

Impact of Next-Generation I/O Architectures on the Design and Performance of Network Servers

Enrique V. Carrera, Muralidharan Rangarajan, Ricardo Bianchini, and Liviu Iftode

Abstract—

The increasing demand for scalable and highly-available network services has challenged computer architects to develop new I/O architecture for servers. The recently released InfiniBand industry standard, for instance, provides scalable bandwidth and protected I/O communication using the memory-mapped communication model and a channel-based switched fabric technology. Advanced device controllers incorporate processor and memory and can execute sophisticated I/O protocols locally. While accommodating the new I/O architecture within the traditional software framework is straightforward, there is little if any research to explore the server designs that are enabled by the new I/O architecture and their performance impact. This paper uses modeling and simulation to investigate the design-performance space for network servers in the presence of I/O switches, such as InfiniBand, and programmable devices. We propose and analyze a range of scenarios starting from providing conventional servers with high I/O bandwidth, to modifying servers to exploit user-level I/O and direct device-to-device communication, to re-designing the operating system to offload file system and networking functions from the host to programmable devices.

I. INTRODUCTION

The computing landscape has changed dramatically in the last decade. While technology trends in processor speed and memory/storage capacity have been anticipated, traditional computing systems were unprepared to support the increased scalability and availability demands of Internet server applications. The World Wide Web revolution has catapulted networking from an auxiliary, peripheral system to a first-class member of a compute node, blurring the traditional boundaries of networking, architecture, and operating systems. With this revolution came the realization that a network server's demand for I/O bandwidth can no longer be satisfied by conventional hardware and software.

Two recently proposed I/O technologies are particularly meant to remove or alleviate the performance bottlenecks of the conventional I/O architecture, switched-based I/O and intelligent/programmable devices. Perhaps the best example of a switch-based I/O technology is the InfiniBand standard [1], which has recently been proposed by the leaders of the computing industry. InfiniBand provides scalable bandwidth and supports remote memory operations, such as remote read, remote write and remote atomic operations, in addition to standard send/receive communication. To a large extent, the InfiniBand specifications for remote memory operations are inspired by an earlier industry standard for system area networks, called Virtual Interface Architecture (VIA) [2], which in turn was heavily inspired by

the university research in user-level and memory-mapped communication [3], [4].

Programmable device controllers that incorporate a powerful processor and memory and can execute sophisticated I/O protocols have also been studied. In particular, programmable disk controllers have been proposed to offload the host CPU and reduce I/O communication [5], [6], [7]. Programmable network interfaces have been in the marketplace for a while [8], but they have been studied almost solely as support for cluster interconnects in distributed shared memory [4] or distributed file systems [9].

Despite this extensive body of research, two major avenues that can decisively define the architecture of the next generation of high-performance network servers have not been addressed. First, there has been no research to investigate a server architecture including multiple communicating intelligent devices (storage, network, switches), and especially their interaction with novel operating systems designed for such an architecture. Second, there has been no research to address the synergy of using intelligent I/O devices and remote memory communication as supported by the new InfiniBand interconnect, and specifically the performance impact of no longer requiring the host processor to mediate that communication.

In this paper we address both of these avenues. In particular, we investigate the server design-performance space defined by the combination of these two I/O technologies. In what follows, we refer to *cluster of intelligent devices* (CID) to designate a server architecture based on intelligent devices and interconnected by a switch-based I/O fabric that supports remote memory operations. We study the impact of exploiting the new I/O architecture with various degrees of software “awareness” ranging from unmodified conventional software to increasingly sophisticated software modifications. We use modeling and simulations to analyze a range of scenarios starting from providing conventional servers with high I/O bandwidth, to modifying servers to exploit user-level I/O and direct device-to-device communication, to re-designing the operating system to offload file system and networking functions from the host to programmable devices.

We develop a simple model of the performance of applications running on conventional and CID architectures. To make the model and its usefulness more concrete, we focus on an interesting I/O-intensive application, a simple Web server. The model is useful in illustrating the main performance trends. Unfortunately, the model does not include enough detail for us to study interesting aspects of CIDs, such as the ability to perform remote memory writes. To

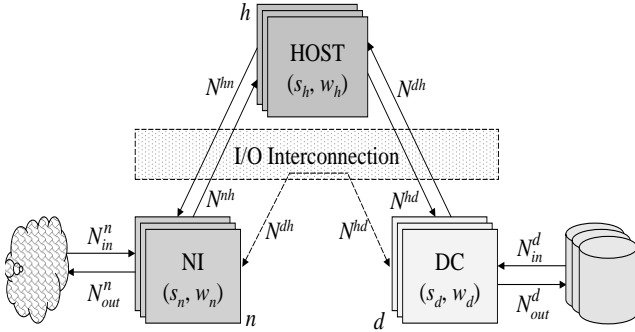


Fig. 1. Components and data flows assumed by the model.

counter this problem, we are also developing a simulation infrastructure to study CIDs and potential software organizations for them. We use our current simulator to study a specific breakdown of the operating system for a CID. In particular, we study scenarios in which the file system and the TCP stack are executed on the host or on intelligent devices for two applications, an SMP-based Web server and an SMP-based video-on-demand server.

