Evaluation of a Cluster-Based System for the OLTP Application

Woo-Jong Hahn, Suk-Han Yoon, Kangwoo Lee, and Michel Dubois

CONTENTS

ABSTRACT

In this paper, we have modeled and evaluated a new parallel processing system called Scalable Parallel computer Architecture based on Xbar (SPAX) for commercial applications. SMP systems are widely used as servers for commercial applications; however, they have very limited scalability. SPAX cost-effectively overcomes the SMP limitation by providing both scalability and application portability. To investigate whether the new architecture satisfies the requirements of commercial applications, we have built a system model and a workload model. The results of the simulation study show that the I/O subsystem becomes the major bottleneck. We found that SPAX can still meet the I/O requirement of the OLTP workload as it supports flexible I/O subsystem. We also investigated what will be the next most important bottleneck in SPAX and how to remove it. We found that the newly developed system network called Xcent-Net will not be a bottleneck in the I/O data path. We also show the optimal configuration that is to be considered for system tuning.

I. INTRODUCTION

Parallel architectures are continuously evolving to fulfill the increasing need for more processing power. Conventionally, parallel computers have been used for scientific and engineering computations. However, there is an increasing interest in applying parallel processing systems to commercial applications, which require more processing power [1], [2].

For large commercial applications, an SMP based on a shared bus is usually adapted for the application server. This architecture has a sharp limit on scalability. On the other hand, Distributed Shared Memory (DSM) architecture still has technical issues to overcome the long latency problem. To provide both scalability and programmability, a message-passing architecture with Shared Virtual Memory (SVM) on top of it can be a more practical solution.

A Network of Workstations (NOW) architecture cost-effectively provides a Message-passing model. It is less expensive than conventional MPPs and supports many commercial applications with minimum rework [3]. Moreover, a NOW naturally provides incremental scalability that is an important requirement of commercial application users who tend to upgrade the system gradually from an inexpensive entry system. As the relatively inexpensive PCs have become competitive with the proprietary workstations in terms of performance, it has been interesting to incorporate PC boards rather than workstations. On this basis, a cluster architecture based on PC boards with a high bandwidth interconnection network is one of the promising architectures in the near future.

Because of the infrastructure based on volumes, Intel's 4-way SMP node, so called Standard High Volume (SHV) node, is being accepted as the most cost-effective way to build a server in the industry. To overcome scalability limitation, one can interconnect the nodes with a communication network and build a clustered system. Unfortunately, it is hard to port conventional commercial applications on the clustered system in which each node has own independent operating system. The emerging micro-kernel technology shows the way to providing a single system image at the operating system level. Each small SMP node runs its own micro-kernel while server processes of the operating system are distributed among nodes. it involves more communication than conventional clustered systems with an integrated-kernel.

ATM is a standard, high-speed network that is suitable for most clustering solutions. However, it has too long latency to be used for such an internal communication channel, by not only having a lower bandwidth than a customized interconnection network but also a protocol overhead. SCI is also a standard and provides higher bandwidth than ATM. It provides an efficient packet network to transfer short mes-

sages that are required by micro-kernel operating system, such as MISIX. However, it lacks capability to transfer large data to or from I/O systems. The distributed I/O nodes of the micro-kernel based system require transferring data which is much larger than a packet. Moreover, SCI involves a significant protocol overhead as it was developed for a cache coherent network. Its ring topology raises the software re-optimization problem since it shows different communication performance depending on the location of the node even in a same cluster.[1]

We have designed a new interconnection network for a micro-kernel based parallel processing system that is based on the commodity SHV design. Based on this new network, so called Xcent-Net, we have developed SPAX for commercial applications, mainly OLTP applications. In this paper, we explore the requirements, specifications and characteristics of the system and the network.

There has been considerable research on scientific parallel processing systems [4], [5]. Parallel processing systems to be used for commercial applications are just emerging and therefore not as much research has been done. Since SPAX is a parallel processing system that targets the commercial application domain, we need to investigate if the system architecture fits the target application.

The I/O subsystem has been considered as the most probable bottleneck for a system running commercial applications. After the I/O subsystem, the interconnection network affects the performance as commercial applications involve huge amount of data movement with minimum data locality. In this study, we evaluate the architecture by investigating the I/O data path to find the bottlenecks and whether they can be removed in the architecture. In addition, we show the balanced configuration to achieve the best performance.

To evaluate the system architecture, we have built the system and a workload model. It is generally accepted that trace-driven simulation shows more accurate results than distribution-driven simulation with a synthetic workload. But the most critical deficiency of trace-driven simulation is inflexibility of the workload. The system configuration from which the traces were collected decisively restricts the target system configuration. Therefore, distribution-driven simulation is used as it provides a relatively simple method to evaluate the system architecture quickly.

We select the TPC-B benchmark [6] as the workload since it can be used to represent the most common core of the OLTP and historically has been one of the most important database applications [7]. Even though the TPC-C is more recent one than the TPC-B, the TPC-B is still a reasonable

[1]Even though there is a SCI switch, it is still implemented on top of the ring topology. For example, Dolphin has a switch based on its SCI link controller, but a packet is switched in the B-Link level, not in the SCI ring itself.

workload for this research, which focuses on investigating the I/O data path bottleneck, not comparing the TPC performance between systems. We show whether the I/O subsystem of SPAX can satisfy the I/O requirements and whether Xcent-Net will become a bottleneck in transferring data for the OLTP workload.

This paper is organized as follows: Section II describes the architecture and design of SPAX. We summarize Xcent-Net, which is the heart of SPAX, in Section III. Section IV describes the system and the workload model. The evaluation results are shown in Section V, which also includes analysis to identify a more balanced, bottleneck-free configuration. We finish with the conclusion and the status of work in Section VI.

II. SPAX

SPAX was developed to serve as a nation-wide information server in Korea. It is the successor of TICOM III, which was also developed at ETRI as an OLTP server. SPAX is designed to overcome the performance and capacity limitations of TICOM III, which is a bus-based SMP system [8]. To be a large OLTP server, the technical requirements of SPAX can be summarized as below.

- Hybrid architecture that can explore the advantages of both SMP and MPP
- Portability and adaptability of commercial applications, such as a parallel DBMS and OLTP, running on a SMP
- Modularity and flexibility in system configuration
- Incremental scalability from a few processors
- Higher bandwidth and lower latency, compared to conventional LANs, in intra-system communication
- Practical packaging which is compact and air-cooled
- High availability

It becomes a hierarchical architecture to provide the characteristics of SMP and MPP at a different level. Each cluster consists of several SMP nodes with shared disks. Because of its SMP node and shared disk, it gives good portability and adaptability for most commercial applications that run on SMP with shared disks. The clustered architecture gives the proper modularity and flexibility in configuring the system. Since it can be partitioned into several clusters, we can provide an air-cooled cabinet that holds two clusters with 192 Gbytes of disks.

The hierarchical architecture is incrementally scalable in nature. Commercial applications show less data parallelism than scientific applications, and most tasks can hardly utilize a large number of processors by itself. So, the users tend to start with the less expensive, smaller configuration. Incremental scalability allows the users to be able to upgrade the system by just adding one more node or disk stream, even a cluster, whenever their work loads or data size

