can be seen in section 3 when we introduce Trapeze project.

Another problem is that the bandwidth of I/O bus is typically lower than the network's bandwidth. For example, a typical bus might have a 32-bit-wide data path running at 25 MHz, giving it a peak transfer rate of 800Mbps. But this peak rate tells us almost nothing about the average rate, which may be much lower. Fig. 2 [19] shows the comparisons between different storage media in a modern computer system.

Level	1	2	3	4
Called	Registers	Cache	Main memory	Disk storage
Typical size	<1 KB	<4 MB	<4 GB	>1 GB
Implementation technology	Custom memory with multiple ports, CMOS or BiCMOS	On-chip or off-chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access time (in ns)	2-5	3-10	80-400	5,000,000
Bandwidth (in MB/sec)	4,000-32,000	800-5,000	400-2,000	4-32
Managed by	Compiler	Hardware	Operating System	Operating System/User
Backed by	Cache	Main Memory	Disk	Tape

Table 1 The typical levels in the memory hierarchy of modern computer systems

To match up the network bandwidth, researchers have proposed different solutions. A promising one is to connect the network interface card with system bus along with main memory, not to I/O bus. Thus the network adapter is treated more like main memory. In this way, the bandwidths of network link and bus can match better. [15, 17] discussed this issue in more details.

2.3.Fibre Channel

2.3.1. What is Fibre Channel (FC)?

Fibre channel is a standard developed under the ANSI X3T9.3 task group. It is a computer communications protocol designed to meet the many requirements related to the ever-increasing demand for high performance information transfer. The goals of Fibre Channel include:

- Allowing many well-known existing channels and networking protocols to run over the same physical interface and media.
- High bandwidth (100MB/s and beyond).
- Flexible topologies.
- Connectivity over several kilometers.
- Support for multiple data rates, media types, and connectors.

In general, Fibre Channel attempts to combine the benefits of both channel and network technologies.

A channel is a closed, direct, structured, and predictable mechanism for transmitting data between relatively few entities. Typically, once a channel is set up, there is very little decision making needed, thus allowing for a high speed, hardware intensive environment. Channels are commonly used to connect peripheral devices such as a disk drive, printer, tape drive, etc. Networks, however, are unstructured and unpredictable. Networks are able to automatically adjust to changing environments and can support a larger number of connected nodes. These factors require that much more decision making take place in order to successfully route data from one point to another. Much of this decision-making is done by software, making networks inherently slower than channels.

Fibre Channel tries to combine these two technologies together to construct a flexible I/O architecture based on current high performance communication networks. The high flexibility of its interconnection topology and high data rate (up to or exceeding 1 Gbps) make it a preferable technology for high performance parallel and distributed computing, especially for I/O-intensive computation. To some extents, high performance network technologies make the externalizing of I/O peripherals possible. That is the I/O devices are not connected to I/O bus on the motherboard, rather they are connected by high performance networks and are shared by every machine on the network.

2.3.2. Advantages of Fibre Channel

The features Fibre Channel provides include:

- Unification of networking and I/O channel data communication: all kinds of I/O can be through high-speed network.
- Bandwidth: the Fibre Channel provides more than 1 Gbps network bandwidth, enabling communication between computing devices and I/O devices more efficient.
- Inexpensive implementation: the 8B/10B encoding used by Fibre Channel makes the communication devices inexpensive. Meanwhile, multi-processors based on fast and inexpensive microprocessors and cheap storage disks are less expensive than mainframe. Fibre Channel provides more total power than their mainframe counterparts at a lower price.
- Low overhead: very low of 10^{-12} bit error rate achievable using a combination of hardware and software achievements.
- Local control: local operations depends very little on global information, so it can achieve better robustness. This is a nice property for distributed computing and shared nothing MIMD parallel computing.
- Flexible topology: physical link topologies can be one of the three: 1) point-to-point links. 2) packet-switching network protocols. 3) shared-media loop topologies. The three topologies are shown in Fig. 3. In these three topologies, shared-media loop (also called arbitrated loop) has become the most dominant Fibre Channel topology,

but it is also the most complex. It's a cost-effective way of connecting up to 127 ports in a single network without the need of a Fabric switch.

- Flexible transmission service: multiple classes of services are available: 1) dedicated bandwidth between Port pairs at full hardware capacity. 2) multiplexed transmission with multiple other source or destination Ports, with acknowledgement of reception. 3) best-effort multiplex datagram transmission with acknowledgement. This is suitable for low error rate lower layer networks to achieve more efficient transmission. The details of different classes of service will be discussed later in this section. With these different kinds of services, different applications can find the appropriate service or combine different services to satisfy their requirements.
- Standard protocol mappings: Fibre Channel has very good compatibility with the existing I/O and networking protocols. It provides interfaces to multiple Upper Level Protocols such as IP, Ultra-3 SCSI, IPI-3, HIPPI, ESCON, and AAL5 for ATM.
- Wide industry support: computer vendors, disk driver and adapter manufacturers are providing supports for Fibre Channels. The standardization by ANSI is in progress.