The remainder of this paper is organized as follows. The next section presents a model of CID performance for a Web server. Section III presents our most interesting modeling results. Section IV describes our simulation infrastructure. Section V discusses our main simulation results. Finally, section VI draws our conclusions.

II. ANALYTICAL MODEL

A. General Form

Figure 1 presents a pictorial description of the main components of our model, namely the hosts (processor + main memory), the I/O interconnection, the network interfaces (labeled “NI”), the disk controllers (labeled “DC”), and the disks. A system can have one or more of each of these resources. A conventional system only involves computation at the hosts, whereas hosts as well as devices can perform computation in a system with intelligent devices.

The figure also shows the data flows through the system. To simplify the model, we assume that applications are driven by data coming from the network interfaces and/or disks, so that each of the resources operates on streams of data in pipelined fashion with negligible activation and post-processing costs. The basic idea of the model is then to estimate the performance of the system based on the performance of its bottleneck resource. Our model is similar in flavor to that described in [10] for active disks.

Table I lists the model parameters. For each application, N is defined as a different function of N_{in}^n , N_{in}^d , N_{out}^n , and N_{out}^d ; N_h is a different function of N^{nh} , N^{nd} , N^{hn} , and N^{hd} ; N_n is a different function of N_{in}^n and N_{out}^n ; and finally, N_d is a different function of N_{in}^d and N_{out}^d .

In the conventional system, the overall run time is the largest of the individual pipeline stages: host processing time, internal I/O interconnection transfer time, network interface transfer time, and disk transfer time. Thus, the run time is:

TABLE I
MODEL PARAMETERS.

Architectural Parameters	
h	Number of hosts
n	Number of network interfaces
d	Number of disk controllers
s_h	Speed of the host processor
s_n	Speed of network interface processor
s_d	Speed of disk controller processor
Architecture/Application Parameters	
r	Effective I/O interconnection transfer rate
r_n	Effective network interface transfer rate
r_d	Effective disk controller transfer rate
w_h	Cycles per byte at host processor (conventional)
w'_h	Cycles per byte at host processor (active)
w_n	Cycles per byte at network interface processor
w_d	Cycles per byte at disk controller processor
Application Parameters	
N	Number of bytes processed by the system
N_h	Number of bytes processed by host processors
N_n	Number of bytes processed by network interfaces
N_d	Number of bytes processed by disk controllers
N_{in}^n	Number of bytes coming from network
N_{in}^d	Number of bytes coming from disks
N_{out}^n	Number of bytes sent out by network interfaces
N_{out}^d	Number of bytes written to disks
N^{nh}	Number of bytes sent from hosts to disk controllers
N^{hn}	Number of bytes sent from hosts to network interfaces
N^{nd}	Number of bytes sent from network interfaces to hosts
N^{dn}	Number of bytes sent from disk controllers to hosts
N^{hd}	Number of bytes sent from disks to network interfaces
N^{dh}	Number of bytes sent from network interfaces to disks
Performance Measures	
t_c	Run time for the conventional system
t_a	Run time for the system with intelligent devices
T_c	Throughput for the conventional system
T_a	Throughput for the system with intelligent devices

$$t_c = \max\left(\frac{w_h \times N_h}{h \times s_h}, \frac{N_{in}^n + N_{in}^d + N_{out}^n + N_{out}^d}{r}, \frac{\max(N_{in}^n, N_{out}^n), N_{in}^d + N_{out}^d}{n \times r_n}, \frac{N_{in}^d + N_{out}^d}{d \times r_d}\right)$$

Given that in the conventional system the network interfaces and disk controllers do not perform any data filtering, we have $N^{nh} = N_{in}^n$, $N^{hn} = N_{out}^n$, $N^{dh} = N_{in}^d$, and $N^{hd} = N_{out}^d$. The throughput for the conventional system is:

$$T_c = \frac{N}{t_c} = \frac{f(N_{in}^n, N_{in}^d, N_{out}^n, N_{out}^d)}{t_c}$$