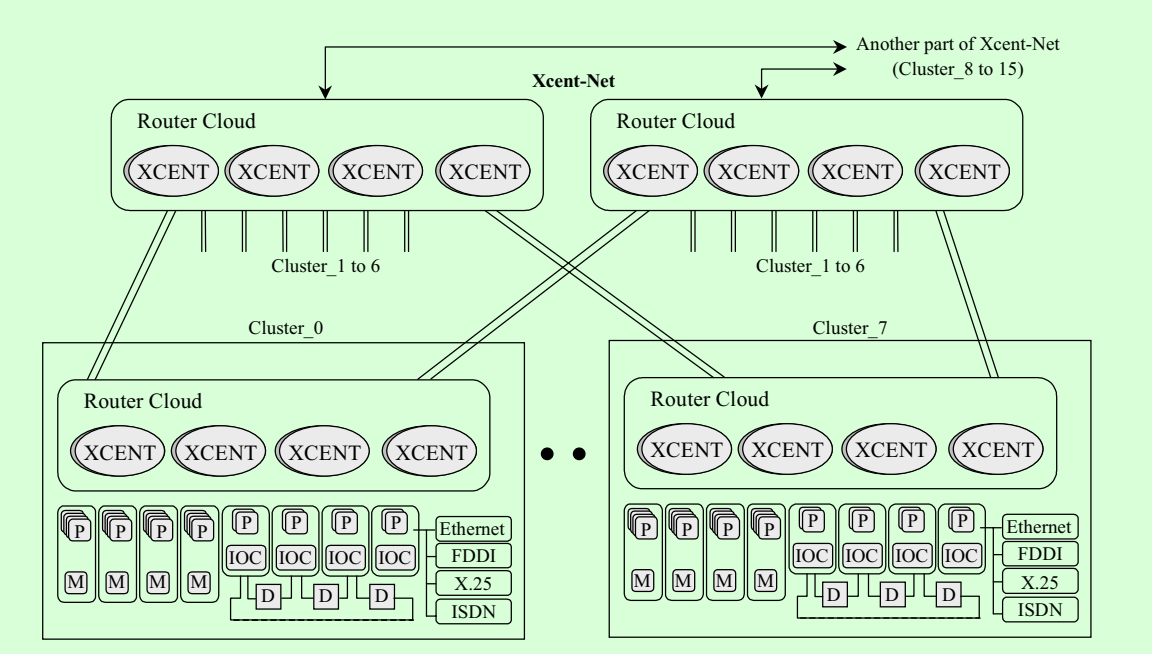

Fig. 1. Architecture of SPAX.

becomes larger.

SPAX consists of many identical components and is very modular. The system is designed to eliminate potential single-point of failures such as loss of a processor, loss of a network, or loss of a disk drive. The processors on an SMP node can be configured as being paired to monitor the other. We also provide a dual path to each disk stream. More importantly, these fault-tolerant features are not mandatory and users can select the non-redundant mode to have more performance. The interconnection network, called Xcent-Net, supports a dual path to every node through the duplicated hierarchical crossbar network. To make the system more cost-effective, both paths of Xcent-Net are active in a fault-free situation. This embedded redundancy provides a reasonable fault-tolerancy with minimum performance degradation [9].

Figure 1 shows the proposed architecture of the parallel processing system for commercial applications.

A maximum configuration consists of 16 clusters and each cluster has up to 8 nodes. Each node can be a quad-processor processing node or a single-processor I/O node. The normal configuration supports up to 256 user processors and 64 I/O processors by having 64 processing nodes and 64 I/O nodes. Since Xcent-Net does not identify the node type, a user can select various configurations by substituting a processing

node with an I/O node or vice versa. The minimum system may contain only one processing node, and users can add nodes one by one to get more processing power or the storage capacity required.

Each node has up to 1 Gbytes of local shared memory. All four processors on a node share the memory but the hardware alone does not support a global address for the memory of other nodes. The operating system provides a Shared Virtual Memory (SVM) to the applications [10]. Two separate PCI buses are provided on each node. One is used to interface Xcent-Net, while the other is for various I/O interfaces. The processing nodes without local disks still have two PCI buses as the I/O nodes. This is mainly because we want to use the same mother board design for all nodes, which increases productivity and maintainability. It also allows SPAX to support a shared-nothing paradigm even though it is designed to easily adapt the shared-disk DBMS, which runs on SMPs.

The heart of the system is Xcent-Net, which is a hierarchical crossbar network based on a 10×10 crossbar router, called Xcent [11]. It is a packet-switched network where each packet carries up to 64-byte data. More details of Xcent-Net will be described in Section III.

The operating system in SPAX is a micro-kernel based UNIX that is being developed at ETRI, called MISIX [10]. Each node has the same copy of the micro-kernel, but may have different servers on top of it. The user nodes have processing servers such as the process manager, inter-process communication manager, or coherency manager, etc. Each user node does not necessarily have all of the processing servers. They can be distributed over the user nodes and reassigned dynamically. I/O nodes have I/O servers which include, the block-I/O manager, disk cache manager, or configuration manager, etc. MISIX provides single system image (SSI) to the user at the operating system level.

III. XCENT-NET

1. Requirements

A central issue in the design of SPAX is the interconnection network. We explore the requirements for Xcent-Net below:

- Reasonable bandwidth allocation for the inter-cluster and intra-cluster communication channels
- Minimum sensitivity to the application software
- Simple and globally uniform protocol
- Ease of packaging
- Network partitionability
- High availability

Considering the bandwidth allocation issue, we do not expect as many communications on the inter-cluster channels as those on the intra-cluster channels, since each cluster has enough processing power

and storage capacity. Moreover, not many commercial applications have enough data parallelism to utilize the whole network at a given time. On the basis of this, we allocate four times more bandwidth on the intra-cluster network than inter-cluster network, which still provides higher bandwidth than ATM or even Fibre-Channel.

Application performance is often sensitive to the topology. For example, HiPi+Bus [8] shows little preference to a specific application but the ring topology does not. This is because the latency is quite different depending on the location of the requester and the responder. As we do not expect to re-optimize the huge number of existing commercial applications, the network latency must be as uniform as possible.

One of the our major design goals is to provide a simple and seamless, global protocol for both inter-cluster and intra-cluster networks. Otherwise we will suffer from a high protocol conversion overhead, which significantly increases the latency, regardless of the peak bandwidth. Moreover, as the system becomes more complex, it becomes harder to test the system in prototyping. When we have a different protocol for the inter-cluster network, we will have even more difficulties to test the system.

Packaging constraint is another important issue, even though it has been overlooked by most academic researchers. It includes board layout, connector pin assignment, and wiring, which limits how fast we can transmit the signals and how wide the channel can be. With given ASIC technology, the number of pins and pads are limited. It is a trade-off between the topology, or the latency of intra-cluster network, and the channel width. Even after this trade-off is made, it is important to assign signals on the pins of ASIC and connectors as it significantly affects the signal integrity and noise. It is again a trade-off between the operational speed and the channel width. Also for maintenance and incremental scalability, it must be easy to rewire the cables.

There are very few commercial applications that have large data parallelism in a single job that can efficiently utilize all the resources of the system. Instead, most commercial applications have many small jobs having their own I/O operations. For these reasons, the interconnection network should provide a natural way to support space sharing, or network partitioning. To have the network partitioned efficiently, we need to be able to distribute the I/O nodes over the entire network.

Also, the commercial applications require reliable environment and we need to remove a single point of failure on the network. A spare network, however, increases cost and therefore we need to provide a more cost-effective method. As described in Section II, our solution is to adapt the embedded redundancy [9].

2. Xcent-Net Design

Based on the afore-mentioned design factors, we have chosen a crossbar topology for the intra-cluster network. Xcent-Net is a dual hierarchical crossbar network since we use the same network for inter-cluster network to avoid the protocol conversion overhead. As we can see in Fig. 1, Xcent-Net consists of router clouds. A router cloud consists of four 8-bit, 10×10 routers, called Xcent, to provide a 32-bit path. Each cluster has dual router clouds to provide high availability. Each router cloud consists of four byte-wide 10×10 router chips, called Xcent chip, to provide a 32-bit channel width for each port. Eight channels of the cloud are for the nodes in the cluster and the remaining two channels build up the inter-cluster network path.

Its design specifications are summarized in Table 1. The detailed description can be found in [11] and are summarized below.

Table 1. Xcent-Net Specification.

Xcent-Net	
Topology	Dual hierarchical crossbar net
Router	10×10 Crossbar router (8:2)
Nodes	128 nodes (16 clusters)
Channel Width	32-bit/channel
Operation Freq.	33.3 MHz (long), 66.6 MHz (short)
Bandwidth (agg.)	34.1 Gbytes/sec at 33.3 MHz
Maximum Distance	33 feet between routers
Fault Tolerancy	Embedded redundancy
	Node isolation
Routing	Virtual Cut-through routing
	Adaptive path control
Clock Scheme	Plesiochronous clocking
	Source synchronous transmission

A. Message

We define two different types of messages which are control messages and data messages. A control message is from 4 bytes to 64 bytes in size and is used to transfer resource or process control information between nodes. A data message is typically as large as a page but can be up to 1 Mbytes. Data messages can increase the traffic on the network since they consist of ten or more packets. According to the preliminary evaluation, Xcent-Net will not be a bottleneck for the typical page size [1].