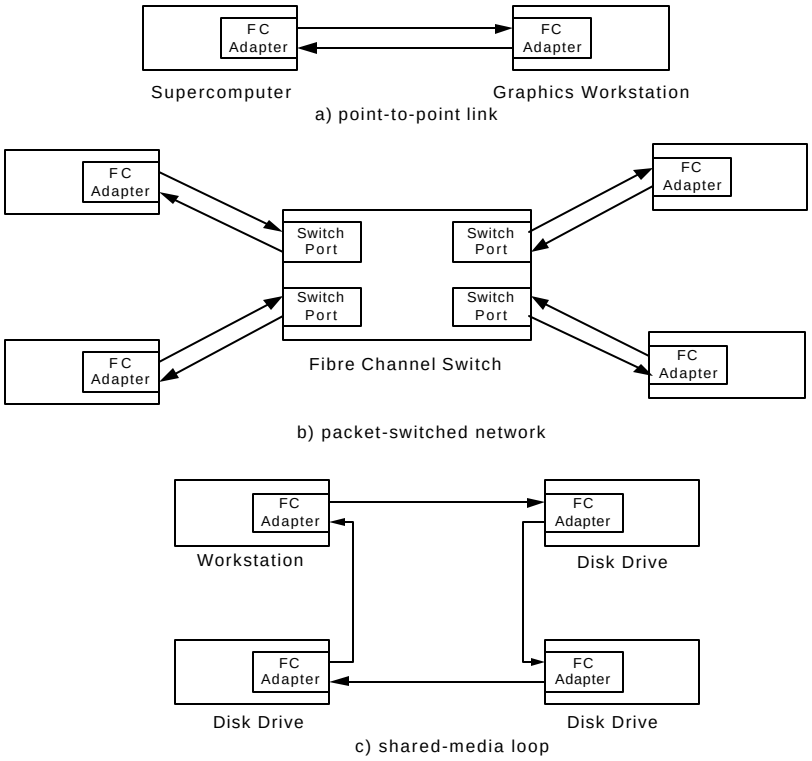

Figure 2 Physical topology for Fibre Channel networks.

One of the most promising features of Fibre Channel is its wide range of bandwidths and compatibility with old transmission media as well as optic fiber. One of the goals of Fibre Channel is to allow HIPPI to map to it. HIPPI is a 100MB/s technology, thus Fibre Channel's primary data rate allows for data to travel 100MB/s. This speed is referred to as full speed. There also exists half speed, quarter speed, and eighth speed. In addition, double and quadruple speeds are defined. The following table illustrates.

Name	Bandwidth (MBps)	Bandwidth (Mbps)
eighth speed	12.5	133
quarter speed	25	266
half speed	50	531
full speed	100	1,063
double speed	200	2,126
quadruple speed	400	4,252

Table 2 Bandwidths of Fibre Channel

Note that the bandwidth in Mbps notation is not 8 times the number of bandwidth in MBps. This is because that Fibre channel uses 8B/10B encoding and the frame overhead and other overheads are factored in the MBps figures.

2.3.3. Fibre Channel Layers

The Fibre Channel standard can be understood easier if it is broken down into layers as it is for networking protocols. Fibre Channel can be divided into five layers:

1. FC-0: its main functions include signaling, media specifications, and receiver/transmitter specifications. It defines the physical media and links with the receivers and transmitters. Single-mode, multi-mode fiber, coaxial cable, and shielded twisted pair are defined as transmission media.

2. FC-1: its main functions include 8B/10B encoding and link maintenance. It describes an 8B/10B transmission code which bounds the maximum run length of a code, maintains DC balance, and provides word alignment.
3. FC-2: its main functions include frame format, sequence management, exchange management, flow control, classes of service, login/logout, topologies, segmentation and reassembly. It defines the signaling protocol which includes the frame structure and byte sequences.
4. FC-3 defines a set of services which are common across multiple ports of a node.
5. FC-4 is the highest level in the standard set. It defines the mapping, between the lower levels of the Fibre Channel and other Upper Level Protocols (ULPs) such as IPI (Intelligent Peripheral Interface), SCSI (Small Computer System Interface) command sets, HIPPI data framing, and IP.

FC-0 and FC-1 can be thought of as defining the physical layer of the OSI model. FC-2 is similar to what other protocols define as a Media Access Control (MAC) layer, which is typically the lower half of the data link layer. FC-3 is not really a layer at all. It is still a largely undefined set of services for devices having more than one port. FC-4 defines how other well-known higher layer protocols are mapped onto and transmitted over Fibre Channel. Thus, one can roughly think of the Fibre Channel layers defining up through the Transport layer of the OSI model.