In a system with intelligent devices, the overall run time is the largest of similar pipeline stages: host processing time, internal I/O interconnection transfer time, network interface transfer time, on-network interface processing time, disk transfer time, and on-disk processing time. Thus, the run time and throughput for the system with intelligent devices are:

$$t_a = \max\left(\frac{w'_h \times N_h}{h \times s_h}, \frac{N^{nh} + N^{dh} + N^{hn} + N^{hd}}{r}, \dots\right)$$

$$\max\left(\frac{N_{in}^n, N_{out}^n}{n \times r_n}, \frac{w_n \times N_n}{n \times s_n}, \frac{N_{in}^d + N_{out}^d}{d \times r_d}, \frac{w_d \times N_d}{d \times s_d}\right)$$

$$T_a = \frac{N}{t_a} = \frac{f(N_{in}^n, N_{in}^d, N_{out}^n, N_{out}^d)}{t_a}$$

If direct device-to-device communication is used, the second term of t_a (the I/O switch component) should become:

$$\frac{N^{nh} + N^{dh} + N^{hn} + N^{hd} + N^{dn} + N^{nd}}{r}$$

B. Example Application: A Web Server

We illustrate the use of the model with the parameter values of a real I/O-intensive application, a Web server. The parameter values listed in table II were collected from the simulation of a simple Web server subject to client requests for 16-KByte files. The requests were processed one at a time with a 10% memory cache miss rate. The details of the simulator are described in section IV. Given these values, the run time and throughput of the server on a conventional system are:

$$t_c = \max\left(\frac{w_h \times S}{h \times s_h}, \frac{S(1 + miss_rate)}{r}, \frac{S}{n \times r_n}, \frac{S \times miss_rate}{d \times r_d}\right)$$

$$T_c = \min\left(\frac{h \times s_h}{w_h}, \frac{r}{1 + miss_rate}, n \times r_n, \frac{d \times r_d}{miss_rate}\right)$$

The same measures for a system with intelligent devices are:

$$t_a = \max\left(\frac{w'_h \times S}{h \times s_h}, \frac{S(1 + miss_rate)}{r}, \frac{S}{n \times r_n}, \frac{w_n \times S}{n \times s_n}, \frac{S \times miss_rate}{d \times r_d}, \frac{w_d \times S \times miss_rate}{d \times s_d}\right)$$

$$T_a = \min\left(\frac{h \times s_h}{w'_h}, \frac{r}{1 + miss_rate}, n \times r_n, \frac{n \times s_n}{w_n}, \frac{d \times r_d}{miss_rate}, \frac{d \times s_d}{w_d \times miss_rate}\right)$$

For the Web server with intelligent devices, we can actually have direct communication between the disk and the network interface for cache misses. More specifically, the host can inform the disk controller that will service a cache miss where to write the file data, i.e. the correct network interface memory. In this scenario, the two formulas above would not have the *miss_rate* factors in their switch transfer components.

TABLE II

MODEL PARAMETERS AND DEFAULT VALUES FOR WEB SERVER. $S \sim$ NUMBER OF BYTES SERVED. *miss_rate* IS THE FRACTION OF FILE REQUESTS THAT MISS IN THE MEMORY CACHE.

Parameters	Default Values for the Web Server
r	2 GB/sec
r_n	116 MB/sec
r_d	6.7 MB/sec
w_h	$(8.5 + miss_rate \times 1.5)$ cycles/byte
w'_h	0.15 cycles/byte (server + rest of OS)
w_n	3.8 cycles/byte (TCP processing)
w_d	1.5 cycles/byte (file system processing)
N	S
N_h	S
N_n	S
N_d	$miss_rate \times S$
N_{in}^n	~ 0
N_{in}^d	$miss_rate \times S$
N_{out}^n	S
N_{out}^d	0
N^{hd}	0
N^{hn}	S
N^{nh}	~ 0
N^{dh}	$miss_rate \times S$
N^{dn}	0
N^{nd}	0
Under device-to-device communication	
N^{hn}	$(1 - miss_rate) \times S$
N^{dh}	0
N^{dn}	$miss_rate \times S$
N^{nd}	0

TABLE III

ARCHITECTURAL PARAMETERS AND THEIR DEFAULT VALUES.

Parameters	Default Values
h	1
d	4
n	4
s_h	1.5 GHz
s_n	300 MHz
s_d	300 MHz

III. MODELING RESULTS

Before we present our modeling results, it is important to discuss our validation of the model. We compared the model predictions against comparable Web server simulation results. Because the correct values for the w_h , w'_h , w_n , w_d , and r_d parameters can differ for different configurations, we set them to the values we observed in the simulations of section V. We considered conventional and active systems connected by a PCI bus or the InfiniBand interconnect.

