Computer Systems Laboratory Research Review

Joseph Pasquale
George Polyzos

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093–0114

April 1992

UCSD Technical Report CS92-236

Abstract

At the Computer Systems Laboratory at UCSD, we conduct research in operating systems, communication networks, performance evaluation, and distributed systems. This document reviews the current research within the Computer Systems Laboratory. We describe our involvement in the Sequoia 2000 Project, and we describe our work in the areas listed above. The final section contains descriptions of research projects undertaken by our graduate students.

This research is supported in part by Digital Equipment Corporation, the National Science Foundation, the Powell Foundation, NCR Corporation, IBM Corporation, the UC Micro Program, and TRW Corporation. For more information, contact the authors at the above address, or by FAX at (619) 534-7029. Joseph Pasquale may be contacted by e-mail at *pasquale@ucsd.edu* or by telephone at (619) 534-2673. George Polyzos may be contacted by e-mail at *polyzos@ucsd.edu* or by telephone at (619) 534-3508.

1. Introduction

At the UCSD Computer Systems Laboratory (CSL), we conduct research in operating systems, communication networks, performance evaluation, and distributed systems. In operating systems, we are focusing on high performance I/O software support for a variety of I/O intensive applications, such as multimedia applications and scientific applications. In communication networks, we concentrate on problems associated with supporting digital continuous media, such as audio and video, over high-speed networks. In performance evaluation, we are interested in characterizing the I/O time-behavior of scientific applications running on supercomputers, such as the Cray Y-MP at the San Diego Supercomputer Center (SDSC). In distributed systems, we are investigating mechanisms for decentralized control for effective group decision-making and distributed resource allocation.

CSL members include two assistant professors (the authors), and twelve graduate students, most of whom are doctoral students. CSL is equipped with a variety of workstations, including seven DECStation 5000/240's, two Sun Sparcstation II, two Sun 3/60's, one Apple Macintosh CI, and one NeXTStation. Networks within the laboratory include Ethernet and FDDI local area networks. CSL is connected to various UCSD campus networks, as well as a private campus-wide FDDI (dual-ring) network which is part of the Sequoia 2000 Network. This connects CSL directly with SDSC and the Scripps Institution of Oceanography, both members of the Sequoia 2000 project. SDSC provides CSL with access to its Cray Y-MP as well as various multiprocessor supercomputers. SDSC is also a primary node of NSFNET, providing CSL with excellent connectivity to the outside world.

This document reviews the current research within CSL. Section 2 contains a description of Sequoia 2000, a project in which CSL is participating. In the next four sections, we describe an overview of the work of the four research areas in the laboratory: operating systems, communication networks, performance evaluation, and distributed systems. The following section contains abstracts of past CSL doctoral dissertations. The last section contains research summaries of the graduate students.

2. Sequoia 2000

CSL is part of Sequoia 2000, a University of California multi-campus project which addresses the massive data storage, network, and visualization requirements of Global Change researchers.[1] Faculty investigators include computer scientists from UC Berkeley and UC San Diego, and environmental scientists from UC Santa Barbara, UCLA, and UC San Diego. The project was initiated by DEC with initial seed funding totaling $15 million for 3 years; additional sponsors are now being sought.

CSL, in collaboration with Domenico Ferrari at UC Berkeley, is responsible for designing and building the Sequoia 2000 Network [FPP92]. The Sequoia 2000 Network is a private, wide area packet switched network optimized for real-time transfer of large objects. It consists of 45 Mb/s T3 backbone connecting the Sequoia 2000 campuses, and 100 Mb/s FDDI local area networks for local distribution at each campus.

We are investigating fast integrated I/O system and network software architectures to support the real-time transfer of massive data objects between a terabyte database being designed at UC Berkeley and any number of scientists' workstations located throughout the Sequoia 2000 network. Much of this data is in the form of a time-sequence of highly detailed images, collected by satellites and remote sensors located on the ocean or on land, which scientists wish to process, visualize, browse and manipulate interactively. Consequently, the network must support high-bandwidth transmission of these images and meet various delay constraints to simulate visual continuity.

[1] M. Stonebraker, "Overview of the Sequoia 2000 Project," *Proceedings 1992 IEEE Computer Conference (COMPCON'92)*, San Francisco, CA, February 24-28, 1992.

Some of the issues we are working on include integration of I/O system software and network protocol software to allow dynamic construction of in-kernel datapaths between the network device(s) and I/O source/sink devices (e.g., the frame buffer), determination of where to locate compression and decompression processes, and determination of how to take advantage of prioritized packets containing hierarchically coded data so that the network has the flexibility of selectively dropping packets if necessary (e.g., to prevent congestion).

To understand the demands of the network, we are studying the I/O time behavior of Sequoia 2000 applications such as climate and ocean general circulation models. In addition, Sequoia 2000 requires tools which allow the geographically distributed faculty investigators to effectively collaborate: CSL is developing workstation videoconferencing and distributed whiteboard tools, supported by high performance multicasting communication protocols which minimize utilized network bandwidth while bounding delay.

3. Operating Systems

We are currently focusing on investigating system software structures which support I/O intensive applications. Current I/O system software found in most operating systems is ill-suited for the high throughput requirements of multimedia and scientific applications which have I/O components consisting of video, high quality audio, high resolution images, and massive scientific data sets [P+91]. We believe that operating systems design must shift from the traditional focus of managing CPU cycles to that of managing I/O channel bandwidth which is increasingly becoming the scarce resource [P92]. We are investigating I/O system software architectures which support fast I/O switching of communication resources (e.g. the system bus) to support multiple I/O streams with time-varying bandwidth requirements and other real-time constraints.

Consider a program which emits motion video at a rate of 30 frames/sec in a window with resolution 1280 $\times$ 1024 pixels. If each pixel requires three bytes for describing color, the total output bandwidth required by this program is approximately 118 megabytes/sec. Even after significant compression, a large fraction of the bandwidth of current workstation busses is consumed given existing I/O system software which inefficiently copies data between domains. Furthermore, there is a real-time delay constraint which must be met: each frame must be delivered to the video interface every 33 milliseconds, otherwise, the continuity of the video may be destroyed. If the program were a distributed multi-media application, where multiple nodes are generating video (and audio) which must be distributed to all the participants, one can easily see that the bandwidth and delay requirements quickly become overwhelming.

We are considering stream I/O architectures, specifically efficient methods of data movement between stream endpoints, which is important for high-bandwidth delay-sensitive continuous media. These endpoints may be a device's buffers and a process's address space (or vice-versa), or directly between the buffers of two devices (one being a source, e.g., a microphone, and one being a destination, e.g., a loudspeaker). Programmers are given the ability to create fast "in-kernel datapaths," which minimize inefficient data copying between memory buffers. Data is transferred between devices by going directly over the system bus or through a single kernel memory buffer while retaining control in user processes. In addition, we are providing better integration of I/O system and network software, optimizing the performance of the latter by using the same minimal data copying techniques.

We are considering distributing the kernel so that parts not only reside in main memory, but also, parts providing major functionality (not just device driver code), reside on I/O devices with powerful processors so that a great deal of processing can occur right on the I/O device board. For instance, when a video frame is captured from a camera into a video frame buffer, assuming there is a fast processor on the video board with the video buffer, it may be desirable to compress (or do other sorts of processing tasks) the video frame within the video board itself. Then the frame can be moved about, within the distributed kernel itself, or over a network, more efficiently since it is smaller.

Scheduling and synchronization is less understood, and depend greatly on the control flow organization of stream modules. For example, if each stream module is a coroutine or process, these can be scheduled and synchronized using already developed real-time scheduling and process synchronization techniques (if they are coroutine-based, synchronization methods are more limited, and coroutines cannot be preempted). However, data transmission along streams is inefficient if each stream module is a process, because of potential physical memory copying between processes. We are considering buffer preallocation strategies which help solve this problem. If, however, stream modules are simply procedures, and a single process describes the flow of control along a string of stream modules connecting source and destination endpoints, different problems arise. Scheduling these processes is more coarse-grained, and therefore less flexible, than in the previous approach. To avoid missing deadlines (or recover from missed deadlines), preemption may be required; therefore, this approach may be less efficient than having separately schedulable stream modules.

Over this last year, we have obtained a great deal of experience in building multimedia applications. The goal in building these applications was not to gain experience in developing multimedia software, but rather to push the performance of the underlying system software, and observe what problems would arise.