Three classes of services are offered to users. Class 1 provides dedicated connection service. Data frames are delivered to the destination in the same order they are transmitted by the source. Both class 2 and class 3 services are connectionless services. Class 2 service guarantees notification of delivery or failure to deliver, while class 3 supports unacknowledged delivery. Class 1 service will be used by applications requiring a guaranteed communication bandwidth for a long period of time (e.g., audio or video on-demand applications). Class 2 will be used by applications where multiple transfers are open at one time with frames from the different transfers multiplexed on a single fiber (e.g., the client/server model of distributed computing). The messages transferred by class 2 service are acknowledged like TCP protocol in the Internet. Class 3 is designed to be used for applications like the data link layer of connectionless network protocols such as IP or UDP in the Internet.

Testing in [11] has shown that if the message size is less than 4K, the latency is dominated by the FC transmission time. For class 1 service, when the message size is greater than 4K, the DMA transfer dominate the latency. For example, for a 3 MB message, the DMA operation accounts for 85% of the write latency. The maximum available bandwidth for user-level, DMA and FC phases are 3.61, 4.24, and 25.4 MBps respectively. DMA operation usually contains delays of physically locking the memory pages of the user buffer, preparing the address list, and moving data from main memory to the interface.

For class 2 and 3 services, the maximum transmission size is 128 bytes. Instead of using DMA for moving data, PIO is used in class 2 and 3 services. PIO reduces the latency of transmitting small messages across the I/O bus. This is because the DMA transfer preparation time and page boundary latency are eliminated. In class 2 service, since it requires acknowledgement from the remote receiver after physically transmitting the message, the FC time is much larger than any other factor (initiation, PIO, and completion phases). Class 3 write operation does not have acknowledgement requirement, so its FC transmission time is less than class 2 write operation.

With the processing of standardization of Fibre Channel, more classes of services are coming into place. For the current time being, we are not introducing all the classes of services. Detailed information can be found in the standard specification published by ANSI.

2.4.InfiniBand Architecture (IBA) System Area Network

2.4.1. What is InfiniBand Architecture?

InfiniBand Architecture has many similarities with Fibre Channel in that both of them attempts to externalize the I/O devices onto high performance networks, but InfiniBand Architecture could have even higher bandwidth and more sophisticated architecture. IBA defines a System Area Network (SAN, sometimes it is called Storage Area Network) for connecting multiple I/O devices, I/O platforms and processors. It provides not only data communication among I/O devices and platforms, but only management functions. IBA

SAN encompasses a wider range of I/O platforms than Fibre Channel, managing I/O platforms such as RAID, or Fibre Channel itself. Fig. 3 shows the typical architecture of InfiniBand.

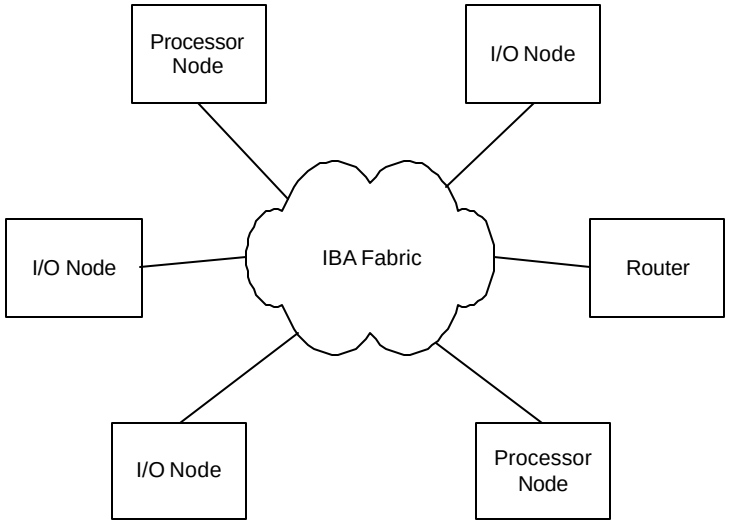

Figure 3 IBA System Area Network

An IBA can be thought of as three kinds of nodes connected by a high performance network called IBA fabric. The three kinds of nodes are processor nodes, I/O nodes and routers. Each processor node can consists of multiple CPU's and shared memories. Similarly, an I/O node can also be as complex as a RAID system or Fibre Channel. The router nodes are used to connect to other subnets. The IBA Fabric consists of switches and routers connected by optic fiber. It can be further divided into IBA subnets connected by router just as the Internet does. So a general case of the IBA System Area Network is that IBA Fabric connects different nodes, which could be processor nodes, I/O nodes or routers connecting to other subnets. IBA subnet consists of end nodes, switches, routers and subnet manager interconnected by links. For different types of nodes, different adapters are used to connect to IBA Fabric. For process nodes, HCA (Host Channel Adapter) is used, while TCA (Target Channel Adapter) is used for I/O devices and I/O platforms.