Our comparisons show that the model predicts throughput with greater accuracy for low miss rates. For instance, assuming a miss rate of 0%, the modeling results for all system configurations are within 9% of the corresponding simulated results. With high miss rates the model overestimates the throughput of the server by a larger margin. The reason is that the simulated Web server does not have enough threads to hide the latency of the disk subsystem and keep components fully utilized. Assuming a miss

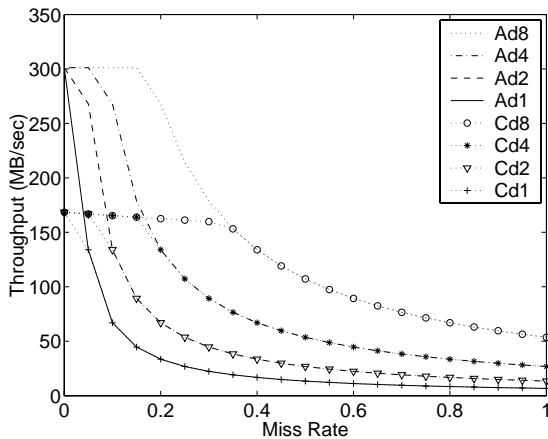


Fig. 2. Throughput as a function of the miss rate for different number of disk controllers.

rate of 10%, for instance, the modeling results overestimate the simulated results by at most 22%. Most importantly though, the model predicts the correct performance trends, regardless of the miss rate.

In the next few paragraphs we use the model to consider a few interesting parts of the parameter space. All results assume a miss rate of 10% and the Web server parameters listed in tables II and III, unless we explicitly mention otherwise. Even though we fix the values of w_h , w'_h , w_n , w_d , and r_d for these results, we do not expect this simplification to qualitatively affect the trends we illustrate.

Disk Controllers. Figure 2 plots the throughput of configurations with different numbers of disk controllers. We consider both conventional (labeled “C”) and active (labeled “A”) versions of the Web server.

This figure shows that an active Web server can perform significantly better than its conventional counterpart for $h = 1$ and $n = 4$, provided that the disk subsystem does not become a bottleneck. In a configuration with a single disk controller, the disk itself becomes a bottleneck at a very low miss rate. With more disk controllers, the bottleneck only appears for higher miss rates, due to the larger aggregate transfer rate of the disk subsystem. For the lower miss rates and larger numbers of disk controllers, the active Web server can achieve a throughput that is a factor of 1.8 as high as that of the conventional server. The reason is that for this range of parameters the host processor is the bottleneck of the conventional servers, whereas the network interface processors are the bottleneck of the active servers. The difference in time per byte spent by these processors is exactly a factor of 1.8.

Figure 3 presents the effect of varying the file system processing requirements when we do not offload the TCP stack to the network interface. Note that we consider a wide range of file system processing requirements; most of these requirements are much higher than the 1.5 cycles/byte that we measured (table II). Our goal here is to illustrate the trends for more computation-intensive file systems, which become feasible with powerful disk controller processors.

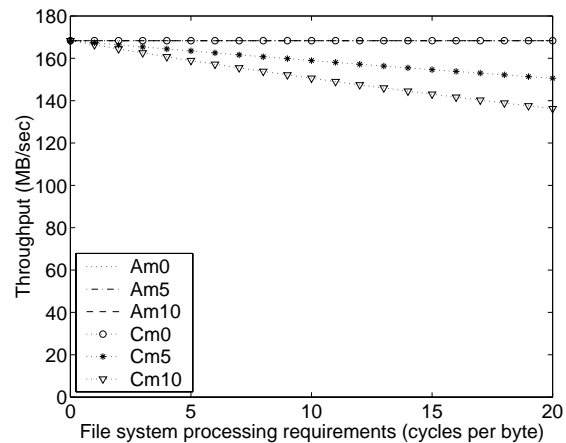


Fig. 3. Throughput as a function of cycles per byte of file system processing for different miss rates (%).

This figure shows that the throughput of the conventional system is degraded as we increase the number of cycles per byte required by the file system. The degradation is proportional to the miss rate, since the bottleneck is the host processor for the whole parameter space considered in the figure. In contrast, the active system is able to retain the same performance regardless of the requirements. Finally, the figure shows that when the processing load imposed on the disk controller is low, less than 5 cycles/byte, the throughput gains of the active system are small, as one would expect.