First, we built a simple audio conferencing system using standard system software, specifically the X Window System, SunOS, and TCP/IP, using Sun Sparcstations connected by Ethernet [TP91]. This study demonstrated the simplicity and naturalness of using the window system interface for audio, and how to support a form of continuous media without having operating system support for real-time guarantees. This was only possible because voice-quality audio does not make high demands on bandwidth, and delays introduced by the system are imperceptible by humans. To accomplish this, we implemented a simple strategy for recovering from missed deadlines by discarding "late" audio data.

For higher quality audio, and especially video, such simple strategies will not be appropriate. The software structure of the X Window System must be redesigned for supporting continuous media. To support requests which actually contain audio or video data, these must be processed within real-time delay bounds. A multi-threaded design is needed so that requests do not have to wait for previous requests to be completely processed. On a uniprocessor, it it important to allow for the preemption of a thread if a higher priority request (e.g., one containing a audio or video request) is received. This is support that must be provided by the operating system.

More recently, we constructed a video conferencing system using DecStation 5000 workstations connected by an FDDI network. The main observation has been that current workstations with their Unix-based operating systems can easily support audio, but not video, because of video's huge requirement for bandwidth.

We have also investigated file system design for very large file I/O such as that required for scientific applications or multimedia (e.g. audio and video) applications [MP91]. The design was driven by results of a file workload characterization study we carried out on a heavily utilized file system at UCSD. File requests at the system call level were characterized as either control (e.g. open, close, chmod), data read, or data write. At the disk level, requests were characterized as either data reads, data writes, control reads, or control writes. Measurements showed that disk level requests were broken down as follows: 15% data reads, 21% data writes, 8% control reads and 56% control writes. Although control requests dominate data requests, the average amount of data transferred by control requests was 250 bytes whereas for data requests it was 4500 bytes. The main observation was that file data and file control requests have very different patterns for accessing disk storage: file control requests arrive at a much higher rate than file data requests, but transfer much smaller amounts of information. This has a significant effect on performance when accessing large sequential files, which is common in scientific or multimedia applications. To increase throughput, the blocks of these large files are often stored contiguously on disk, so that minimal disk head movement is required to access them. This is important because disk head movement is the major cause of performance degradation. However, given the high fraction of control requests to disk, there is a high likelihood that the accesses to sequential file blocks will be interspersed by accesses to other parts of the disk because of control requests.

Our design was based on using separate storage structures to contain control information and file data. These storage structures are constructed using disk arrays, configured in different ways (e.g., mirrored or striped) so that performance for the corresponding control or data request pattern is optimized. A further improvement was obtained by optimizing file cache performance. Disk requests for cache backups (for reliability) were separated from those for cache flushes (for managing the cache storage). The third (and final) storage structure, a disk array configured as a sequential write-only log, is used for cache backups, so that only flushes need to go to the control or data storage. Our *multi-structured* file system can support much higher data rates than existing file systems, especially when the workload consists of very large file requests.

4. Communication Networks

In the area of Communication Networks we explore issues related to (a) the analysis and modeling of network traffic, (b) the performance evaluation of local, metropolitan, and wide area networks through measurement, analysis, and simulation, and (c) the design of protocols for high-speed real-time transport of Digital Continuous Media.

We have recently initiated an NSF-sponsored effort to **characterize and model the traffic** on the NSFNET backbone and other wide area network environments [B+92]. Unlike other similar projects, our goal is to model the network as a whole rather than characterizing network performance as perceived by an application. This approach focuses on the integrity of the network, and tools and methodologies to facilitate its graceful evolution. Our research should allow more accurate capacity planning studies in high speed realms and network tuning, such as of routing mechanisms, at optimal levels of granularity. This is of particular importance now since new services not only dramatically increase network traffic, but can also completely change traffic profiles.

Since FDDI is becoming the high-speed Local Area Network of choice and since the Sequoia 2000 network consists of interconnected FDDI rings, we are investigating how to extend the performance guarantees offered by the **FDDI synchronous mode** to interconnected rings. Furthermore, although the synchronous mode is an integral part of the FDDI standard, as well as a distinguishing characteristic of the FDDI medium access protocol, appropriate system software for synchronous mode is unfortunately not typically implemented. Therefore, we are currently modifying Ultrix device drivers and extending network protocols in order to enable applications with real-time requirements to take advantage of the synchronous mode delay guarantees.

In the area of **Metropolitan Area Networks** (MANs) we have focused on a key problem in the DQDB protocol. The Distributed Queue Dual Bus (DQDB) protocol is the channel access mechanism of the proposed IEEE MAN standard (IEEE 802.6). This is a very interesting system because its frame structure is compatible with ATM/B-ISDN and is believed to be the first form of high-speed connectivity to be offered for the local loop. We have developed the Rotating Slot Generator (RSG) scheme, which overcomes the well known unfairness of DQDB through architectural modifications that allow the system to operate essentially as a dual ring rather than as a dual bus. The basic idea is to use a folded dual bus, with the original slot generators embodied into a 'slot generator' station which is rotating with a fixed period around the 'ring.' Geometry then takes care of any inherent unfairness. Of course, the difficult part is to make the scheme operate without undue complexity and external intelligence or intervention, and also to guarantee that no traffic is lost during the transients. We present this design in [KPP92] together with some initial performance results which support our expectations for fairness, in terms of both throughput and delay. We further investigate the performance of the RSG scheme in [KPP92b] where we show that not only does the approach provide fair service, but it is also more efficient than other mechanisms proposed in order to improve DQDB fairness. RSG provides higher throughputs and lower delays, and also improves other system characteristics, such as the degree of discrimination between priority classes for multiple priority systems.

We have also contributed to the performance analysis of **Conflict Resolution Algorithms** for broadcast communication channels. These protocols are applicable to satellite channels and terrestrial packet radio networks. We have developed a queueing theoretic methodology for the analysis of these protocols which allows us to obtain exact steady state distribution results for the packet delay for many interesting configurations [PMV87]. In [PM91] we provide a complete delay analysis for the celebrated FCFS 0.487 Algorithm.

Another problem of increasing importance with the proliferation of portable computers and the integration of computing and communications is the provision of **location transparency** to users and applications. Providing location transparent services across a network is a problem which can be addressed at multiple levels, from the data-link layer to the application layer. In [CP92] we provide a survey of issues related to this problem as well as a set of existing and proposed solutions.

One of our main areas of concentration is investigation of network architectures and protocols supporting **Digital Continuous Media** (DCM), i.e., digital audio, video, and other media in which data is produced or consumed in a periodic manner (e.g., scientific visualization). Audio and video are often associated with real-time applications such as teleconferencing. The interactive nature of these applications imposes stringent delay requirements on the audio and video flows. Various guidelines set the end-to-end delay tolerance to tens or at most a few hundreds of milliseconds. Meeting these delay requirements may require dropping packets when they accumulate between the sender and the receiver because of transient congestion. Fortunately, humans receiving real-time audio and video have some tolerance to transmission errors. For example, telephone quality audio can usually withstand a 1-2% data loss during talkspurts. The perceived error may vary, however, depending on the particular encoding schemes used and the level of compression. These two observations lead to a characterization of DCM as *soft real-time traffic*. This traffic requires specific performance levels, but has some tolerance to data loss when the requirements cannot be met. **Hierarchical coding** (or component coding) techniques can split DCM signals into subcomponents of varying importance. Therefore, less important components can be dropped without causing an interruption of service. Communication can continue in the presence of delays, errors, or network congestion, with only a moderate drop in service quality.

Another important characteristic of DCM traffic is its relatively high volume. For example, motion video with 640 x 480 pixel-frames and 3 bytes of color per pixel, transmitted at 30 frames per second, requires about 220 Mb/s of bandwidth. Of course, efficient coding schemes and compression techniques can significantly reduce this bandwidth. Thus, for most broadcast quality video, 30-45 Mb/s is sufficient. However, even though current link transmission speeds can support such bandwidth requirements, most networks and protocols in use today are not capable of sustaining this throughput. As fiber-optic technology has moved the network bottleneck from the physical link speed to the switching and higher-layer protocol processing, network architects have been caught unprepared. The speed limitations of network protocols are unfortunately deeply rooted in the networking architectures, which do not seem to scale-up well to satisfy the requirements of DCM. New high-speed network architectures attempt to address this state of affairs by making every effort to minimize the processing required in each protocol layer and eliminate the unnecessary copying of information between layers. However, although many network designers are currently paying attention to the significantly higher speed and larger bandwidth-delay product (the amount of information "stored" in transit at any one time), they have devoted less attention to the specific characteristics and requirements of real-time DCM.