2.4.2. Advantages of IBA

The value propositions for InfiniBand Architecture are in the following aspects:

- Easy of connect: there is only one fabric connection for all host I/O, which includes Inter-Process Communication (IPC), storage I/O and network I/O.
- Scalability: IBA is easy to scale to thousands of nodes per subnet. Subnets can be connected by routers to construct an even larger packet-switched network.
- Performance: CPU-offload hardware support for message queuing, memory protection and fabric protocol processing.
- Reliability, availability and serviceability: the network can achieve reliability by redundant paths and/or fabrics, in-board management for connectors, baseboard, chassis and power, and error management is handled by layered architecture.
- Flexibility: IBA can use different topologies for different circumstances. Also it can be built on top of optical fiber as well as coaxial copper. Routers can connect IBA network to other types of networks such as the Internet.

In general, InfiniBand Architecture is a scalable hardware platform that can encompass wide varieties of computation platforms and storage platforms. Its basic architecture is very close to the Internet architecture in that nodes (CPU nodes or I/O nodes) are connected by switches to form a subnet, and many subnets can be connected by routers to form a larger IBA network. Depending on the capability of the routers, it can connect to almost all kinds of communication networks including the Interenet. Thus IBA is an open architecture not only for boosting computing and communication power within the IBA network, but also for allowing accesses from non-IBA networks. This feature makes it very appropriate for parallel and distributed computing.

2.4.3. InfiniBand Architecture Layers

As with Fibre Channel, InfiniBand also provides a layered architecture for modularity. Fibre Channel has 5 layers corresponding to the physical layer and data link layer in the ISO/OSI reference model. InfiniBand Architecture provides not only physical

layer and data link layer, but also network layer and transport layer. With these two extra layers, it is easy to deal with the problems such as packet routing, message segmentation and re-ordering, and networks management. So IBA can inherently be more sophisticated than Fibre channel. The five layers in IBA are as follows:

1. Physical layer: defines how the bits are placed on the wire and defines the symbols for framing. The signaling protocol is specified.
2. Link layer: defines packet format and protocols for packet operations. For example, it defines flow control protocols and how packet is routed within a subnet between the source and destination.
3. Network layer: defines protocols for routing packets between subnets.
4. Transport layer: segments messages that are larger than the Maximum Transfer Unit (MTU) in the sender's side and reorganizes the packets into messages in the receiver's side.
5. Upper layer protocols: IBA supports any number of upper layer protocols by various user consumers. This includes certain management functions such as subnet management and subnet services.