Network Interfaces. Figure 4 presents the effect of varying the bandwidth of the network interfaces. For low bandwidths, increasing the number of interfaces increases throughput significantly. The throughput of the conventional servers increases up to the 168 MB/sec limit imposed by the host processor. In contrast, the throughput of the active server reaches 268 MB/sec with four or more network interfaces and is limited by the disks.

Another interesting observation from this figure is that the active Web server with a small number of network interfaces (1 or 2) performs worse than the conventional server for network bandwidths higher than about 75 MB/sec. The reason for this result is that the network interface subsystem cannot run the TCP processing with high enough throughput with only a few processors, given our processor speed assumptions. With a few relatively slow processors, it is more appropriate to run lower layers of the communication protocol at the network interfaces. This highlights the importance of breaking the software down carefully in an active system.

Figure 5 presents the effect of varying the TCP processing requirements. We can see that a single 300 MHz processor is able to handle up to about 2.5 cycles/byte for a Gigabit Ethernet network interface. After that point, the active system starts to perform worse than its conventional counterpart, as the network interface processor becomes a bottleneck. A similar effect is observed for two 300 MHz processors. However, the use of more network interfaces al-

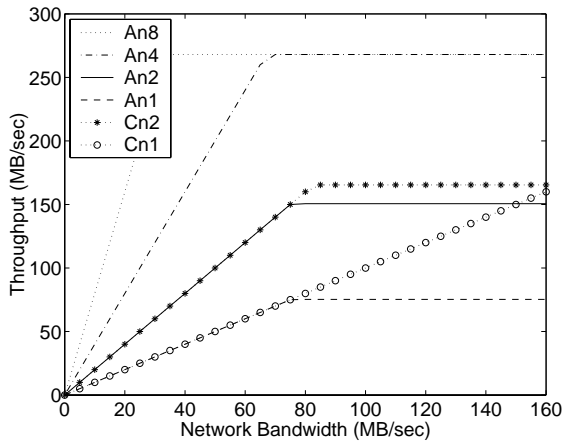


Fig. 4. Throughput as a function of the network bandwidth for different numbers of NIs.

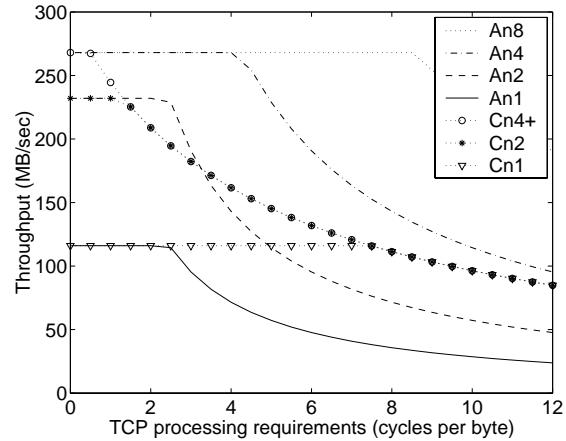


Fig. 5. Throughput as a function of cycles per byte of TCP processing requirements for different numbers of NIs.

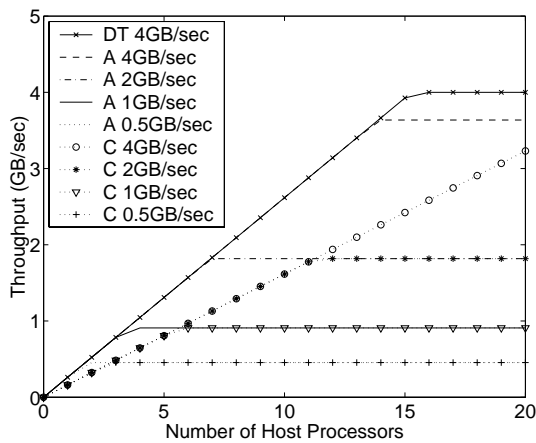


Fig. 6. Throughput as a function of number of host processors for different switch bandwidths.

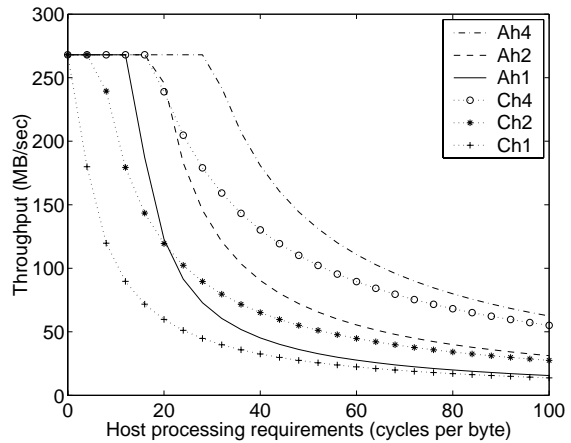


Fig. 7. Throughput as a function of cycles per byte of host processing requirements for different number of processors.

lows the active system to significantly outperform the conventional system. The maximum gain strongly depends on the number of devices and the particular TCP processing requirements.