To efficiently support the two different message types, we support both DMA-based and memory-mapped transfer methods. We reserve the memory space for control messages, which are stored to or retrieved from the dedicated buffer memory in the network interface. Once it is stored in the dedicated memory, the network interface automatically sends the message to the destination. Since it is not efficient to move large data messages to or from the dedicated memory by the processor, we provide a special DMA that partitions the large data in the shared memory into packets and moves them to the network interface, or vise versa.

B. Packet

Xcent-Net is a packet-switched network and the packet is the basic unit for the communication. A packet consists of a minimum 3 flits and up to 21 flits, including 16 information flits. In Xcent-Net, the maximum pay-load of a packet is determined by balancing among the allowable clock skew, latency involved to synchronize a packet, the operation speed, and the distance between the routers. Restricting the packet length also improves fairness in using the channels. Each packet has one or more tags and one control flit that contains information for the network interface, such as a packet sequence number.

C. Routing

Xcent-Net uses virtual cut-through routing to shorten the network latency and to minimize the blocking of other messages when one is passing through the network. The routing scheme is determined by balancing among network efficiency and latency at a given physical environment, such that the neighbor routers can be placed 33 feet away. When the distance between the routers is not short enough, the propagation time of the handshaking signals to transmit a flit in wormhole routing can not be ignored and the total latency increase is not tolerable.

Xcent-Net has source-based routing in which the source node attaches the tag flits to each packet to designate the destination. It is flexible, since the routing policy can be changed by modifying the routing algorithm at the sender interface. To increase the network efficiency, the channels in the upper-hierarchy incorporate adaptive routing. As a result, Xcent-Net implements a source-based adaptive routing.

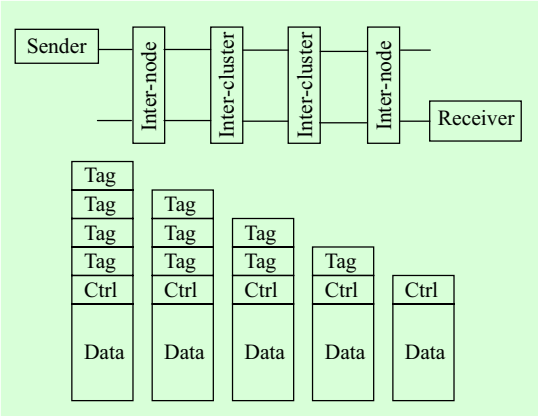

Fig. 2. Use-and-drop scheme in Xcent-Net.

Each packet can have multiple tags, depending on how far away the destination is. The routers use the proper tag and drop it when the packet is transferred to the next router. So the number of tags is determined by the number of router stages the packet has to pass through. Figure 2 shows this use-and-drop scheme for the source-based routing.

D. Flow Control

Xcent-Net uses a clear-to-send flow control. A router or a node confirms the availability of the destination's packet buffer before it sends the packet. When the destination is available, the input packet flows

through the router to the destination even before the entire packet has arrived.

We use a source synchronous transmission method to transfer the flits to the adjacent router. This avoids the skew problem of global clocking and each router cloud operates at its own clock.

E. Interface

The Xcent-Net interface developed has a PCI interface to communicate with the mother board. The PCI interface allows Xcent-Net to be utilized as a high performance interconnection network for virtually every system or boards that have PCI interface. Eight routers forming dual router clouds are installed on a single backplane, which builds up one cluster with nodes.

3. Router

Most of the Xcent-Net functions are implemented in the Xcent router. Xcent consists of separate input and output ports, arbitration logic, 10×10 crossbar, buffer memories, and global control logic. Figure 3 shows the block diagram of Xcent.

The input port synchronizes the incoming packet with the local clock. The incoming packet is sampled with the incoming *Sync* signal and synchronized with the local clock in the packet buffer. We adapt the input buffer scheme for Xcent since it can naturally synchronize the incoming packet with minimum overhead

Ready and *Valid* signals are flow control signals. Xcent drives *Ready* when its input buffer is empty. When it sees *Ready* from the destination, it sends the packet with *Valid*, which notifies the packet on the *Data* lines is valid.

There are two types of arbiters; local arbiter and global arbiter. Each output port has its own local arbiter, and the local arbiter selects the packet to flow through the output port. The global arbiter performs arbitration for broadcasting, which needs all the output ports.

The backplane that holds the router clouds for a cluster is built to be an impedance controlled, 16-layer PCB to minimize signal distortion. We have analyzed and controlled the skew on the backplane signals of the same port to be less than 0.3 ns. We have selected drivers for the backplane and cable after careful analysis using SPICE simulations [12].

IV. SIMULATION MODEL

We have simulated the system architecture to understand the most likely bottleneck in SPAX along with Xcent-Net. The simulation is driven by a synthetic transaction workload [13]. Rather than executing the real codes, each transaction, control and data message transfers required by the transaction are modeled. Although the maximal configuration of SPAX consists of 16 clusters, we only consider the performance of 8 clusters. For this configuration,

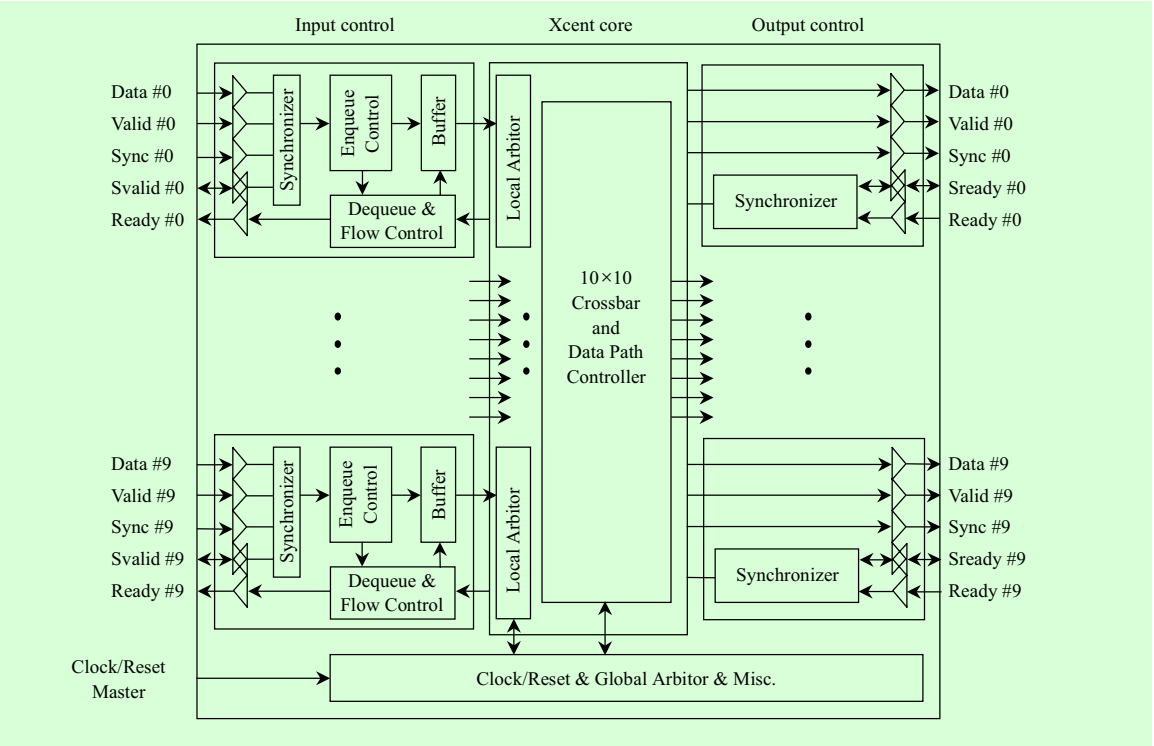

Fig. 3. Block Diagram of Xcent.

the network is a two-level hierarchy. As shown later, the second level of the network does not become a bottleneck because of the locality of workload data. The system is modeled by using the CSIM package [14].

1. System Model

The system model includes disk drives, buffers in the nodes, packet buffers in the router, and DMA. The processing node is abstracted as a message generator. There are three essential constituents in our simulator, which are processes, shared resources, and events.

The typical shared resources in the SPAX simulation model are the processors that run transactions, and disks that hold the data required for the transactions. Another important shared resource is the packet buffers in the network. The packet buffers and the disk drives are the main components to be investigated in this study as they significantly affect the performance of Xcent-Net, and hence, SPAX.