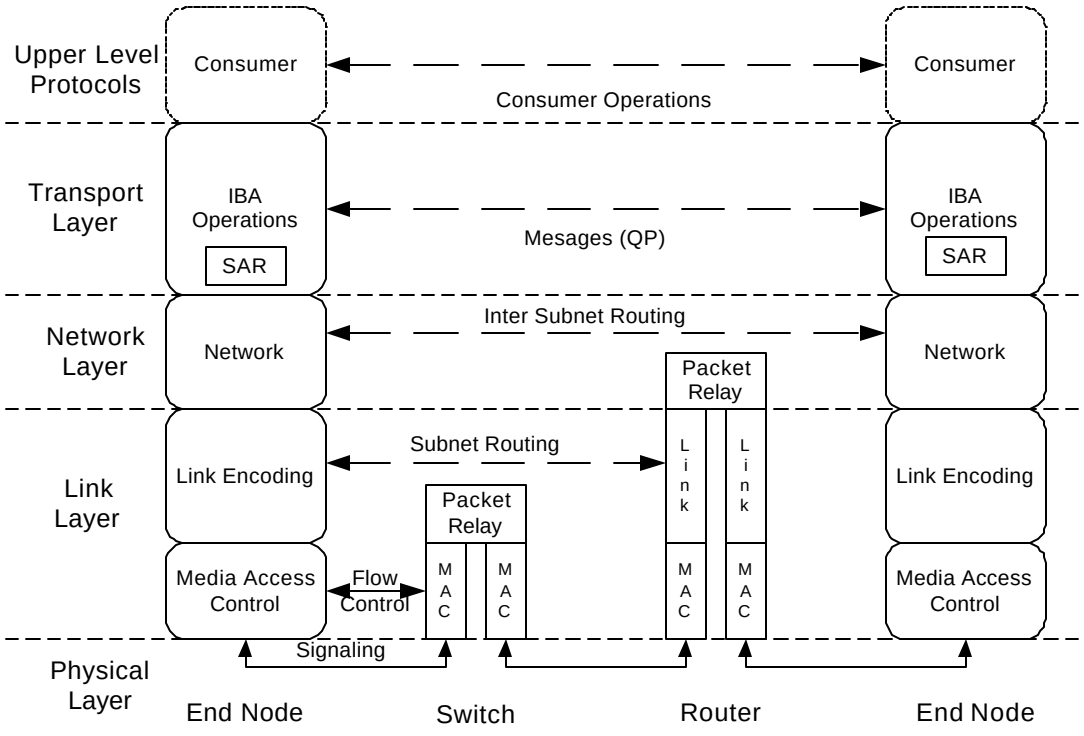

Figure 4 IBA Layers

Fig. 4 shows the five layers in InfiniBand Architecture. Consumer at the highest level is any program or upper level protocols that request network communication to the lower layers. The lower four layers of IBA have the similar functionality as the Internet protocol's physical layer, data link layer, network layer and transport layer in the same hierarchy. One can think of it as a variant of Internet protocol on the high performance network environment, in which interconnecting devices and protocols as well as the nodes are well defined. In fact, the IBA uses IPv6 addressing in the transport layer and also support multicast. The upper layer protocols in IBA are an extension of network management protocols defined in the application layer in the Internet protocols. Actually the upper layer protocols in IBA include the SNMP (Simple Network Management Protocol) protocols defined in the Internet protocols. In addition, it also provides services such as connection management, baseboard management, I/O device management, performance management and others. All of these management protocols ensure that the sophisticated networking architecture performs best on top of high-speed network infrastructure.

At the top level of the protocol layers, a consumer can queue up a set of instructions that the hardware executes. This facility is referred to as a work queue. Work queues are always created in pairs, called Queue Pair (QP), one for send operations and one for receive operations. In general, send queue holds instructions that cause data to be transferred from this consumer's memory to another consumer's memory. And receive work queue holds the instructions about where to put the data received from other consumers. IBA only defines QP for HCA (interface for processor node), but not for TCA (interface for I/O nodes). The QP is the virtual interface that the hardware provides to an IBA consumer and it provides a virtual communication port to the consumer. The IBA supports at most 2^{24} QP's per channel adapter, so a host can simultaneously communicates with at most 2^{24} other hosts. Each QP provides isolation and protection from other QP's and consumers. Thus QP can be considered as a private resource assigned to a consumer.

There are 5 different classes of transportation services provided to the consumer: reliable connection, unreliable connection, reliable datagram, unreliable datagram, and RAW datagram. Their semantics are shown in the following table.

Service Type	Connection Oriented	Acknowledged	Transport
Reliable Connection	Yes	Yes	IBA
Unreliable Connection	Yes	Yes	IBA
Reliable Datagram	No	Yes	IBA
Unreliable Datagram	No	No	IBA
RAW datagram	No	No	Raw

Table 3 IBA Service Type

Each QP is configured for a certain class of service type. Both the source and destination QP must be assigned to the same type of service. Each service type has a certain set of operations available to the consumer, and requires different initialization steps. In addition to the type of services, IBA provides several mechanisms that permit a subnet manager to administrate various quality of service.

Data in different layers are organized in different format and are given different names. In Transport Layer, data are organized into messages. The semantics of messages could be memory oriented such as read/write in Remote Directed Memory Access (RDMA), or channel-oriented operations such as send/receive. When a message is too large to be transferred by the Network Layer, the Network Layer segments the messages into packets in proper size. Packets are end-to-end fabric data unit to transfer and are routable. End-to-end transfer guarantees reliable transport service by acknowledgment and sequencing the packets.

In summary, InfiniBand Architecture is a suitable architecture for high performance I/O applications. It can connect wide varieties of computing platforms and I/O platforms together to form a reliable and flexible system. The Internet-like architecture enables it to exploit current research results of ever growing Internet hardware and software, for example IPv6 and multicast. The embedded network management facilities make possible the IBA grows as large as it is needed. All of them make it a preferable hardware architecture for parallel and distributed I/O-intensive computing. But since it is a very new technology, no much research and experiment have been done on top of InfiniBand Architecture.

3. Software Changes for High-Speed Network Subsystems

There are two ways to improve end-to-end bandwidth:

- using broadband networking technologies such as Fibre Channel and Infiniband.
- modifying the operating system's architecture.

In this section, we will investigate the second way to see what have been done in the operating system's side and the networking protocols

3.1.Changes in Operating System (Tapeze on FreeBSD)

From operating system's point of view, the network subsystem has two possible ways to reduce the overhead imposed by the network interface:

1. Make I/O devices more peripheral: push I/O devices such as hard disks out on the high-speed network, and make them available to every machine on the network. This is the strategy realized by Fibre Channle, InfiniBand Architecture and many other researchers [5, 6].
2. Make high-peed network interface card closer to the CPU: push network interface card up to system bus from I/O bus.