Note that more than four network interfaces for the conventional server do not improve the throughput further. The reason is that this number of interfaces is enough to make the disk or the host processor the performance bottleneck.

Switch. Figure 6 presents the effect of varying the switch bisection bandwidth on the throughput of the conventional, active, and direct (disk to network interface) transfer (“DT”) systems. As we can see in the figure, four processors are enough to saturate an active system based on the PCI-X bus (1 GB/sec), while we need six processors to saturate a conventional system based on the same bus. More generally, we can see that a conventional system requires a larger number of host processors in order to get the same performance of an active system, especially for the higher bandwidths. The reason is that in the former type of system the disk controller and network interface processors do not participate in the execution of the ap-

plication or operating system. Note also that, at a large number of hosts, the active system saturates at a throughput that is about 10% lower than the active system with direct disk-to-network interface data transfers.

Host. Figure 7 shows the effect of the (non-TCP or file system-related) host processing requirements as we vary the number of host processors. These results illustrate the effect of requests for CPU-intensive dynamic content. The figure shows that the disk subsystem is the bottleneck for low requirements. As the requirements are increased, the host processor quickly becomes the bottleneck for the conventional systems. The active systems can deal with significantly higher requirements without degradation because the TCP and file system processing is offloaded to the corresponding devices. The performance advantage of the active systems sharply decreases however, as we move to very high processor requirements.

In summary, our modeling results illustrate the performance benefits of having greater I/O bandwidth, multiple intelligent devices, direct device-to-device communication, and splitting the operating system between the processors of a CID. More importantly, the results illustrate the need

TABLE IV
MAIN PARAMETERS AND THEIR DEFAULT VALUES. † ESTIMATED.

Parameter	Default Value
Number of host processors	16
Clock frequency of the host CPU	1.5 GHz
First-level cache size	32 KB
Second-level cache size	128 KB
Main memory size	128 MB
Number of memory banks	8
Memory bus frequency	133 MHz
Number of disk controllers	16
Disk controller cache size	128 KB
Disk seek time	2 millisecs + 20 microseconds/track
Raw disk transfer rate	40 MB/sec
Number of NIs	16
Raw NI transfer rate	128 MB/sec
Raw PCI transfer rate	512 MB/sec
RapidIO 4-byte message latency	250 nanosecs†
RapidIO transfer rate	2 GB/sec
InfiniBand 4-byte message latency	4 microsecs†
InfiniBand transfer rate	2 GB/sec

for careful co-design of the software and hardware to avoid unwanted bottlenecks.

IV. SIMULATION METHODOLOGY

Simulator. In order to evaluate the behavior of real applications running on CIDs, we are developing a detailed event-driven simulator. The simulator is based on the MINT [11] front-end, and allows us to simulate a cluster of SMPs connected to several I/O devices.

We simulate the behavior of an N -way SMP, where each processor has a coalescing write-buffer and a first-level cache. Second-level caches are external to processors and implement a three-phase write-back cache coherence protocol. The memory hierarchy is completed with a memory controller and M independent memory banks. The processors and caches are connected to the memory through a bus (the memory bus). An I/O bridge is also plugged into the memory bus. The function of the I/O bridge is to serve as an interface between the memory bus and the I/O interconnection. Several disk controllers and/or network interfaces can be plugged into the I/O interconnection.

Our simulator is able to implement several architectures for the I/O interconnection. For this paper we evaluate an I/O bus (PCI), and two switch-based I/O interconnects (RapidIO and InfiniBand). RapidIO [12] and InfiniBand [1] are communication standards based on a packet-switched interconnection. RapidIO is a flexible architecture internal to high-end servers with high bandwidth, low latency and dedicated hardware interfaces between chips. InfiniBand is focused on distributed systems allowing the communication between server hosts and external resources such as I/O devices. In terms of bandwidth, the standards are comparable. However, the per packet latency of InfiniBand should be higher than that of RapidIO, because the latter standard assumes chip-to-chip communication. Nevertheless, InfiniBand allows larger packets than RapidIO, mitigating the higher latency problem.

Contention for buses and memories is simulated in detail for all I/O architectures. However, contention for the I/O interconnection is only simulated at the end-points. Given the high interconnection bandwidths we are considering, this assumption should not cause relevant inaccuracies. The main architectural parameters used for each one of these implementations are listed in table IV.