We are investigating network and transport layer protocols which provide various degrees of quality-of-service by exploiting the periodicity and the particular statistical properties of DCM flows. Simple, lightweight protocols, optimized for specific media characteristics, seem the most appropriate for supporting DCM traffic. These protocols eliminate some of the functionality of traditional protocols which is of limited use to DCM.

We also address the analytical performance evaluation of systems supporting DCM flows. For traditional data traffic, Poisson arrivals and either exponential or constant packet sizes (leading to M/M/1 or M/D/1 queueing systems, respectively), have been shown in the past to provide acceptable analytic approximations. DCM sources, on the other hand, generate traffic at regular intervals. Coding and compression of the information generated at these instants results in a variable data rate which depends on the actual information content.

Still, data are produced at times that are multiples of an easily identifiable basic time interval. From an analytic point of view, it then becomes more difficult, or even impossible, to justify exponential interarrival times assumptions, i.e., the Poisson assumption. Therefore, a more appropriate model of the output of DCM sources should have constant interarrival times and general "packet" length distribution (which can be determined experimentally). Therefore, a discrete time model with periodic arrivals and general service time distribution (such as a discrete time D/G/1 queue), naturally represents a single DCM flow going through a server. Analytical solutions for discrete time D/G/1 queues, i.e., a single DCM flow, are known. We are currently extending this model to represent multiple DCM flows.

Even with the availability of high-speed fiber-optic networks, the large data volume of DCM requires extremely efficient data transfer mechanisms. Therefore, support for efficient multicasting is an important requirement, well recognized, but poorly supported at the present time. **Multicasting** refers to one-to-many communication, in contrast to *unicasting* which is one-to-one communication, and *broadcasting* which is one-to-all communication. Multicast can be thought of as intermediate service between unicast and broadcast, and, unfortunately, many times it is implemented as either a broadcast (with filtering at higher protocol layers), or as a series of unicasts from the source. These naive approaches to multicasting are inefficient because they may place an unnecessary load on the network. Broadcasting sends the data to many more nodes than required. Multiple unicasts will likely use the same link several times to carry the same data, and the source must work harder to send the multiple copies. However, multicasting can be done efficiently using tree-based communication structures. These structures can reduce bandwidth utilization by delaying duplication of data until the routes to different destinations diverge, thus reducing the number of copies sent over the network. In addition to reducing bandwidth requirements for multimedia communication, it is important to determine routes which have low end-to-end delay from the source to possibly multiple destinations. In selecting data paths, one should minimize the crossing of either network relay nodes or abstraction layers (e.g., protocol layers), both of which contribute to delay in data delivery. Switching, queuing, protocol conversion, as well as propagation delay all contribute to total delay. However, since delay is often tolerable below a certain threshold level, a good routing algorithm does not necessarily have to minimize the delay along the route; bounding the delay is adequate.

The problem of finding optimal tree forwarding routes from a source to the members of a multicast group under given delay constraints is intractable. Therefore, we have developed heuristics which are both efficient and effective, typically providing feasible solutions that are only a few percent less than optimal. Source routing heuristics are based on an abstraction of the problem from a full network to that of a complete graph, with the members of the multicast group (only) as vertices, and the shortest paths (with respect to bandwidth — or link count) between any two such members as edges. Trees on this much smaller graph are much easier to track. In particular, we show that for two metrics proposed as selection functions for the heuristics, one leads to better solutions in the source routing case and the other in the distributed routing case. We are investigating the performance of these schemes both empirically and analytically on random graphs and graphs resulting from specific network topologies.

5. Performance Evaluation

Based on the need to design well-balanced supercomputer systems that deliver the high performance expected of them, we are undertaking a comprehensive study of the relationship between the I/O requirements of scientific applications and the capabilities of the I/O system. We are working closely with the San Diego Supercomputer Center (SDSC) which is one of the four national supercomputer centers sponsored by the National Science Foundation. SDSC currently maintains a Cray Y-MP supercomputer (among other machines) which we have been measuring. SDSC offers Cray time to users all over the United States, who generally wish to run large scientific applications. These users comprise 2700 researchers from 170 institutions, spanning 44 states.

The goals of this research are:

1. to characterize the I/O behavior of a wide spectrum of scientific applications
2. to develop models for well-defined classes of scientific application I/O activity
3. to construct an executable I/O benchmark set for the purpose of locating existing or potential I/O bottlenecks
4. to recommend operating system software enhancements to improve supercomputer I/O performance.

The first goal addresses the problem that our understanding of scientific applications needs to be improved. Scientific applications form an interesting set of programs worthy of study because they absolutely need high performance supercomputers and therefore, systems should be designed to meet their requirements. Before we can do the latter, we must understand these requirements. This involves observing an environment supporting a variety of scientific applications (e.g. SDSC Cray Y-MP), and determining what to observe to deduce these requirements in a way that does not interfere with their execution.

The second goal involves summarizing these observations into a form that may be used for descriptive or predictive purposes. The intent of the third goal is to provide a comprehensive set of scientific I/O benchmarks so that users can select the I/O requirements that best represent their own workloads. Finally, the fourth goal is to put results into practice: to improve I/O performance on the supercomputer at our disposal which supports scientific workloads. This will be limited to operating system software enhancements, such as the design and implementation of I/O resource scheduling algorithms, or modification of software cache sizes. This may also involve suggestions to the designers of applications of how to improve their codes.

We are focusing mainly on application characterization rather than overall system behavior because our intent is to investigate not only current but also future potential problems which are not limited to what is observable in current systems. Specifically, we are interested in understanding the issues involved around a potential I/O bottleneck in future supercomputers and to provide tools to prevent it. This scenario is associated with supercomputers that are computationally much more powerful than those of today and also with workloads that are much more I/O intensive. The latter is driven by user expectations about performance of the future machines. Based on current application I/O characterizations one can extrapolate future I/O application requirements, and then through synthesis determine overall I/O workload intensities and patterns.

Our emphasis on application I/O and what we learn will be applicable to and benefit a larger audience including disciplines outside of computer science. It is our experience that the scientists and engineers running these codes are very interested in learning about the performance of their applications.

We also performed a workload characterization study of the SDSC Cray X-MP (the predecessor to the Cray Y-MP), running the CTSS operating system [PBK91]. In this work, our underlying strategy has been to recognize the attributes of the workload being submitted by the national academic research community for execution on the supercomputers at SDSC. Many of the jobs comprising the workload require either large fractions of a given resource, or require support which traditionally has not been available on supercomputers.

The workload was characterized at the functional level, by grouping jobs according to the name of the command executed by the job. These commands were classified as either public or private, depending on whether the program was available to all users (e.g. system utilities, generally available scientific packages) or just a single user (e.g. user-developed codes not made generally available).

We found that 93% of all jobs were public jobs, yet 90% of all CPU time was consumed by private jobs. Although the public utilities make relatively small demands on the CPU, they consume 22% of the total memory space-time product, and 75% of the total I/O channel time. The latter indicates that a very significant I/O demand is being made by the public component of the workload.

The workload was also characterized at the resource level, by classifying jobs according to their resource consumption statistics, specifically CPU time, I/O channel time, and memory space-time product. The jobs were then clustered according to their command name and resource usage, using a variant of the *k-means* clustering algorithm.

We related the functional and resource level characterizations by (1) examining what command types made up each cluster and (2) seeing what percentage of system resources each cluster consumed. The cluster **Pub-Low** contained commands whose functions are listing, removing, displaying, and deleting files. **Pub-Medium** contained commands whose functions are interactively managing files, printing manual pages, editing text files, and searching files for specified text strings. **Pub-High-CPU** contained commands such as *g86lib*, which is a quantum chemistry application. **Pub-High-I/O** contained commands such as *ftp*, which performs file transfers.

Table 1 shows the seven clusters produced by this analysis, including the amounts and relative percentages of system resources consumed by each cluster.

Table 1: Individual Cluster Statistics

Class	Number of Jobs	%	CPU (s)	%	I/O (s)	%	Mem-Time (Mw × s)	%
Pub-Low	471392	46.60	5268	0.03	53870	1.72	2268	0.01
Pub-Medium	415267	41.05	254723	1.50	892470	28.50	128604	0.82
Pub-High-I/O	43975	4.35	242541	1.42	1291697	41.25	1581887	10.11
Pub-High-CPU	17038	1.68	1234901	7.24	99618	3.18	1782237	11.39
Priv-Low	17712	1.75	4836	0.03	5888	0.19	1019	0.01
Priv-High	27889	2.76	689063	4.04	111281	3.55	146545	0.94
Priv-Massive	18318	1.81	14631825	85.75	676757	21.61	12002096	76.72

The basic results were the following. 88% of all jobs are either in clusters **Pub-Low** or **Pub-Medium**, but these jobs used only 2% of the total CPU time. Only 2% of all jobs are in the cluster **Priv-Massive**, which used 86% of the total CPU time. **Priv-Massive** was also the largest user of memory, utilizing 77% of the total memory space-time product (over all jobs). Three clusters used 92% of the total I/O time: **Pub-Medium**, **Pub-High-I/O**, and **Priv-Massive**. One important observation was that even jobs that make insignificant demands on the CPU (e.g. interactive system utilities) can have a major effect on the I/O system.