The processes are classified depending on the resources they use. The Teller process is running on the processing node and services the transactions except I/O operations. The I/O process is running on

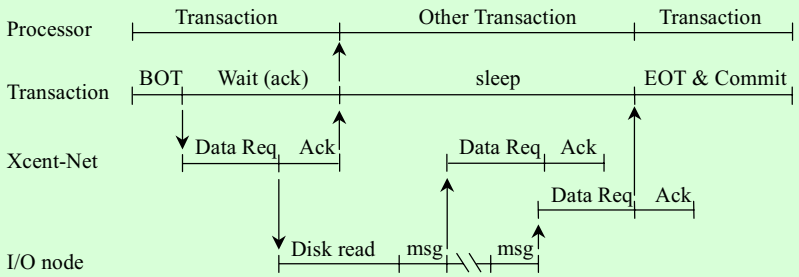

Fig. 4. Communication sequences among the processes.

the I/O node and services disk I/O operations. The Xcent-Net process uses the packet buffer in the Xcent-Net.

Teller processes compete with each other to acquire the processor. Teller processes wait for a processor in a ready queue while another Teller process uses the processor for its transaction processing. When the Teller process requires data from disks, it initiates the Xcent-Net process to communicate with the I/O process and enters into a sleeping queue. Another Teller process in the ready queue uses the processor until the signals come from the I/O process that is servicing the I/O request of the previous Teller process. When the signal arrives, the Teller process in the sleeping queue enters into the ready queue and completes the remaining portion of the transaction when it becomes scheduled. In this simulation model, up to 10 Teller processes run concurrently on a processor. This sequence is shown in Fig. 4 and it repeats until all transactions are committed.

As shown in Fig. 4, when the I/O process on the I/O node finishes reading data from disks, it generates messages to transfer the data. The typical message size is the same as the page size, which is 4 Kbytes in this system. Each message initiates the Xcent-Net process that carries the message to the Teller process.

A Xcent-Net process is created by the Teller process or by the I/O process to carry out the communications between the Teller process and I/O process. The Xcent-Net process is used to transfer the message from the source to the destination through Xcent-Net by allocating the packet buffers in the network interface of the node and the Xcent router. The Xcent-Net process is also initiated for the communications among the Teller processes on the different processing nodes.

2. Communication

Figure 5 shows the three different sequences to transfer messages among processes or processors running the processes. It assumes that the database engine allocates data to utilize multiple processors efficiently and, hence, each processor has its

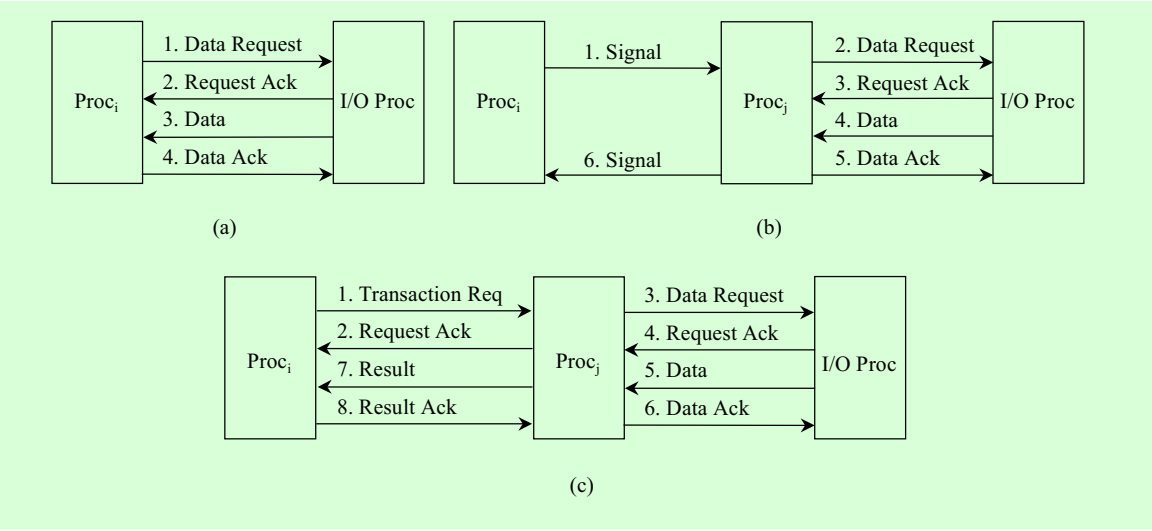

Fig. 5. Message sequences among the processors.

own home I/O node that carries the allocated data. When the data requested is in the home I/O node, the processor generates the data request message and sends it to the home I/O node. When data is ready, the I/O processor sends the data back to the processor by sending one or more messages that carry data. Each message transmission should be confirmed by an acknowledgement packet.

Since the processing node of SPAX has four processors, a home I/O node is shared by the four processors. When the data requested is already held by another processor in the same node, the processor generates a signal for the processor that holds data, which is Proc$_j$ in Fig. 5(b). Proc$_j$ sends a signal for the requesting processor, Proc$_i$, to complete the remaining part of the transaction. Proc$_j$ is responsible to update the database, such as history. No message is involved in communicating between both processors since they share the memory on the node. When the Proc$_j$ is in another node, messages should be used instead of signals. This is depicted in Fig. 5(c).

We assume the model shown in Fig. 5 since it reduces the additional messages to interlock the disk data, which is neccessary to share the disk drives. The operating system and the database engine can dynamically bind the processing nodes and the I/O nodes for the best performance in the runtime. SPAX architecture allows this by providing uniform access to all the I/O nodes in a cluster.

The cases when the transaction requests data from a I/O node is shown in Fig. 6. When data is requested, it is checked if the request can be serviced within a local

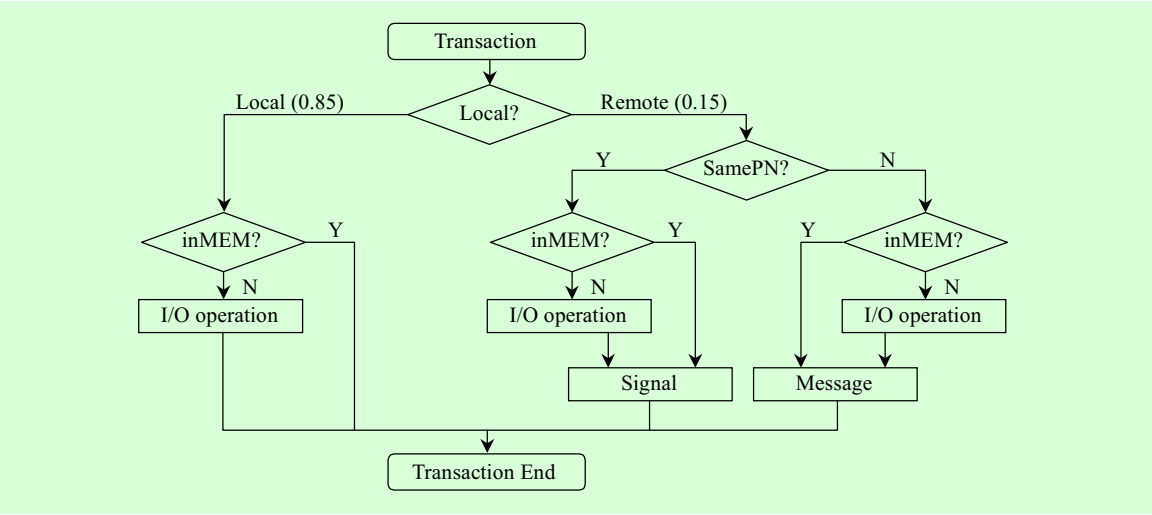

Fig. 6. Cases for disk I/O requests.

database. If not, it generates a remote request. The probability of the remote request is defined in the TPC-B workload specification and is set to 15 %. The remote request is checked if the processor that holds the Account data is in the same node. If it is, no extra message transfer is involved to request and receive data. If not, the processor where the transaction takes place initiates an extra message transfer to the processor that holds data. Finally, the processor that holds data checks if it is already loaded in the main memory. If not, it initiates disk I/O operation. We have evaluated the system for various I/O probabilities.