In [17], the authors identified that the key bottleneck for high-speed network is network interface due to the low bandwidth of I/O bus and lack of cache of device registers on the network interface card. Also the current network interface card is designed with an interface similar to a disk's interface. Most current network interfaces require application to use operating system calls. The CPU can only access the on-board registers in an in-order and non-speculative way. All these limitations make network interface card a bottleneck for the high performance network. The solutions that [17] proposed are to:

1. mapping network interface memory to virtual memory so that accessing network interface is the same as accessing memory.
2. connecting network interface to memory bus instead of I/O bus.

3. allowing caching network interface registers, out-of-order and speculative access to these registers.
4. removing some side-effects from the API in the Operating System.

Trapeze project [5, 6] tries to boost the performance of network communication in the network protocols level by using lightweight high-level protocols. In the next section, we introduce lightweight protocols used in Trapeze project.

3.2.Lightweight Network Protocols (lighweight RPC, End-to-End Communication, Active Message, Split-C, Fast Message)

There are basically two types of communication between difference processes:

- Message passing model: processes send messages to one another through communication channels.
- Shared memory model: processes communicate by performing operations on shared data structures called *objects.* Each object has a type associated with it. Each type describes the set of operations that can be performed on an object, and the response that it should return if it is accessed by one operation at a time. The relationship between objects and types are the same relationship between object and class in object-oriented programming languages. For example, register type is responsible to store values that can be read or written by all processes. The serializable problem arises for the objects.

Remote Procedure Call (RPC) belongs to the second kinds of service. It provides user an interface to services provided by remote hosts. Each host can register services it provides to a directory server. When a client requests some service, it sends looking up queries to the directory server. The directory server will return the host name who provides the type of service. The client then sends service requests and parametersto that server. Typically, RPC is built on top of transport layer by using reliable/unreliable connectionless services (for example UDP). New lightweight variants of RPC have been a hot research topic. Trapeze is one of the projects.

Trapeze was designed primarily to support fast kernel-to-kernel messaging alongside conventional TCP/IP networking. Its prototype on TCP/IP is shown in Fig. 5.

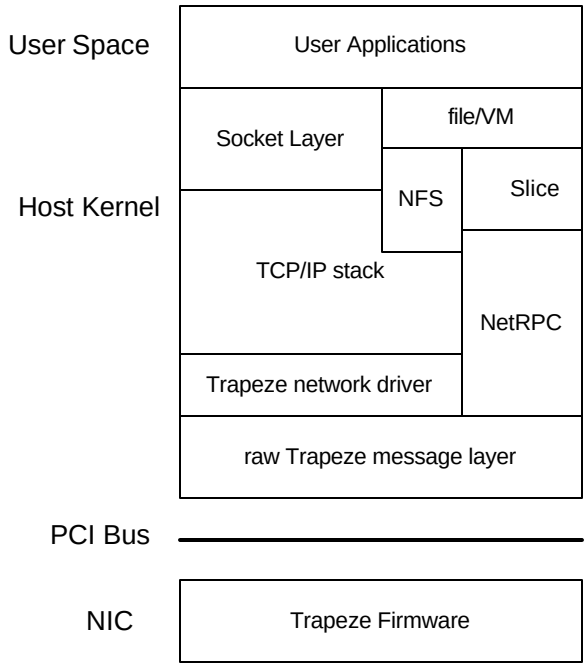

Figure 5 An view of Trapeze prototype

Trapeze is based on Myricom's Myrinet, which is a cost-effective, high-performance, packet-communication and switching technology that is widely used to interconnect clusters of workstations, PCs, servers, or single-board computers. Myrinet provides an interface for programming on the firmware on the network interface card, so that it is extensible for the user to enhance the network interface. Trapeze is based on the custom Myrinet firmware and kernel-to-kernel message layer optimized for block I/O traffic. In the Trapeze Network Interface Card (NIC) firmware, it enables zero-copy block movement, that is, no additional buffers needed in the operating system's space. Data was directly copied to the buffers in user's space.

In the prototype shown in Fig. 5, NetRPC is a lightweight RPC built on top of raw Trapeze message layer. Trapeze messages are short (128 bytes) control messages with optional attached payloads. If there is payload, the host attachés a payload buffer to a message by placing its DMA address in a designated field of the message header. Although the control message and its payload are transferred in one packet, it is separated

in the receiver's side automatically by the firmware. Zero-copy is achieved by demultiplexing the payload from the message using an incoming payload table on the NIC. In addition, the upper layers also have to complement the zero-copy feature. The TCP/IP driver, socket layer and NetRPC share a common pool of aligned network payload buffer allocated from virtual memory page frame pool. Therefore, when you execute a DMA operation, the driver just hands on the address of this virtual memory buffer. The NIC firmware reads from or writes to this buffer. The operating system does not need to copy payloads to other virtual memory buffers. All it needs to do is to mapping the network payload address to the user space address. Zero-copy feature significantly reduces overheads for TCP streams.