6. Distributed Systems

Resource sharing is a major reason motivating the design of distributed computer systems. We are concerned with decentralized allocation of resources, its fundamental problems, and how to overcome these problems [P91]. Decentralized control offers many benefits over centralized control algorithms, such as speed (through parallel processing and keeping control near data), reliability (by avoiding single points of failure), scalability (because of symmetric control relationships), and autonomy (by allowing control over local resources to be kept locally).

However, these benefits have not been fully realized because of difficult problems associated with distributing control. There are two fundamental problems: (1) no decision-making agent can be certain of the global system state; (2) no decision-making agent can be certain of the global system action. The first problem exists because because communication cannot occur instantaneously (and we assume there is no shared memory in a distributed system with decentralized control). Thus, any communicated state information necessarily describes a past state, and at best, the current state may be inferred but not known with complete certainty.

The second problem is more subtle and exists because there is no common knowledge about actions. It is often the case that an agent makes a decision based on its view of the global system state, and even if this view is perfect (i.e., (1) is not a problem), its decision may conflict with decisions made by other agents at the same time. This is best illustrated through the following example. Consider a load balancing system, where every agent which makes decisions about offloading jobs knows perfectly the global system state. The agent on a heavily loaded machine will decide which machine is least-loaded, and offload there; that is the locally optimal decision for that agent. However, if there are many heavily loaded machines, those agents may all decide to make their locally optimal decision which is to offload to the same least-loaded machine, and overload it. Thus, these locally optimal decisions conflict, and result in a global disaster.

We are focusing on three aspects of these problems: the effects of global state uncertainty on decision-making; multi-agent coordination; and group formation. Our approach to the global state uncertainty problem is to characterize two loss functions: loss due to decision quality degradation, and loss due to communication overhead, both as a function of state update period. The loss due to decision quality degradation is low for small update periods (because the communicated state information is timely), increases as the period gets larger, and eventually levels off at some maximum value (because after some point, the information becomes worthless and remains worthless regardless of how much older it gets). The loss due to communication overhead is infinite for a period of zero, and rapidly decays as the period increases. Adding these loss functions gives the total loss, which generally has some minimum point, which is the optimal communication period.

We have developed a new model for distributed decision-making called distributed game automata, which focuses on how communication affects decision-making quality [BP92]. Agents are modeled as stochastic automata which play games with each other, and which try to improve performance using learning heuristics. Performance is quantified by payoffs from these games. The model allows us to determine the extent to which one can limit communication between agents such that its cost is reduced but good decisions still result.

Decision-making problems in distributed systems generally require coordination among the entities in the system. We have investigated agents that coordinate their actions by independently adapting their decision functions in ways that increase the global effectiveness of decisions. As each agent's decision function changes, the global decision function changes; we call this co-adaptation, literally "adapting together." As each agent finds its appropriate individual decision function, the collection of agents co-adapts into a good, although not necessarily optimal, global decision process. We applied co-adaptive decision-making to the problem of distributed job scheduling in a distributed computer system, and found this to be effective in load balancing [GP92].

Finally, we are considering how agents dynamically form groups such that these groups are more effective in solving problems than independent agents. A group is recognized by the amount of communication taking place between agents: agents within a group communicate at a much higher rate than agents in different groups. We wish to understand how groups form spontaneously, why they form, and whether such activity can be predicted. We are considering a number of prediction models based on game theory (agents within a group playing games, and groups playing games against each other) and learning automata (adaptive algorithms based on environmental feedback which reinforce or break down inter-agent relationships).

7. CSL Doctoral Dissertations

This section contains abstracts of past doctoral disserations by CSL students.

K. Muller: Multi-structured File System Design for High-performance Computing, 1990

File system I/O is increasingly becoming a performance bottleneck in large distributed computer systems. This is due to the slow decline of disk access times (latency and seek) relative to the rapid increase in CPU speeds, memory size, and network bandwidth. Indeed, although disk storage capacity increases while the physical size and cost of disks decrease, file system performance remains limited by long disk access times and disk contention. The problem is further aggravated by new applications with intensive I/O components, such as multi-media or visualization programs.

Technology trends indicate that it is unlikely that disk access delays will significantly decline in the near future. Thus, solutions cannot be based on improvements in disk access time, and must somehow circumvent this problem. Until recently, the high cost of disk storage, limited the design of file systems to techniques which optimize storage efficiency. However, with the rapid decline in disk costs, solutions explicitly designed to optimize performance have become very attractive.

We present a design for a multi-structured file system. Our design is based on the coordinated and cooperative integration of disk caching with three different storage structures each implemented on independent and isolated disk systems. The individual storage structure were designed to optimize a different set of file system access characteristics such as cache writes, directory searches, file attribute requests or large sequential reads/writes. The isolated storage storage structures reduce contention and lower disk access delays, so high bandwidth I/O and decoupled concurrent access to multiple disks can be provided. Both file system performance and storage size can be scaled to meet the requirements of different work loads by adding disks to the appropriate storage structures. As part of our study, we analyze the performance of an existing file system using trace data from UNIX disk I/O-intensive workloads. In trace driven simulations, we show how performance improves using our technique of dividing file system requests based on access type to optimized storage structures, as implemented by our multi-structured file system.

A. Glockner: Coadaptive Decision-making in Distributed Computing Systems, 1991

Decision-making in distributed computing systems is complicated by the introduction of communication delays and the resultant uncertainty both in the global system state and in current actions in the system. This dissertation examines the effects of introducing *coadaptation* into distributed decision-making. By constructing the decision-making process as a set of autonomous agents that can individually learn from feedback, the set of agents adapt together, or *coadapt*, to form a good, though not necessarily optimal, global decision process.

These effects are demonstrated with simulation experiments in distributed computer job scheduling; the agents are implemented as learning automata. The resulting behavior is analyzed both in terms of a global performance metric — mean job turnaround time — and in terms of the states of the agents themselves. Several key elements in the behavior of coadaptive systems are illustrated. These include demonstrations of: situtions where apparently good individual decisions result in worse performance, correlations between automata type and decision-making policy of the agents, complementarity in agent behavior, differences in the relative performance of agents both with small changes in agent parameters and and with changes in the environment, and properties of stability in coadaptive systems. Simulations performed with changes to agents and to the distributed computing system model show further principles of coadaptive behavior. The dissertation concludes with a summary of principles of coadaptive behavior and a projection of future methods of coadaptive distributed job scheduling.

8. Graduate Student Research Summaries

Research summaries of the graduate students who are members of the Computer Systems Laboratory follow.

Operating System Support for High Speed Digital Media

Eric Anderson

High speed digital media communication, such as video teleconferencing and multimedia shared workspaces, promises to deliver new communication potential for scientists. Using computer workstations, researchers will be able to visualize, analyze, and share information with greater ease and speed than current technologies permit. This enhanced interaction will facilitate advances in fields which rely on the prompt analysis and exchange of information. Project Sequoia 2000 is an ideal example of the research which will be best supported by a system for the rapid exchange of large visual data sets.

Advances in operating systems are necessary to support the demands of high speed and high volume communication. Modern workstation operating systems are not capable of handling the data transfer rates required for continuous video presentation. Workstation hardware has evolved to the point where preliminary work is possible. Generally, system software, such as the Unix operating system and its derivatives, is unable to take full advantage of this new hardware. Future hardware will further exceed the capabilities of present operating systems.

I am studying operating system enhancements which will make possible the full use of modern digital media, including audio and video. Options which I intend to explore involve expediting the transfer of data within the system. Faster data transfer is required between network interfaces, user processes, and audio and video devices, such as cameras and graphic displays. In addition, system support for the optimal transfer of compressed data, and for the reconstruction of damaged data, will permit greater usage of available resources in the face of network limitations.

A private network has already been established to connect Sequoia 2000 labs in Berkeley, Santa Barbara, Los Angeles, and San Diego. This network of 1.5 megabit per second T1 links has the capacity to support digital media communication. The network will be enhanced in 1993 to support the T3 rate of 45 megabits per second. This rate will be sufficient to transfer multiple video signals of reasonable quality simultaneously. Preliminary research has shown that existing workstations have difficulty communicating digital video at the current network rate of 1.5 megabits per second. Exchanging video at rates approaching 45 megabits per second will not be possible without advances in operating system technology.