3. System Parameters

The parameters for the system configuration that we investigated are summarized in Table 2. Most system parameters are taken from our system design specification. Parameters for the disk subsystem are taken from the disk data sheet. The I/O node under simulation has two fast SCSI-II ports but it can be easily expanded to four ports when it is neccessary, as the I/O node provides four PCI slots for I/O devices. Since each fast SCSI-II can hold up to seven nodes, we can easily expand the number of disks up to 28 drives per node. When the disk happens to be a bottleneck, we can estimate whether or not SPAX can accommodate the required number of disks. If SPAX can hold enough number of disks, then we remove the disk bottleneck in our simulation model to find out what becomes the next bottleneck.

We assume only one processor per I/O node since the I/O process is not a CPU-bound process. Even though we can add more processors on a I/O node that is based

Table 2. Parameters used in simulation.

System configuration	
Number of processing nodes in a cluster	4
Number of I/O nodes in a cluster	3
Number of processors in a processing node	4
Number of processors in an I/O node	1
Number of fast SCSI-II ports in an I/O node	2
Network subsystem	
Number of data buffers in a sender	1×4-packet
Number of data buffers in a receiver	1×4-packet
Message passing	
Page size (Data message size)	4,096 bytes
Packet size (Control message size)	64 bytes
Disk subsystem	
Average disk seek time	10 ms
Data transfer rate	40 Mbytes/sec

on the same board as the processing node, it does not affect the I/O performance much.

4. Workload Parameters

We depict the TPC-B benchmark as a workload model, which represents a typical OLTP workload [6]. In our TPC-B model, the ratio between the number of record types that are Account, Teller and Branch is 100,000:10:1. A processor is assigned to a Branch. The logical database design is shown in Fig. 7.

The records of Teller, Branch, Account and History are separately built and managed during the transaction processing.

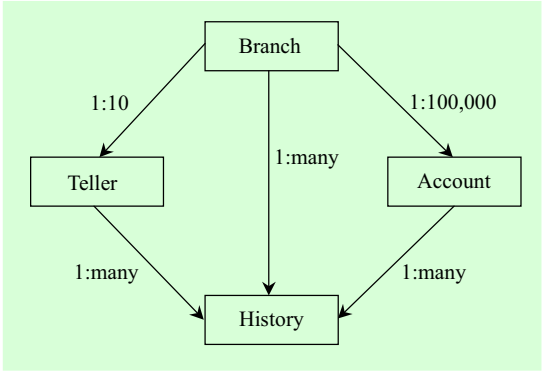

Fig. 7. Logical database design.

Each record for Teller, Branch and Account must be at least 100 bytes. However, in our simulation model, when the Teller process requests the data from the I/O node, it receives a full 4K-byte page instead of a couple of packets that fits for each records. This is because the operating system of SPAX loads the new page when the page containing the requested data is missed in the memory.

We focused on the Account record since it is bigger than other records and is accessed more randomly. We assume that other records such as Teller or Branch are small enough to reside in memory. There is less chance to find the requested Account in the local main memory and it causes significant I/O operations. When the request Account is not found in the main memory, it causes a page fault and load the entire page from the disk.

An average of 15 % of transactions are on the accounts of remote branches. Since each Teller generates transactions, we have

Table 3. Parameters used in workload.

Transaction	
Beginning of transaction (BOT)	25,000 inst. cycles
Reference to account	25,000 inst. cycles
Reference to teller	25,000 inst. cycles
Reference to branch	25,000 inst. cycles
Reference to history	25,000 inst. cycles
End of transaction (EOT)	25,000 inst. cycles
Communication	
Signal preparation	5,000 inst. cycles
Message preparation	5,000 inst. cycles
Initialize remote transaction	5,000 inst. cycles
Interrupt service time	5,000 inst. cycles
Disk operation	
I/O instructions/disk operation	3,000 inst. cycles

10 transaction generators per processor. The simulation is driven by a synthetic transaction workload and the number of instruction cycles to service a transaction is suggested as shown in Table 3 [13]. More parameter that decide the amount of I/O operations and their values are diversely given when we present results. Those are probability that a transaction requires data from disk, probability that the requested data is found in the disk cache.

V. RESULT

Transaction processing is I/O-bound and therefore the I/O subsystem is suspected to be the first bottleneck. Obviously, the higher the disk I/O probability is, the longer the time that the requesting process waits in the sleeping queue. To maintain a fast response time and short average disk queue length, we need to provide large memory so that as many records as possible can be kept in the memory. It is, however, not practical to keep most of ever increasing database in the memory. The other possibility is to increase the number of disks in I/O nodes. We have investigated the effect of the number of disk drives within a cluster, and the results are shown in Figs. 8 and 9.

When the I/O subsystem is heavily loaded, the average queue length of disks is inversely proportional to the number of disk drives. Figure 9 shows that the transaction processing performance is also inversely proportional to the number of disk drives.

When the I/O probability is 20 %, eight disk drives will satisfy the I/O requirement. We need about sixteen disk drives to avoid the disk bottleneck in the case that the I/O probability is 30 % and the disk cache miss rate is higher than 50 %. The I/O node of SPAX can hold up to four fast SCSI-II, and it can easily support sixteen disk drives per I/O node. Figure 9 shows SPAX can avoid the disk bottleneck problem with sixteen disk drives.[2]

The disk cache miss rate affects the performance of the I/O subsystem. The disk

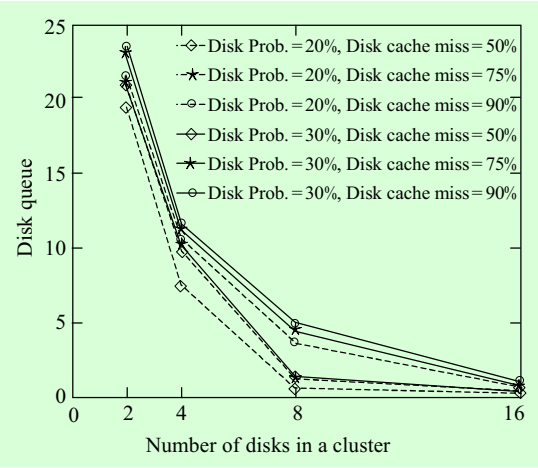

Fig. 8. Effect of the number of disk drives for various miss rates.

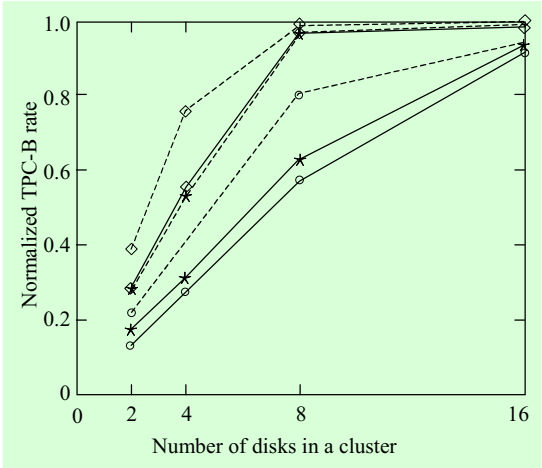

Fig. 9. Normalized TPS rates for various number of disk drives and miss rates (legend is the same as in Fig. 8).

cache miss rate for the database applications is usually higher than what we usually have in scientific applications since the database applications do not have much data locality. Higher disk cache miss rate results in the longer queue in I/O node and disks. However, Figs. 8 and 9 show that even up to 90 % of disk cache miss rate, which is significantly higher than that of the most scientific applications, has little effect on the average queue length and performance when we have sixteen disk drives per I/O node.

The effect of disk cache miss rate on performance is more significant when we have eight disk drives than when we have less than eight drives. This can be explained as follows. When we do not have enough disk drives, a significantly long queue already exists in the disk drives even for low disk cache miss rates. The long queue in the disk drives increases the time that the requesting Teller process waits in the sleeping queue. The more the Teller processes wait in the sleeping queue, the fewer the number of requests to the I/O node. This lower request rate explains why lower disk cache miss rate does not improve the performance much. With eight disk drives, the Teller processes do not have excessive waiting time in the sleeping queue. Having not enough number of requests to the I/O node makes disk cache miss rate affect the queue length less.

Next to the disk drives, data buffers in an I/O node can be a bottleneck. We

[2]All TPC-B results of this study are shown as relative values to the maximum value. In this figure, having 16 disk drives and 20 % I/O rate shows the maximum TPC-B result. The reason of using the relative value is to show the trend, not the absolute value. The absolute value shall be measured in the real system as it is under testing.