In order to get high performance, Trapeze also employs an adaptive message pipelining mechanism to pipeline the DMA transfer on I/O bus and network link. This will minimize the latency of I/O block transfers. Another feature of NetRPC is *non-blocking* RPC, in which the calling thread or process does not have to wait the return of an RPC call. Instead it sets a continuation procedure to be executed when the remote procedure reply arrives. Non-blocking RPC is a simple extension of kernel facilities already in place for asynchronous I/O on disks (select/poll system calls on Unix).

The Slice built on top of NetRPC is a block I/O service. It is called "network storage" in that it is built on a collection of PCs that share disk storage. It is different from traditional file server and System Area Network (SAN), but something "in-between". On top of Slice and NFS, the operating system also provides file or virtual memory (VM) interfaces to the application programs. The application programs can just call a system call (open/read/write) to access the file or mapping some I/O devices to some virtual memory and directly access that memory. Both mechanisms take advantages of the high performance block I/O done by the Trapeze message layer.

In addition to Trapeze, there are many other approaches trying to connect the low-speed Internet with the high performance network [7, 8, 14]. Their common idea is to place a software router between the low-speed and high-speed network. The router switches packets from the low-speed network to the high-speed network by extracting its payload and then adding new headers. Zero-copy can be achieved from the router to the

host on the high-speed network. On the other way, from high-speed network to the low-speed network, the router does the reverse thing.

4. High-Speed Network Impact on Distributed Database Systems

Based on the discussion above, we have seen the major improvements on the hardware architecture and protocols for high-speed networks. It is time to think what they can do for distribute database or parallel database systems. It is obvious that broadband, high-speed network architectures such as Fibre Channel and InfiniBand can be automatically applied to shared-disk parallel database system. But since shared-memory and shared-disk parallel databases are not as promising as shared-nothing parallel databases anymore [23], we are more interested in the impact of broadband high-speed networks on shared-nothing or distributed database systems. When network speed gets faster and bandwidth get higher, distributed database systems can be thought of as a shared-nothing parallel database system in some sense. Technologies used in shared-nothing parallel databases can be automatically used to distributed database systems. In the shared-nothing parallel database systems, one obvious problem is how to place data. In traditional distributed database systems, tables can be partitioned either horizontally or vertically. Once the partitioning is done, little should be changed until the next run of partition. This is based on the assumption that network bandwidth is low. So we had better reduce data transmission and localize the work. However, in the shared-nothing parallel database system, data can be partitioning using many mechanisms. [23] introduced three ways: range partitioning which is the same as horizontal partitioning, round-robin partitioning which is also a kind of horizontal partitioning but is not based on some semantics, and hashing partitioning in which the placement of each tuple is determined by a hashing function. I believe there are many more ways of partitioning tables. Data placement issue should be one of the interests of future research.

Another major technical problem for parallel database system is how to make relational operators run in more parallel. Distributed databases based on high performance networks should also take steps in this direction. There are two types of parallelisms that can be exploited by database systems: pipelined parallelism and partitioned parallelism. The former can be achieved by streaming the output of one operator into the input of another operator such that the two operators can work in series. The latter can be achieved by partitioning the input data among multiple processors and memories. An operator can usually be divided into multiple sub-operators and execute in parallel. Distributed database systems should use some parallelism ideas of parallel databases to exploit the faster network infrastructure.

Although distributed database systems can borrow some ideas from parallel database systems in the environment of high-speed network, the low-speed network is still a possibility for general purposed distributed database systems. Moreover, many difficulties of distributed database systems come from the autonomy nature (or local knowledge) of distributed systems, not the network speed or bandwidth. Distributed algorithm theories for typical database problems (for example, distributed transaction management and query optimization) need more investigation. This should definitely be one of the future research areas in distributed database systems.

With more and more information being put onto the Web, no list of research areas would be complete without mentioning the Web. The new fundamentals here are long response time (but maybe with high bandwidth), autonomy, heterogeneity, and security. For long response time, we need to understand when and how to cache information and how to validate or invalidate it. To cope with autonomy requires cooperative, non-intrusive protocols that Web sites will want to sign up for. Security for distributed systems has long been a ``black hole," and the Web makes its solution more pressing. With the Internet backbone getting faster and faster, a tremendously huge highly distributed heterogeneous database system has been generated. Managing these data are extremely difficult and should be the future research of all database community.

5. Conclusion and Future Works

In this report, I have introduced the hardware and software progress to broadband high-speed network systems. Two major hardware architectures are introduced – Fibre Channel and InfiniBand Architecture. Both of them are trying to push the I/O devices out onto high performance networks. Protocols similar to the Internet protocols are developed for communication within these architectures and connecting them to the outside world. Based on this new hardware architecture, some high level lightweight (low overhead) protocols are also developed to achieve high bandwidth in the user level. Lightweight RPC is such a typical protocol that allows transmitting messages quickly. Operating systems should also be adapted to the high bandwidth to eliminate copying overhead. Zero-copy is one of these mechanisms.