I am presently pursuing two projects: A prototype video conference application, and a distributed whiteboard application. These projects have dual purposes: They each serve as a collaboration tool for Project Sequoia 2000, and they will be put into use immediately when they become available. More importantly, each has already illustrated the areas where extra support is required from the operating system in order to make full use of available hardware resources. Both have been tested across the Sequoia network, and the video conference application has been tuned to use the entire 1.5 megabits per second of available network bandwidth.

The Effects of Delayed Communication in ACTION-SPACE Randomization

Ted Billard

My interest is in large distributed systems in which decision-makers or agents must make good, though not necessarily optimal, decisions. It is assumed that the costs of communication and the nature of the systems do not permit high bandwidth communication in making each decision. There is a tradeoff in the frequency of communication: too much communication increases the overhead and too little communication results in poor decision-making due to an inaccurate view of the global system state.

I have developed a new model for distributed decision-making called distributed game automata, which focuses on how communication affects decision-making quality [1]. Agents are modeled as stochastic automata which play games with each other, and which try to improve performance using learning heuristics; performance is quantified by payoffs from these games.

The decisions in the game regarding what action to perform are made probabilistically; I call this *action* randomization. The agent must also decide, probabilistically, with whom to coordinate; I call this *space* randomization. This implies finding a subset of agents, from a possibly large network, that have compatible strategies, resources and constraints.

Each agent's view of the global state is updated by communication from other agents of their local state (i.e. current action and space probabilities). Delay is modeled by lagging the transfer of the local state such that each agent's feedback from a move is based on aged information.

I have examined a load balancing application where each agent has a stream of jobs and it is best to place jobs on distinct processors. Each agent has two decisions to make; first, in the space dimension, shall the agent choose to coordinate with other agents in the system? The result of all of these decisions determines whether a coalition is formed, hence the potential for improved performance due to synergy, say the use of shared resources. Otherwise, the agents act alone and the performance is typical of an isolated agent. The second decision, in the action dimension, concerns the choice of processor for a job; however, there is uncertainty in the result of the first decision (i.e. is the agent in a coordination or non-coordination mode?) and uncertainty in the other agents' probabilistic strategy for placing jobs.

Simulations show two primary, but similar, results [2], [3]. If the uncertainty concerning coalition formation is removed (by always forming a coalition), there is a maximum communication delay before poor decisions result concerning processor selection; the agents' decisions oscillate between overloading one processor and then another processor.

If the uncertainty concerning processor selection is removed (by assuming rational choice), there is a maximum communication delay before poor decisions result concerning coordination; the agents adaptively reach a local maximum in the non-coordination mode even though there is a potential for improved performance in the coordination mode. The equilibria attained by space randomization agrees with an analytic determination and can be categorized as strongly stable, stable or weakly stable. This has implications for agents that search for optimal groups within a network.

The performance of this environment is compared to a baseline environment in which all uncertainty is removed; this baseline assumes that all agents have common knowledge of coalition decisions and that rational decisions are made concerning processor selection [4]. I have also examined cases where it is better for one agent to always know a coalition fact than for two agents to sometimes know the fact (and vice versa).

My work illustrates the fundamental uncertainty and the role of communication in decision-making and the level of knowledge in a distributed system.

References

[1] Billard, T. and Pasquale, J. Modeling delayed communication in distributed systems using distributed game automata. Submitted for publication.

[2] Billard, T. and Pasquale, J. Effects of delayed communication in coalition formation. Submitted for publication.

[3] Billard, T. and Pasquale, J. Effects of delayed communication in adaptive coalition formation. Submitted for publication.

[4] Billard, T. and Pasquale, J. Common knowledge of coalition formation in distributed systems. Submitted for publication.

[5] Billard, T. and Pasquale, J. A Taxonomy of Probabilistic Coalition Games. Submitted for publication.

[6] Billard, T. and Pasquale, J. Optimal Communication Periods for Coalition Strategies. Submitted for publication.

Characterizing the I/O Behavior of Scientific Applications

Barbara K. Bittel

Scientific applications represent the class of computational problems that require extensive resources in terms of CPU cycles, memory and data transfer. It is exactly the resource demands of these scientific applications that continue to fuel the need for larger and more powerful supercomputers. Research and progress in CPU performance has kept pace with these demands whereas improvements in I/O systems and memory systems are seriously lagging behind. As a result, supercomputer I/O architectures are becoming less able to match the requirements of today's I/O demands and will be a major obstacle to achieving high performance in future computer systems [1].

The goal of my research is to develop formal descriptions of I/O behavior for well-defined classes of scientific applications. Obtaining a thorough understanding of the I/O behavior of the workload is an essential first step in the design, simulation, and evaluation of new supercomputer I/O systems. In a recent study, Miller and Katz [2] profiled the I/O behavior of 7 scientific applications running at NASA Ames to investigate various buffering and caching policies. The profiles obtained exhibited a highly regular pattern of bursty I/O request rates, due to checkpointing and data staging, which seem to indicate that I/O resource scheduling is one viable solution for enhancing I/O support. To explore this area further, I shall investigate the I/O behavior of the San Diego Supercomputer Center (SDSC) Cray YMP. SDSC is one of four NSF supercomputer centers and supports a wide variety of users from many scientific disciplines, making the observable workload worthy of study.

Currently, I am conducting a general workload characterization study of SDSCs Cray YMP to identify the major scientific applications and to classify these applications in terms of static resource usage and I/O attributes. Data for this work has been obtained from Cray System Accounting files and includes such parameters as system and user CPU time, maximum memory size, and physical and logical I/O requests. Following this static characterization, representative applications from each class will then be monitored and I/O activity traced to characterize the dynamic I/O behavior over time. Particular measures of interest are: number and types of I/O events, duration of events, bytes transferred, time between events, mode of transfer, number of active data files. Since many of these measures and events are not automatically recorded, it will be necessary to instrument the operating system to obtain the required data. Combining both the static and dynamic characteristic features obtained through analysis and measurement, it will then be possible to derive formal descriptions of I/O behavior for classes of scientific applications.

References

[1] Katz, R. H., Gibson, G. A., and Patterson, D. A., "Disk System Architectures for High Performance Computing," *Proceedings of the IEEE*, Vol. 17, No. 12, December 1989, pp. 1842–1858.

[2] Miller, E. L., and Katz, R. H., "Input/Output Behavior of Supercomputing Applications," *Proceedings of Supercomputing '91*, November 1991.

Analysis and Modeling of Wide-area Networks

Kimberly C. Claffy

During the past several years, the Internet community has invested significant efforts in the development of computer networking infrastructure. Despite the tremendous growth of what is now a pervasive global infrastructure, little attention has gone into the investigation, modeling, and analysis of this infrastructure. It is essential to postpone no further an exploration of research strategies which can enable a greater understanding and predictability of the behavior of the deployed network, as well as models and methodologies for facilitating future network development into ever higher speed and more ubiquitous realms.

The sparse existing documentation on aggregate traffic behavior within the Internet environment was created years ago; current research into network analysis addresses only a subcomponent of the network. Without a larger systems approach, these efforts lack realistic applicability to the current system.

One essential ingredient to a greater understanding of traffic behavior is a set of methodologies and tools that not only analyze and model available data in its multidimensionality, but also assist in defining additional data sets for further evaluation and optimization of the network. Such capabilities become only more critical as the Internet and its NREN component evolve at a dramatic speed toward higher performance and ubiquity.

In this important new area of research I will have to compensate for the existence of only rather sparse existing library of professional references upon which to draw. Other institutions are also just beginning research in this area, and we hope to pursue collaborative efforts with those who have similar interests in applicability to the large scale network system.

I will initially focus on the analysis of existing data and attempt to define key research issues, guidelines for the development of realistic traffic models for networks such as the NSFNET and the upcoming NREN. We expect the analysis and modeling results to influence traffic characterization on the Internet at large. Later stages of investigation will inevitably bring refinements to our modeling methodologies, refinements which will benefit not only future Internet management, but also the research, development and maintenance of other networks in today's modern high-speed realm. Initial areas of project investigation include network latency assessments, bottleneck identification, performance monitoring, finer-grained analysis of time domain, finer-grained analysis of traffic, usage characteristics, application determination and prediction of bandwidth requirements, routing stability, traffic locality, and performance metrics oriented toward user requirements.

My research effort will investigate national and international networks as statistics data becomes available, and will over time include collaboration with both domestic and international researchers, in order to help define common goals, analysis methods and models.