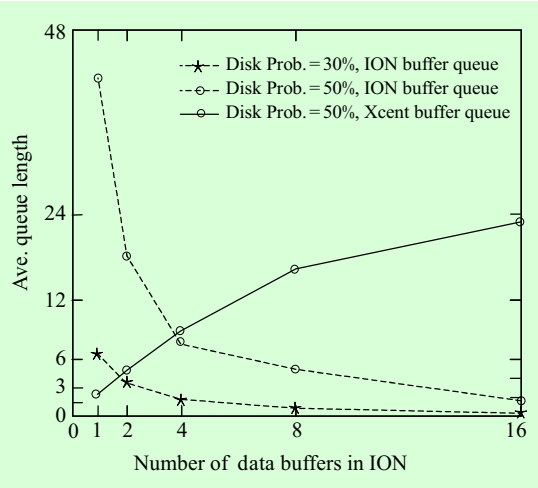

Fig. 10. Effect of the number of data buffers in an I/O node.

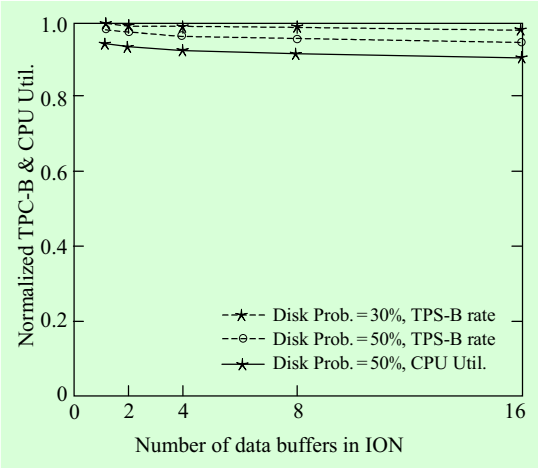

Fig. 11. Normalized TPS rates for various number of data buffers in an I/O node.

have investigated how the number of data buffers in an I/O node affects the performance, provided enough disk drives. As shown in Fig. 10, the average queue length in the I/O node decreases as the number of data buffers increases. When the I/O subsystem is heavily loaded, such as having 50 % of I/O probability, the buffer queue length in the I/O node becomes significantly higher. In this case, the processes in the ready queue of the processing node migrates to the sleeping queue. This reduces the number of I/O requests. Since not all of the transactions have the same number of I/O requests to the I/O node, the transaction performance will vary depending on how many transactions we have in the ready queue. The Xcent buffer will not be the bottleneck unless we have an excessive number of data buffers and disk drives per I/O node.

Figure 11 shows the normalized TPS rates as the number of data buffers in an I/O node varies. Unlike the effect on the average queue length shown in Fig. 10, the change of TPS rates is not significant. Another interesting point in Fig. 11 is that the TPS rates decrease as we increase the number of data buffers.

To analyze why the TPS rates decrease, recall that data from the disk goes through the data buffers in XNIF of the I/O node and the packet buffer of the Xcent router. As we increase the number of data buffers to remove the bottleneck, the packet buffer of the Xcent that is the next component of the path will receive more packets in a given time. This increases the queue length of the packet buffer of Xcent, as shown in Fig. 10. Since the acknowledgement packet which signals that the data request message is received at the I/O node uses the same

packet buffer of Xcent, the long queue of the packet buffer delays the acknowledgement packet to the message sender. The processor utilization falls since the requesting transaction does not release the processor until it receives the acknowledgement packet. This, in turn, becomes the reason for the lower TPS rate.

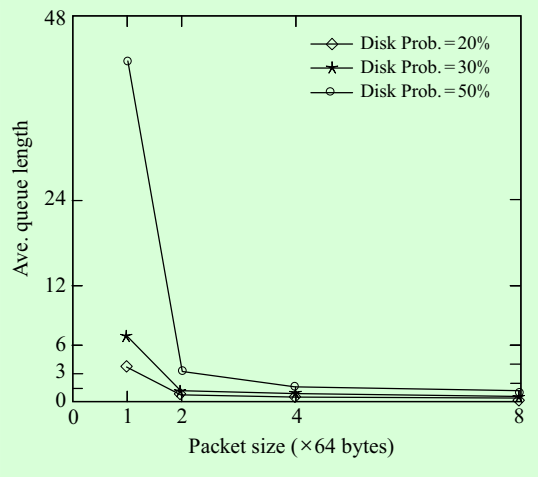

Fig. 12. Effect of the packet size.

Figures 12 and 13 show the effect of the packet size in terms of the average queue length of the data buffer on TPS rates. We investigate the effect for three values of I/O probability, and applied various values of packet size for each I/O probability. As the packet size grows, the average queue length of data buffers shrinks. When the I/O probability moves to 50 % the average queue length of the 64-byte packet increases abruptly. But when the packet size is bigger than 128 bytes, the difference is not outstanding. This shows that the packet size

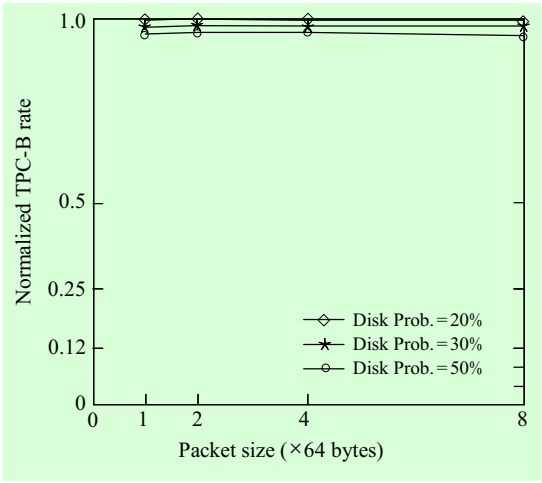

Fig. 13. Normalized TPS rates for various packet sizes.

can play a significant role when the amount of data movement becomes large.

The TPS rate decreases as the I/O probability increases regardless of the packet size. However, the change of TPS rate for a given I/O probability is not significant. Interestingly, for a given I/O probability, the change of TPS rate is not monotonic as the packet size varies. This is because a larger packet holds the path for longer time and it makes other requests wait longer for the packet buffer. Even though a 64-byte packet reveals a significant shift of the average queue length as the I/O probability increases, 64-byte is still acceptable in terms of the TPS rate.

Since the operating system itself carries information for every packet, not all of the 64 bytes will contain message information. Therefore, we also need to consider 128-byte or larger for the packet size. The optimal

packet size must be determined cautiously since the operating system overhead may play a critical role, while the architecture itself is not as sensitive to the packet size from the viewpoint of the TPS rate.

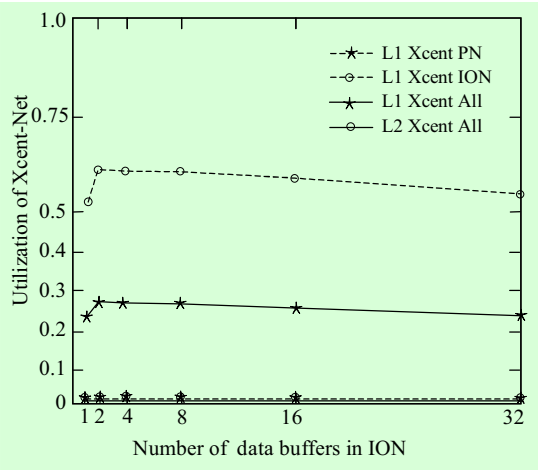

Fig. 14. Xcent-Net utilization for 8 clusters.

We have investigated how Xcent-Net behaves for the various I/O configurations and processor speeds. Figure 14 shows the Xcent-Net utilization for the 8 cluster configuration with enough number of disk drives. As shown in Fig. 14, utilization of Xcent-Net path from the I/O node (ION) is significantly higher than that from the processing node (PN). This is because the processing node can not generate more requests to the network due to the I/O node holding return-messages in its buffer. When there is only one data buffer in the I/O node, the utilization becomes lower than when there are more buffers. This anomaly can be explained by Fig. 10. The average queue length in the I/O node data buffer is so long that the packet buffer in Xcent-Net can not be utilized fully. Adding one more data buffer in the I/O node sharply reduces the queue length and increases the Xcent-Net utilization. The upper-level Xcent-Net, which is L2 in Fig. 14, shows negligible utilization even with enough disk drives and data buffers in the I/O node. It is mainly because the transaction process tends to be allocated to the processing node that is nearest to the I/O node holding data. So the traffic to another cluster is relatively smaller than within the same cluster. Furthermore, the packet buffer in Xcent-Net, holding the message from the I/O node, becomes bottleneck before the upper level Xcent-Net.