The simulator also allows us to run several application threads or processes on each processor. The threads or processes share all the resources of the SMP. In addition, the simulator allows us to have a programmable processor in each I/O device. The architecture of the each I/O device is similar to the architecture of a host/memory pair.

In order to simulate the operating system functionality, we have written user-level libraries that implement the file system and the TCP stack. The TCP library is currently very simple, simulating data copying and checksum computation only. The active systems we simulate split the operating system in the following way. Each active disk runs a copy of the file system, but any data caching is implemented by the server (user-space) running on the host. Each network interface executes the simplified TCP stack. A message send involves the application sending a message to the network interface. The message is transferred directly from user-space to a buffer at the interface. A message receive is first buffered at the network interface and then copied directly to user-space at the host. All intra-server communication in the active systems is performed with remote memory reads and writes.

Applications. We use two server applications. The first application is a Web server that creates several threads. Each thread listens to a socket for requests. After receiving a request, each thread parses and serves the request from disk or from its internal file cache. The workload of the Web server consists of requests for 16-KByte files. The memory cache miss rate is 10%.

The second application is a video-on-demand (VoD) server that also uses several threads for servicing requests, but it does not use any internal caching mechanism. The workload of the VoD server consists of requests for very large files without temporal locality. Thus, caching is not useful in this case; all requests access the disk. For both types of servers, the clients are real processes, not simulations.

V. SIMULATION RESULTS

Figure 8 shows the throughput of a Web server for three different I/O interconnections. For each I/O interconnection, the figure differentiates the performance of a conventional and three active servers. Each active system assumes a different device processor speed; the second bar from the left is for 100MHz device processors, the third bar is from 300MHz device processors, and the rightmost bar is for 1.5GHz device processors. These bars correspond to ratios of 1:15, 1:5, and 1:1 for the device/host processor speeds. All active servers assume that the TCP stack runs on the network interfaces and the file system runs on the disk controllers.

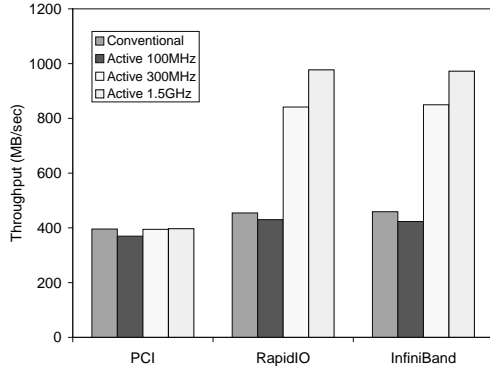


Fig. 8. Throughput of our Web server.

We can see that for this application, the I/O bus-based system performs worse than those systems that are based on a switched I/O architecture. The reason for this result is the bottleneck created by the bus; most of the I/O time is wasted in contention for the I/O bus. If we use a switched I/O interconnection (RapidIO or InfiniBand), the performance of the conventional servers improves by 15%. The active servers improve even more when fast device processors are used. The improvements for the active servers can be as large as 146% for RapidIO and InfiniBand when the I/O devices and the host all have 1.5GHz processors.

Under switched I/O the host processors become the major performance limitation. Thus, the active servers perform better than the conventional ones when the I/O interconnection is not the bottleneck, provided that the device processors are fast enough. For both switches, the difference between the conventional and the active servers with 300MHz and 1.5GHz processors is about 85% and 112%, respectively. These performance improvements are basically due to the offloading of the host processors, the elimination of the data copy (from user to kernel space) made at the host, and the corresponding reduction in memory bus contention. In the active servers, this copy is not necessary because the application transfers the data to the network interface directly from user space.

Figure 9 shows the throughput of a VoD server for the same three different I/O interconnections. For each I/O interconnection, the figure also differentiates the performance of a conventional and the same three active servers.

We can see in this figure that the throughput of all server-architecture combinations is low. The reason is that the performance of this application is limited by the disk access time. Processors in the conventional server on a PCI bus spend more than 90% of their time waiting for disk accesses to complete. The reason is that disk access requests coming from different processors are interleaved by the bus arbitration, thereby reducing the locality of accesses. Moving to an active server or a switched I/O architecture alleviates this problem. The active servers alleviate the problem because prefetching is performed by the disk controller itself, guaranteeing an ordering of accesses with good locality. The throughput of the active servers is 47% higher than the conventional server under the PCI

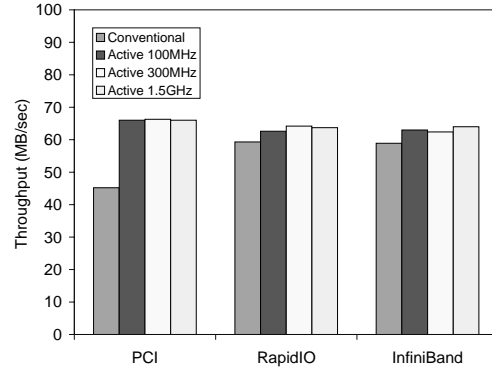


Fig. 9. Throughput of our VoD server.

bus. The switched I/O architectures alleviate the problem because the requests are not serialized by the I/O interconnect. Using either RapidIO or InfiniBand increases the throughput of the conventional server by 31%.