Operating System Support for High-performance I/O

Kevin Fall

My research is focused on operating system software support for high-performance I/O. With particular attention paid to the bandwidth and delay requirements characteristic of multi-media information, I am interested in software structures capable of routing large continuous data streams quickly. Moving large amounts of data requires efficient software structuring to avoid unnecessary data copies and virtual memory mapping operations. To date, the research has included several phases. Initially, I worked with other members of our research group at UCSD Computer Systems Lab to help design a multicast "software pipe" capable of providing connection management for several unidirectional multicast streams. This design was built atop the Berkeley UNIX socket interface using TCP/IP. From this design, we became aware of the limitations imposed by the traditional socket interface when used in a multicast, multimedia environment.

The second phase of my research has been to identify the particular areas of the traditional UNIX I/O subsystem which need to be re-examined in the face of multimedia, multicast streams. A method for abstract device characterization as suggested by Cheriton [1] would allow applications to determine quantitatively what performance characteristics they can reasonably expect from an I/O device. For example, an application may need to know it is communicating with a 44.1KHz digital audio device rather than a voice-quality mono 8KHz device. Additionally, the architecture of a stream-based I/O system providing the ability to interconnect I/O devices within the operating system itself has lead to my most recent work.

Most recently, I have been investigating different buffering strategies and control strategies to be used in a general stream-like I/O subsystem. Researchers at the University of Arizona indicate improved performance over the Berkeley UNIX network buffering scheme *mbufs* whereas my own results indicate a contradictory result. My results are based on a comparison of the 4.4 BSD UNIX implementation of mbufs and the XKernel's message structures [2]. The XKernel is a small operating system designed for implementing protocol stacks. I plan to create a buffering strategy incorporating the performance advantages of mbufs but with an improved software structuring, resulting in improved modularity and ease-of-use.

Moving data quickly through an operating system entails not only an efficient buffering mechanism but also a control flow strategy which minimizes the execution path taken to deliver data to its consumer. Systems based on the UNIX model involve a downward calling sequence, from user program to hardware. Programs are made to wait the arrival of data they request. An alternative known as *upcalls* [3] can improve performance by creating an execution thread at the hardware level and propagating received data upward toward a user program, causing it to be run immediately. The XKernel is designed to use upcalls in support of incoming data manipulation. I intend to use this system as a starting point for the design and implementation of a device-driver "toolbox" containing a set of modular functions used by all device drivers. The toolbox will provide a uniform interface allowing for the direct in-kernel connection between devices.

My immediate work involves the verification of my buffering performance results, and the beginning of a detailed study of device drivers. The goal is to ascertain points of commonality between different device drivers leading to an indication of the functions which should be included in the toolbox. Due to our lab's involvement in the Sequoia 2000 project recently awarded to the University of California by Digital Equipment Corp., my work will be tested in an operating system vehicle such as the X Kernel, for eventual integration into OSF/1 or OSF/2 (if it retains this name). The next phase of my work involves further performance studies and the design of a new I/O subsystem optimized for constructing in-kernel datapaths, allowing devices to interconnect without the intervention of a user-level process.

References

[1] Cheriton, D.R. UIO: A Uniform I/O System Interface for Distributed Systems. *TOCS*, Vol 5, No 1, pp 12–46, Feb. 1987.

[2] Hutchinson, N.C., Mishra, S., Peterson, L.L., and Thomas, V.T. Tools for Implementing Network Protocols. *Software Practice and Experience*, Vol 19, pp 895–916, Sept. 1989.

[3] Clark, D.D. The Structuring of Systems Using Upcalls *Proc. 10th Symp. on Operating System Principles*, pp 171–180, Dec. 1985.

Traffic Characterization on the NSFNET

Nancy Groschwitz

The NSFNET is the cross-country backbone of the Internet. The purpose of my research is to examine the characteristics of the NSFNET traffic.

Over the past twenty years, as the original ARPA network evolved into the current Internet, it has expanded in size and complexity at a rapid rate. Accurate information about the behavior of the Internet is required to assess current performance and to plan for future growth. Such information could also be used to develop new models of the system, to verify and correct existing models, or to test alternative algorithms and protocols.

Initially, the NSFNET site examined most closely will be the San Diego Supercomputer Center, which has single-hop links to three other NSFNET backbone sites: Houston, Palo Alto, and Seattle. The research will later be expanded to include other NSFNET nodes.

The available data includes 15-minute and monthly counts of total number of packets and bytes, number of errors (damaged packets), delay data, and service type. The delay data consists of round trip times from San Diego to each of the three single-hop nodes (Seattle, Palo Alto, Houston), measured from an external address to an external address; thus, it is comparable to what might happen to a normal data packet.

Examination of packet and byte counts shows daily and weekly cycles that closely resemble traffic patterns that have been reported in the literature for many local area networks. I am currently conducting time-series and autocorrelation analyses to look for long range trends. Although the total traffic trend, over the years available, is overwhelmingly up, I want to see if there are other cycles that overlay this basic trend.

Preliminary results show that the autocorrelation between successive 15-minute samples is very high (above .90); there are also 24 hour oscillations in the autocorrelations. In other words, the correlation is highest for the immediately preceding 15-minute bin, but is also high for a 15-minute bin from exactly 24 hours earlier, and from successive bins set at 24-hour intervals.

I am also examining the delay and error data, looking both for long-range trends and for short-term correlations. For example, most 15-minute data bins have no packet errors at all, but some have as many as several hundred. I want to determine whether the packet errors can be correlated with anything else — delays, link utilization, time of day, or even hardware errors.

I eventually intend to extend this research to all nodes of the NSFNET, and to examine relationships between nodes. For example, is traffic between some node pairs higher than between other node pairs? Is there a correlation between traffic rates between node pairs and average delays? Even without a correlation, are there some node pairs that habitually have longer delays, or more errors, than the network average?

Network Software Optimization

Jonathan Kay

I am working on implementation techniques to improve network protocol throughput. A big problem with modern workstations is that their networking software cannot keep up with network speeds. A major problem for the Sequoia 2000 project is that networks cannot keep up with scientific visualization data. I am currently working on finding bottlenecks in network throughput, and I plan on using that study to learn how to greatly improve network software.

Over the course of the last year, I have made two studies of two different operating systems [1]: Ultrix 4.2 running on a DECstation 5000/200 over FDDI, and the x-kernel (an operating system specially designed for network research) [2] running on a Sun 3/60 over Ethernet. The methods used were was similar in each case. Many have suggested that network performance is largely held up by a few operating system primitives [3], [4], so I chose a number of functions used by network software (copying, checksumming, memory allocation, memory descriptor operations, etc.) and measured how much time was spent to send a packet. I concentrated on large UDP packets, since I am mostly interested in improving throughputs, and NFS-generated UDP is the most common type of packet on the CSE networks.

I found that under Ultrix, 40% of the time is spent checksumming packets, 40% is spent copying data, 10% is spent on memory allocation, and 10% is spent doing everything else. The authors of the x-kernel designed to avoid copying, and chose not to implement data checksumming, so in that case memory allocation is dominant at 30%, copying took 20% of the time, and the miscellany that took 10% under Ultrix took 50% under the x-kernel.

I have already begun to use the data above to improve network software performance. Under Ultrix the greatest expense was the checksum, and as it turned out the checksum was poorly designed for a RISC processor. I tripled the checksum function's performance by redesigning it for RISC. That alone yielded a 25% improvement in overall network throughput. Similarly, reduction of x-kernel malloc usage improved performance there by 15%.

Unfortunately, another thing that the studies described showed was that the hardware clocks included in workstations typically do not have enough resolution to study the cheaper operations, even using the sort of statistical techniques normally used in profilers. I have borrowed a logic analyzer with a 10 ns resolution to make a improved study with improved resolution, this time of both a DECstation 5000/200 and a Vax 11/780 to get a picture of how performance characteristics are changing with time.

Once performance studies are complete, I plan to work harder on performance improvements. I believe that it is possible to severely reduce most of the existing overheads, partially by more careful redesign with the idea that copying is expensive, partially by use of integrated layer processing techniques [4].

References

[1] Kay, J., "Network Sloth: Where the Time Goes in Network Software," UCSD Technical Report CS92-238, April 1992.

[2] Hutchinson N. C., and Peterson, L. L., % "Design of the x-Kernel," *Proceedings of the SIGCOMM '88 Symposium*, pp 65–75, Stanford, CA, Aug 1988.