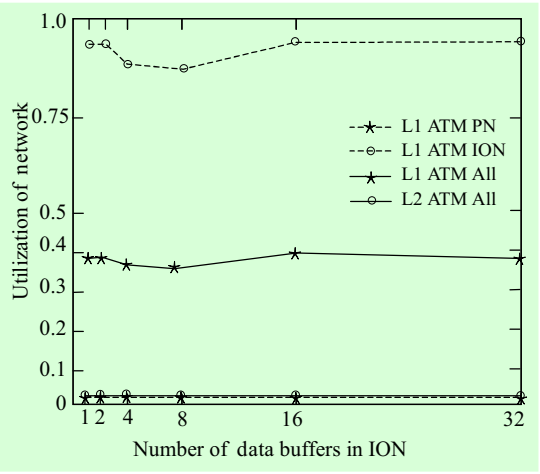

Fig. 15. ATM utilization for 8 clusters.

For comparison, we have replaced the network with a standard 155Mbps ATM network. In Fig. 15, we show the network utilization for ATM. ATM shows significantly high utilization for the I/O side.

Based on this, we find that the ATM network becomes saturated even with a single data buffer in the I/O node. In other words, ATM can be a bottleneck even prior to the data buffer in the I/O node, while Xcent-Net will not. One reason for ATM to show such high utilization is that its bandwidth is lower than required to carry the I/O transactions. Another important reason is that ATM is a connection-oriented network and it needs to set up the path before sending information packet.[3] Other important drawback of ATM is its fixed packet size that prevents it from adapting itself to the requirement of transferring larger data gracefully. Xcent-Net can be easily adapted just by increasing the packet buffer inside of the router.

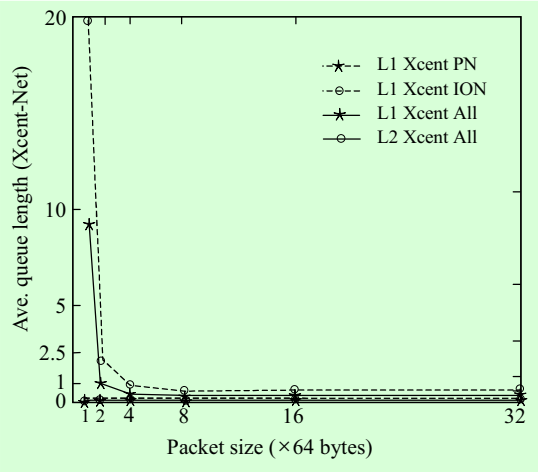

Fig. 16. Effect of the packet size on Xcent-Net (8 clusters).

[3]Even though the ATM standard itself contains the connectionless mode, virtually all of the current implementations are connection-oriented.

As we have found that the Xcent-Net buffer for the I/O node can be the next bottleneck in multi-cluster configuration, we have investigated how the packet size affects Xcent-Net next. Figure 16 shows the effect of the packet size. For the same reason as described above, the upper level Xcent-Net will not be the bottleneck. The queue length of the Xcent packet buffer for the I/O node is much longer than that of the processing node. When the packet size becomes 256 bytes or larger, the difference becomes quite small and both packet buffers result short queue lengths. It matches with Fig. 12, which shows that the data buffers in I/O node will not be the bottleneck for 256-byte or larger packets.

In Figs. 12 and 16, we see that a 256-byte packet yields the most balanced network and can increase the overall performance. However, the operating system itself used to require a significant amount of communication to control processes itself, rather than processing the OLTP application. These communications use short control messages that require fewer than 256 bytes. Based on this analysis, we need to consider supporting another packet size later; current packet size for short messages and about four times larger one for long messages.

As an alternative, we have investigated a network based on 6×6 routers, rather than current 10×10 Xcent router. The 10×10 router fits better for SPAX, which has eight nodes per cluster. But 10×10 Xcent router

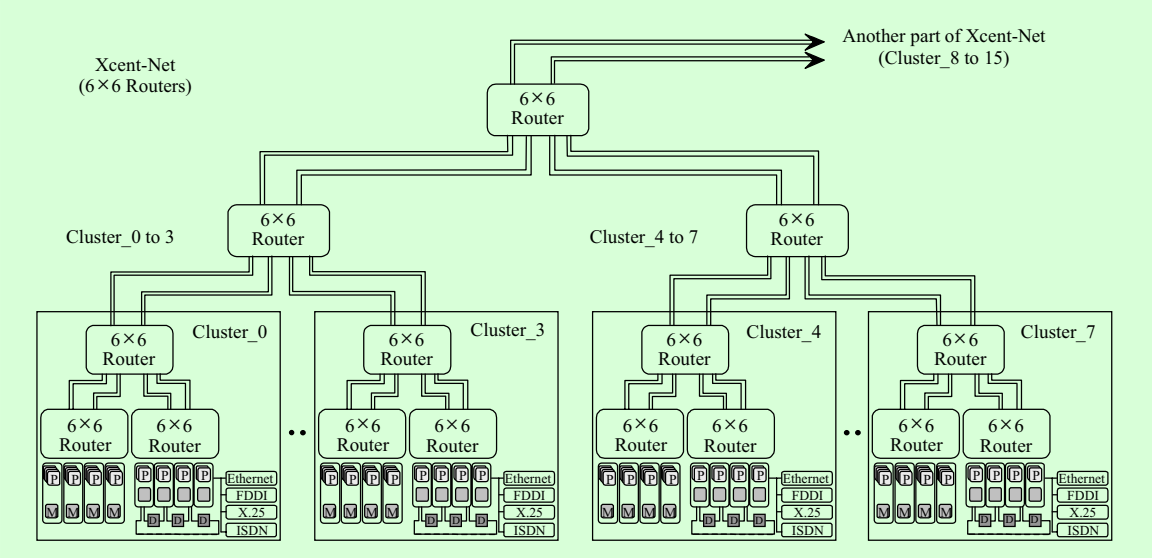

Fig. 17. Clusters connected by 6×6 μXcent routers.

requires a large chip and backplane that result in high manufacturing cost. We have considered a smaller router and built a 6×6 router, called microXcent or μXcent, for the smaller configuration. μXcent uses the same packet protocol and flow control as Xcent. It provides four input and output ports for the nodes and two for connecting to the upper hierarchy. Figure 17 shows the typical architecture using μXcent. It has more levels than Xcent, which involves longer latency even for the intra-cluster communication in a cluster.

In investigating the alternative shown in Fig. 17, we fix the I/O configuration and vary the processor performance, since it is clear that it follows the similar behavior in terms of the data buffers in the I/O node or number of disk drives. By varying the processor performance, we can see how the alternative with longer latency affects the system.[4]

Figure 18 shows the average queue length in the router, μXcent. In all cases, the path to the processing node, which comes from the I/O node, shows longer queue length than that to the I/O node. When there is one data buffer in the I/O node as the current configuration, the difference is more substantial. It means the network is still bounded by I/O. Another reason is that there are only two ports to the second level routers and they are shared by four I/O nodes. We can conclude that

[4]Until now, we have assumed that the processor performance is 150 MIPS. Even though the raw performance of a today's processor is significantly higher than this, it is still challenging to achieve the 150 MIPS for the real performance because of the memory bottleneck.

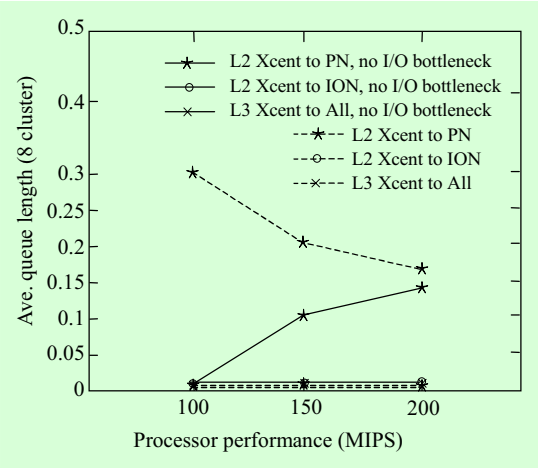

Fig. 18. Queue length of 6 × 6 Xcent-Net (8 clusters).