In summary, these simulation results reinforce our modeling observations, which highlight the importance of having enough I/O bandwidth, programmable devices, and split operating systems. In addition, the simulation results demonstrate that there are no significant performance differences between RapidIO and InfiniBand for the applications we considered. We are currently in the process of extending our infrastructure to simulate device-to-device communication, more sophisticated disk scheduling, and split applications.

VI. CONCLUSIONS

In this paper we investigated the design-performance space for next-generation network servers, which we envision will be built as clusters of intelligent devices with remote memory communication. From our preliminary evaluation of several hardware-software architectures we conclude that (i) intelligent devices amplify the performance benefit of switch-based I/O, and (ii) the new I/O architecture for servers requires a new system software architecture (Split-OS [13]) in order to fully reveal its potential.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

- [1] “The infiniband trade association,” <http://www.infinibandta.org>, August 2000.
- [2] Dave Dunning, Greg Regnier, Gary McAlpine, Don Cameron, Bill Shubert, Frank Berry, Anne Marie Merritt, Ed Gronke, and Chris Dodd, “The Virtual Interface Architecture,” *IEEE Micro*, vol. 18, no. 2, 1998.
- [3] Anindya Basu, Vineet Buch, Werner Vogels, and Thorsten von Eicken, “U-net: A user-level network interface for parallel and distributed computing,” in *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995.
- [4] Edward W. Felten, Richard D. Alpert, Angelos Bilas, Matthias A. Blumrich, Douglas W. Clark, Stefanos Damianakis, Cezary Dubnicki, Liviu Iftode, and Kai Li, “Early experience with message-passing on the shrimp multicomputer,” in *Proceedings of the 23rd Annual Symposium on Computer Architecture*, May 1996.

- [5] Garth A. Gibson, David F. Nagle, Khalil Amiri, Fay W. Chang, Howard Gobioff, Charles Hardin, Erik Riedel, David Rochberg, and Jim Zelenka, "A cost-effective, high-bandwidth storage architecture," in *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems*, October 1998.
- [6] A. Brown, D. Oppenheimer, K. Keeton, R. Thomas, J. Kubiatowicz, and D.A. Patterson, "Istore: Introspective storage for data-intensive network services.," in *Proceedings of the 7th Workshop on Hot Topics in Operating Systems (HotOS-VII)*, March 1999.
- [7] Anurag Acharya, Mustafa Uysal, and Joel H. Saltz, "Active disks: Programming model, algorithms and evaluation," in *Architectural Support for Programming Languages and Operating Systems*, 1998, pp. 81–91.
- [8] "Myricom: Creators of myrinet," <http://www.myri.com>.
- [9] Darrell C. Anderson, Jeffrey S. Chase, Syam Gadde, Andrew J. Gallatin, Kenneth G. Yocum, and Michael J. Feeley, "Cheating the I/O bottleneck: Network storage with Trapeze/Myrinet," in *Proceedings of the 1998 USENIX Technical Conference*, June 1998, pp. 143–154.
- [10] Erik Riedel, Garth A. Gibson, and Christos Faloutsos, "Active storage for large-scale data mining and multimedia," in *Proceedings of the 24th International Conference on Very Large Data Bases, VLDB*, August 1998, pp. 62–73.
- [11] J. E. Veenstra and R. J. Fowler, "MINT: A Front End for Efficient Simulation of Shared-Memory Multiprocessors," in *Proceedings of the 2nd International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Durham, NC, January 1994, pp. 201–207.
- [12] D. Bouvier, "RapidIO – An Embedded System Component Network Architecture," Tech. Rep. <http://www.rapidio.org/>, RapidIO Technical Working Group, June 2001.
- [13] M. Rangarajan, A. Bohra, K. Banerjee, S. Gopalakrishnan, and L. Iftode, "Split-OS: An Operating System Architecture for Clusters of Intelligent Devices," in *Work in Progress, SOSP'01*, October 2001, Technical report in preparation.