[3] Watson, R. W., and Mamrak, S. A., % "Gaining Efficiency in Transport Services by Appropriate Design and Implementation Choices," *ACM Transactions on Computer Systems*, 5(2), 97–120, May 1987.

[4] Clark, D. D., Jacobson, V., Romkey, J., and Salwen, H., "An Analysis of TCP Processing Overhead," *IEEE Communications Magazine*, pp 23–29, June 1989.

[5] Clark, D. D., "Modularity and Efficiency in Protocol Implementation," *Internet RFC No. 817*, 1982.

[6] Clark, D. D., "The Structuring of Systems Using Upcalls," *Proceedings of the Tenth ACM Symposium on Operating System Principles*, pp 171–180, Dec. 1985.

Multicast Routing for High-Bandwidth Delay-Sensitive Applications

Vachaspathi Kompella

The problem under investigation is how best to transport large volumes of data to multiple nodes in a network, given a delay constraint that represents the lifetime of the data. An example of such data is digitized voice/video information from a live conference. If the digitized data reaches late, it loses its value.

Multicasting is a convenient mechanism for such communication. In addition, it can also make efficient utilization of network bandwidth. This is possible when the same data has to be transported to several nodes in a network. If we were to send the data to each node individually, it is likely that a link may be used several times, once to carry the data to one node, a second time to carry the data to another node, and so on. However, if we build a tree structure and inform the relaying nodes to route the data along the links of the tree, then the duplication of data will be minimized. Fig. 1 below shows such a tree, in which A is the source node, and B and E are destination nodes. The tree is shown in thick lines.

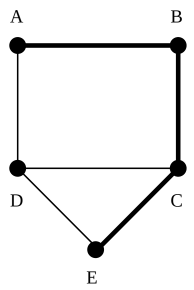

Fig. 1 Multicast from A to B and E.

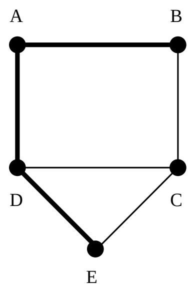

Fig.2 Multicast from A to B and E (bounded delay)

A problem with the tree of Fig. 1 is that, given equal delay on any link, the delay from node A to node E may be excessive. In particular, in a conference, the participant at node E may always be the last to hear what is being discussed. This would effectively leave E out of the conversation. The problem we are looking into attempts to add a delay constraint to multicast route determination, so that no path from one participant to another will have too large a delay. If the delay had to be under 3 units of time, then a tree as shown in Fig. 2 would be constructed. We call this tree a *constrained Steiner tree* (CMT).

The "vanilla" multicast route problem is NP-complete, and with the additional delay constraint problem, it is just as hard. We have devised a heuristic that produces near minimum cost CMTs if information on the network status (*i.e.*, edge costs and delays) is available to the source. The performance of this heuristic is documented in [1]. The heuristic scales well to large networks. We have also developed a distributed algorithm that performs well if only local information about adjacent edges is known to each node. These results can be found in [2].

The work on multicasting will be used to make collaborative research possible over the Sequoia 2000 network that spans four University of California campuses.

References

[1] Kompella, V.P., Pasquale, J.C., and Polyzos, G.C. "Multicasting for Multimedia Applications," *Proc. of IEEE Infocom '92*, May 1992, Florence, Italy.

[2] Kompella, V.P., Pasquale, J.C., and Polyzos, G.C. "Multicast Routing Algorithms for Multimedia Communication," submitted for publication to the *IEEE Transactions on Communications.*

I/O Performance of Scientific Applications on Supercomputers

Daniel Kraiman

My current research focuses on applying performance measurement methodology to better understand supercomputer and parallel computer I/O. With the development of faster CPUs in both supercomputers and parallel computers, I/O bandwidth will increasingly become a limiting factor on the actual speed of computation. Also these computers are are being used more and more frequently for I/O-intensive applications, such as visualization. One of the main sources of these increased I/O demands are scientific applications. Thus by investigating the I/O behavior of these applications and the I/O demands they make on the system, we can gain a general understanding of supercomputer I/O.

I have examined the I/O behavior of one such scientific application, an Ocean Global Circulation Model (OGCM) running on the Cray Y-MP8/864 at the San Diego Supercomputer Center . This is a Fortran program developed at NOAA/Princeton. It uses a 3-Dimensional grid and 9 depth levels to simulate ocean basins.

The I/O performance of this program was measured by looking at two different sources. The first was timestamped traces of file operations (write, open, close, and seek) gathered by instrumenting the appropriate I/O system library calls. These trace files were then analyzed using C programs and the S (AT&T) statistical package to produce aggregate I/O Statistics and graphs of the program's I/O rate over time. The second source was looking at the Process Account records for the individual programs using the Cray System Accounting (CSA) utilities.

By examining the aggregate I/O statistics we obtained the following profile of this application's I/O behavior. The program does around 1500 I/O operations and transfers about 280 MB of data. Most of the I/O in terms of both operations (97%) and bytes transferred (99%) is done by write operations. A majority of the individual I/O operations transferred large chunks of data, especially for writes. In addition, most of the I/O operations were to a small group of files. 99% of the bytes read were from a single file, bathymetry, which contains bathymetry data. About 96% of the write operations and 99% of the bytes written were to two files, orestart.0000 and osfcflx. The former file is used for checkpointing, which saves current data values for restart purposes. The latter contains results regarding ocean flux data which is used to interface to an atmospheric model (AGCM).

If we look at the I/O rate of OGCM over both CPU and Wall-Clock Times we find that a pattern emerges. Most of the reading comes in a single .6 MB burst during the first second of program execution. This represents the reading of the bathymetry data. The writing of the checkpoint data comes in 6 45 MB bursts, spaced about 6 CPU seconds apart, except for the final one, which immediately follows the 5th one. This is because the program ends shortly after the 5th burst and an additional checkpoint step is done at the end of program execution. The writing of ocean flux data is also cyclical, coming in 5 1.8 MB bursts, spaced at intervals of 6 CPU seconds. Thus the peak I/O rates occur at equally spaced intervals. These results confirm the findings of Miller and Katz [1], who looked at the I/O behavior of several different scientific applications on a Cray Y-MP at NASA/Ames and found that the patterns were also bursty and cyclical, with request rate peaks at evenly spaced intervals. They also found that there were three major types of I/O: required (initial and final results), checkpoint, and data staging. While OGCM has the first two types of I/O, it does not do data staging (paging large data sets between disk and memory).

My immediate future research will be to extend this work to look at other scientific applications to see if they have similar I/O behavior patterns. It is important to find out what kind of I/O dominates other applications: Do they also do a lot of checkpointing, or do some do a lot of data staging I/O? We are currently undertaking a workload characterization study of the Cray Y-MP to identify other I/O intensive scientific applications. Once we do so we plan to obtain these codes and measure their I/O performance. Another source of scientific applications is the Sequoia project here and at other UC campuses. We are working jointly with scientists in other fields such as oceanography, chemistry, and biology, who run scientific applications on supercomputers.

By analyzing the I/O behavior of many different applications we will be able to come up with a set of canonical I/O models for them. We will thus be able to divide supercomputer scientific applications into several classes and predict their I/O behavior using their class type and a set of parameters which determine such characteristics as the number of iterations and time interval between checkpoints. These models can then be used by systems programmers to formulate I/O scheduling algorithms for supercomputer operating systems, which will improve the I/O performance of supercomputers.

References

[1] Miller, E., and Katz, R. Input/Output Behavior of Supercomputing Applications. *Proceedings Supercomputing*, '91, 1991.

Prioritized Network Traffic in Accommodating Variable Bit Rate Digital Video

Scott McMullan

In transporting real-time digital video and audio over general purpose packet switched networks, problems of insufficient throughput and highly variable/unbounded delay arise. Because digital video and audio require high throughputs and low delay/delay variances, protocol mechanisms must provide levels of assurance that client applications will receive adequate service in the face of competing traffic.

One simple approach is to provide statistical guarantees with a given confidence and support these guarantees by prioritizing traffic. This approach is a particularly attractive solution for accommodating variable bit rate video encoding schemes. These schemes break down frames into independent data segments, each segment contributing to a percentage of the frame's resolution and resulting quality. When segments are later recombined, a full video frame results. The idea is to give higher priority to the segments which contribute to the bulk of the image, while segments which only act to increase resolution and detail are given lower priority. Even if these lower priority segments are dropped due to delay, the continuity of the image is maintained. This scheme has advantages of simplicity in protocol implementation and the ability to tune image quality by changing the proportion of frames with higher priority.