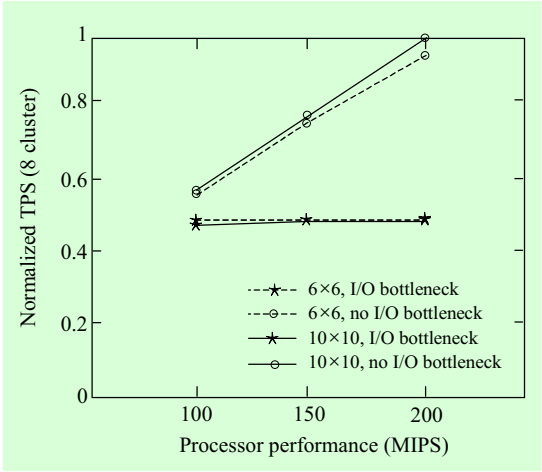

Fig. 19. Normalized TPS rates for various Xcent-Net.

even Xcent-Net based on 6×6 μXcent shall not be the bottleneck before the I/O subsystem. When we eliminate the I/O bottleneck in SPAX by having more disk drives, and data buffers in the I/O node, then we can take advantage of the higher performance processor.[5]

Figure 19 summarizes the effect of the processor performance for the various network configurations. For both networks based on 10×10 and 6×6 routers, it shows very little difference when we upgrade the processors, having a single data buffer in the I/O node. It means it is still I/O-bound. By providing enough data buffers in the I/O node and larger packet size, the system performance scales up along with the faster processor. Moreover, we see the case of 10×10 routers is slightly better than 6×6 cases after eliminating bottlenecks in the I/O path.

[5]As the cache miss rate for the OLTP applications is quite high, the actual MIPS of the processor becomes much less than the operating frequency [15].

VI. SUMMARY

We have designed a new parallel processing system based on a hierarchical crossbar network. We have analyzed the requirements and design alternatives in designing a parallel processing system for commercial applications.

The system supports up to 16 clusters or 128 nodes based on a multiprocessor PC board. Each node can have up to four Pentium Pro processors and two PCI interfaces. I/O nodes can be distributed and mixed with processing nodes in any combination. It makes the system configuration flexible to meet each user's requirement.

The interconnection network of the system, Xcent-Net, is also designed to meet the

requirement of the characteristics of commercial applications. It is a virtual cut-through network and is based on 10×10 crossbar routers. Xcent-Net adapts the source synchronous transmission method with clear-to-send flow control.

We have built a system model to investigate which component might be the bottleneck under the OLTP workload. We applied the TPC-B benchmark model in the course of simulation. The disk drives were found to be the major bottleneck; however, SPAX can provide enough disk drives to avoid the bottleneck. After the disk drives, the data buffer in the I/O node was found to be a bottleneck. The Xcent-Net utilization becomes slightly higher than 50 % for the worst case and far lower for other cases. As a result, we can claim the new system network, Xcent-Net should not be a bottleneck in running a typical commercial application.

As an alternative, we have investigated the system with the smaller router, 6×6 μXcent router. μXcent router is more cost-effective in manufacturing[6] and reveals only marginal performance degradation, which is due to the longer message path. We have also varied the processor performance and found that we have to remove all the bottlenecks in the I/O path to take advantage of higher performance processors.

The system is being developed in ETRI. After finishing the base system, a DBMS engine will be ported to verify the system. After the verification, we will measure the dynamic behaviors and performance on the system. We can also verify the correctness of our system model and workload model. In the measurement, we expect to see the operating system effect and locking effect which are not included in this simulation model.

It is to be used as a nation-wide information network server after commercialization by the cooperative companies.

[6]Xcent is implemented by using 383-pin package, but μXcent requres less than 300-pin, which is significantly cheaper to produce.

REFERENCES

[1] W.J. Hahn, S.H. Yoon, and K.W. Rim, "Design of the New Parallel Processing Architecture for Commercial Applications," *Journal of Korean Institute of Telematics and Electronics*, Vol. 31, No. 5, May 1996, pp. 897–907.

[2] Gartner group, *Second Annual Advanced Technology Groups Conference*, 1992.

[3] T.E. Anderson, D.E. Culler and D.A. Patterson, "A Case for NOW," *IEEE Micro*, Vol. 15, No. 1, Feb. 1995, pp. 54–64.

[4] D.J. Torrellas and D. Koufaty, "Comparing the Performance of the DASH and Cedar Multiprocessor for Scientific Applications," *Proc. of ICPP'94*, Vol. 2, Aug. 1994, pp. 304–308.

[5] L. Barroso and M. Dubois, "Performance Evaluation of the Slotted Ring Multiprocessor," *IEEE TOC*, Vol. 44, No. 7, July 1995, pp. 878–890.

[6] Transaction Processing Council (TPC), *TPC Benchmatk B-Standard Specification Revision 1.1*, 1992.

[7] Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993.

[8] W.J. Hahn, K.W. Rim and S.W. Kim, "A Multiprocessor Server with a New Highly Pipelined Bus," *Proc. of 10th IPPS*, April 1996, pp. 512−517.

[9] B.J. Min, S.H. Shin and K.W. Rim, "Design and Analysis of a Multiprocessor System with Extended Fault Tolerance," *Proc. of IEEE 5th Workshop on Future Trends in Distributed Computing Systems*, August 1995, pp. 301−307.

[10] H.J. Kim, J.K. Lee and K.W. Rim, "Parallel Operating System for MPP System: Design and Implementation," *Lecture Notes in Computer Science*, May 1996, pp. 413−422.

[11] K. Park, J.S. Hahn, S.W. Sim and W.J. Hahn, "Xcent-Net: A Hierarchical Crossbar Network for a Cluster Based Parallel Architecture," *Proc. of 8th PDCS*, October 1996.

[12] S.W. Sim and A. Martinez, "Switching Router Network in a Parallel Computer," *Proc. of Design SuperCon'97*, January 1997.

[13] R. Mark and E. Rahm, "Performance Evaluation of Parallel Transaction Processing in Shared Nothing Database Systems," *Proc. of PARLE*, May 1992, pp. 295−310.

[14] H.D. Schwetman, "CSIM: A C-based Process-oriented Simulation Language," *Proc. of the 1986 Winter Simulation Conference*, December 1986, pp. 387−396.

[15] D. Suggs and R. Reynolds, "Constructing Multiprocessor Workload Characterizations," *Proc. of 33rd ACM Annual Southeast Conference*, March 1995.

Woo-Jong Hahn received the B.S. M.S. and Ph.D. degrees from Korea University in 1981, 1984, and 1995 respectively. He joined ETRI in 1984. Form 1986 to 1988 he was working on 64-bit processor and workstation server development project at AIT in Cupertino, CA., USA. He is a Principal Member of Engineering Staff at ETRI. His main interests are computer architecture with a focus on multiprocessor and parallel processing, memory hierarchy and interconnection network in a large scale system, microprocessor architecture, and multimedia server architecture.

Suk-Han Yoon received the B.E. degree in electronic engineering from Korea University, Seoul, Korea in 1977, the M.S. degree in computer science from KAIST, Taejon, Korea in 1986, and the Ph.D degree in electronic engineering from Korea University, Seoul, Korea in 1995. He joined ETRI in 1977, where he is currently working as Director of Computer System Division and in charge of High-Performance Multimedia Server Development Project. His current research interests include high-performance computer architecture, parallel computing systems and microprocessor architecture.

Kangwoo Lee holds a B.S. degree from Yonsei University in 1985, a M.S. and Ph.D. degree from USC in 1991, 1997, respectively. He is an Assistant Professor in the Department of Computer and Communications Engineering of Dongguk University. His main interests are simulation technique and environment, performance prediction system design, formal language development for behavioral characterization of SMP architectures, multiprocessor system architecture design, and multimedia server system.

Michel Dubois holds an engineering degree from the Faculte Polytechnique de Mons in Belgium, the M.S. degree from the University of Minnesota, and the Ph.D. from Purdue University. He is an Associate Professor in the Department of Electrical Engineering of the USC. Before joining USC, he was a research engineer at the Thomson-CSF in Orsay, France. His main interests are computer architecture and parallel processing, with a focus on multiprocessor architecture, performance, and algorithms. He is a member of the ACM and a senior member of the IEEE Computer Society.