Distributed database systems are very complex system. Figuring out what problems in the distributed systems can be solved and what cannot be solved remain partly open. In those problems that can be solved, what are the efficient algorithms or protocols given high performance communication also remains open. My proposed future research should concentrate on these two problems.

Reference:

1. *The Asilomar Report on Database Research*, Phil Bernstein, Michael Brodie, Stefano Ceri, David DeWitt, Mike Franklin, Hector Garcia-Molina, Jim Gray, Jerry Held, Joe Hellerstein, H. V. Jagadish, Michael Lesk, Dave Maiver, Jeff Naughton, Hamid Pirahesh, Mike Stonebraker, and Jeff Ullman, *ACM SIGMOD Record 27*, 1998.
2. *Fibre Channel, Gigabit Communications and I/O for Computer Networks,* Alan F. Benner, McGraw-Hill 1995.
3. *Infiniband Architecture Specification Volum 1 Release 1.0*, Infiniband Trade Association, October 24, 2000
4. *Infiniband Architecture Specification Volum 2 Release 1.0*, Infiniband Trade Association, October 24, 2000
5. *Network I/O with Trapeze*, Jeffrey S. Chase, Darrell C. Anderson, Andrew J. Gallatin, Alvin R. Lebeck, and Kenneth G. Yocum, *1999 Hot Interconnects Symposium*. August 1999.
6. *Cheating the I/O Bottleneck: Network Storage with Trapeze/Myrinet*, Darrell C. Anderson, Jeffrey S. Chase, Syam Gadde, Andrew J. Gallatin, Kenneth G. Yocum,

and Michael J. Feeley, *Proceedings of the 1998 USENIX Technical Conference, New Orleans, June 1998.*

7. *Interposed Request Routing for Scalable Network Storage*, Darrell Anderson, Jeffrey Chase, and Amin Vahdat. *In the.4th Symposium on Operating System Design and Implementation.*
8. *Payload Caching: High-Speed Data Forwarding for Network Intermediaries*, Ken Yocum and Jeff Chase, *appear in the 2001 USENIX Technical Conference (June 2001)*, December 2000.
9. *The Case of RDMA*, C. Sapuntzakis, A. Romanow, and J. Chase, Internet-Draft, Dec. 2000.
10. *Storage Area Networking*, Michael Peterson, http://www.sresearch.com/wp_9801.htm
11. *Performance of High-Speed Network I/O Subsystems: Case Study of A Fibre Channel Network*, Mengjou Lin, Jenwei Hsieh, David H.C. Du, and James A. MacDonald, Supercomputing'94 (November 1994).
12. *An Analysis of VI Architecture Primitives in Support of Parallel and Distributed Communication*
13. *Design Challenges for High-Performance Network Interfaces,* Andrew A. Chien, Mark D. Hill, Shubhendu S. Mukherjee, *IEEE Computer*, November 1998.
14. *Efficient High-Speed Data Paths for IP Forwarding using Host Based Routers,* S. Walton, A. Hutton, and J. Touch, *In Proceedings of the 9th IEEE Workshop on Local and Metropolitan Area Networks, pages 46--52*, November 1998.
15. *Design Issues for User-Level Network Interface Protocols on Myrinet*, R. A. F. Bhoedjang, T. Ruhl, and H. E. Bal, *IEEE Computer*, 1998.
16. *Efficient, Protected Message Interface in the MIT M-Machine*, Whay Sing Lee, William J. Dally Stephen W. Keckler, Nicholas P. Carter Andrew Chang, *IEEE Computer Special Issue on Design Challenges for High Performance Network Interfaces* , November 1998. pp 69-75.
17. *Making Network Interfaces Less Peripheral*, Shubhendu S. Mukherjee and Mark D. Hill, IEEE Computer, October 1998.
18. *Design Issues for User-Level Network Interface Protocols on Myrinet*, Raoul Bhoedjang Tim Ruhl Henri E. Bal, *IEEE Computer*, 1998.
19. *Computer Architecture A Quantitative Approach, Second Editioin*, David A. Patterson, John L. Hennessy, Morgan Kaufmann, 1996.
20. *DBMSs on a Modern Processor: Where Does Time Go?*, Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, David A. Wood, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.
21. *Readings in Database Systems,* Morgan Kaufmann Series in Data Management Systems, 1998
22. *A Hundred Impossibility Proofs for Distributed Computing*, Nancy Lynch, *In Proceedings of the 8th Annual ACM Symposium on Principles of Distriubted Computing, pp. 1-27, 1989.*
23. *Parallel Database Systems: The Future of High Performance Database Processing*, David J. DeWitt, Jim Gray, *Communication of ACM, Vol. 36, No. 6, June 1992.*