A simple priority scheme using multiple FIFO priority queues was simulated using the Sequoia 2000 topology. The simulation indicated improved delay characteristics for video sources in an environment of interactive, file-transfer, and variable rate hierarchically encoded video sources. The simulation substantiated the claim that such a priority scheme is only useful in providing statistical guarantees, where extreme network loading necessitates these guarantees be relatively weak.

Current and future work consists of implementing the priority scheme in the Sequoia 2000 network and testing the limits and usefulness of such guarantees in supporting video applications. The idea is to use both the simulator and physical implementation to synergistically explore the merits and problems of priority schemes.

Real-Time Traffic Transport via Packet-Switched Networks

Hanrijanto Sariowan

I am investigating various schemes to make packet-switched networks a viable media for transporting real-time traffic, such as the traffic generated by digital-continuous media. All of these schemes have a common goal of reducing delays in the networks. Packet-transit delays can be reduced by short packet lengths or by virtual cut-through switching which allows a packet to be forwarded before it is stored entirely. Queueing delays can be reduced by giving higher priorities to real-time packets than to data packets. Another relatively-recent scheme is embedded or hierarchical coding which encodes real-time data into a set of prioritized components. Switching nodes will be allowed to discard lower-priority components when the networks become heavily congested, resulting in graceful degradation in the quality of the received real-time data. I will use mathematical analysis and simulation methods to examine the effectiveness of these schemes.

As an initial effort, I am presently analyzing a packet multiplexer which accepts two deterministic streams with equal interarrival times but having a phase difference, and different general service distributions. I hope to use this model to predict the performance of a statistical multiplexer which accepts packets of real-time data previously encoded using embedded coding.

At the later stage, this model of statistical multiplexer can be used to represent a packet source in a network in which all nodes may discard lower-priority components of real-time data during congestion. Additional models for intermediate nodes as well as for a packet sink will be developed, and end-to-end performance analysis will be carried out.

Extending FDDI Synchronous Mode Delay Guarantees to Interconnected Rings

Riadh Ben Taarit

The Fiber Distributed Data Interface (FDDI) is becoming the preferred high- speed Local Area Network (LAN). With a data rate of 100 Mb/s (for each of the two counter rotating rings) and support for both asynchronous and synchronous communication modes, it can be used not only as a campus backbone, but also for real-time multimedia communications. An important FDDI feature is the capability to provide performance guarantees for the synchronous traffic in the form of bounded delay packet transport. Furthermore, these performance guarantees are provided without the need for static resource allocation, making the scheme flexible and efficient for variable bit-rate coded continuous media.

Since FDDI is becoming the most popular high-speed LAN, interconnecting FDDI rings becomes both interesting and important. The non trivial aspect of such an interconnection design is how to provide performance guarantees analogous to those offered by a single ring. My goal is to design a scheme that would use the exact same mechanism used on a single ring to negotiate parameters of ring and bridge operation that would guarantee the desired upper bound on packet delay end-to-end (assuming this bound is feasible).

A key FDDI parameter for providing the delay guarantees is the Target Token Rotation Time (TTRT). My approach involves coordinating the adjustment of TTRT across rings whenever this is necessary. On a single FDDI ring, TTRT is set by considering the requirements of all stations on the ring through negotiation. In the case of interconnected rings there is the additional complication of bounding the delay across bridges, as well as, the link delay. This can be achieved by making sure during the negotiation process that the total synchronous traffic through the bridges and the interconnecting links is well within their capacity and that synchronous traffic has priority over asynchronous traffic.

In order to have a concrete and flexible testbed for investigating the various alternative schemes, I have implemented a discrete event simulator for interconnected FDDI rings. The simulator permits detailed investigations of throughput and delay performance for both synchronous and asynchronous traffic, as well as, the analysis of the effect of link capacity and propagation delay to feasible delay guarantees.

9. CSL Publications

[PMV87] Polyzos, G.C., Molle, M.L., and Venetsanopoulos, A.N., "Performance Analysis of Finite Non-homogeneous Population Tree Conflict Resolution Algorithms Using Constant Size Window Access," *IEEE Transactions on Communications*, Vol. COM-35, No. 11, pp. 1124-1138, November 1987.

[P88] Pasquale, J., "Using Expert Systems for Managing Distributed Systems," *IEEE Network*, Vol. 2, No. 5, Sept. 1988.

[P89] Pasquale, J., "The Effects of Using Aging State Information in Decentralized Load Balancing," Technical Report CS89-161, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, October 1989.

[P89b] Pasquale, J., "Decentralized Resource Management Research in Large Distributed Computer Systems," Technical Report CS89-162, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1989.

[PM90] Polyzos, G. C. and Molle, M. L. "Conflict Resolution Algorithms for High Error-rate Multi-access Channels," *IEEE International Symposium on Information Theory*, San Diego, CA, January 1990.

[P90] Pasquale, J., "Beyond Client Server Architectures," *NCR Perspective*, Vol. 4, No. 2, Spring 1990.

[P+90] Pasquale, J., Anderson, E., Bubien, M., Fall, K, Kompella, V., McMullan, S., Ranganathan, D., and Terek, R. "Operating System and Window System Research for Multi-media Applications: A Status Report," Technical Report CS90-176, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, July 1990.

[PM90a] Polyzos, G. C. and Molle, M. L. "Delay Analysis for the FCFS 0.487 Conflict Resolution Algorithm," Technical Report CS90-179, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, October 1990.

[PP90] Pasquale, J. and Polyzos, G. C. "System Support for Multimedia Applications," *Proc. First International Workshop on Network and Operating System Support for Digital Audio and Video*, International Computer Science Institute, Berkeley, CA, November 8-9, 1990.

[M90] Muller, K. "Multi-structured File System Design for High Performance Computing," Ph.D. Thesis, Department of Computer Science and Engineering, University of California, San Diego, December 1990.

[G90] Glockner, A. "Coadaptive Decision-making in Distributed Computer Systems," Ph.D. Thesis, Department of Computer Science and Engineering, University of California, San Diego, December 1990.

[P+91] Pasquale, J., Polyzos, G. C., Anderson, E. W., Fall, K. R., Kay, J. S. , Kompella, V. P., McMullan, S. R., and Ranganathan, D. "Network and Operating System Support for Multimedia Applications," Technical Report CS91-186, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, March 1991.

[KPP91] Kompella, V. P., Pasquale, J. C., and Polyzos, G. C., "Multicast Routing for High-Bandwidth Delay-Sensitive Applications," Technical Report CS91-185, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, March 1991.

[PBK91] Pasquale, J., Bittel, B., and Kraiman, D. "A Workload Characterization Study of a Cray XMP Running CTSS," Technical Report CS91-222, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, April 1991.

[PBK91b] Pasquale, J., Bittel, B., and Kraiman, D., "A Static and Dynamic Workload Characterization of the SDSC Cray XMP," *1991 ACM SIGMETRICS Proceedings* (extended abstract), San Diego, May 1991.

[P91] Pasquale, J., "Decentralized Control in Large Distributed Systems," *Coordination Theory and Collaboration Technology Workshop Proceedings*, National Science Foundation, Washington D.C., June 1991, pp. 9-20.

[TP91] Terek, R., and Pasquale, J., "Experiences with Audio Conferencing Using X, UNIX, and TCP/IP over an Ethernet," *1991 Summer Usenix Proceedings*, Nashville, June 1991.

[MP91] Muller K., and Pasquale, J., "A High-performance Multi-structured File System Design," *1991 ACM Symposium on Operating System Principles (SOSP) Proceedings*, Asilomar, CA, October 1991.

[P+91b] Pasquale, J., Polyzos, G., Fall, K., Kompella, V. "Internet Throughput and Delay Measurements Between Sequoia 2000 Sites," Technical Report CS91-219, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1991.

[FPP92] Ferrari, D., J.C. Pasquale, and Polyzos, G.C., "Network Issues for Sequoia 2000," *Proc. IEEE Compcon Spring '92*, San Francisco, CA, February 24-28, 1992.

[CP92] Claffy K., and Polyzos, G.C., "Location Transparent Connection Management: A Survey of Protocol Issues," *Proc. 11th Annual IEEE International Phoenix Conference on Computers and Communications*, Scottsdale, AZ, April 1-3, 1992.

[P92] Polyzos, G.C., "Networking for Sequoia 2000," *Proc. 4th IEEE COMSOC International Workshop on Multimedia Communications*, MULTIMEDIA '92, Monterey, CA, April 1-4, 1992.

[KPP92] Karvelas, D., Papamichail, M., and Polyzos, G. "The Rotating Slot Generator Dual Bus," *Proc. of the 23rd Pittsburgh Conference on Modeling and Simulation*, Pittsburgh, PA, April 1992.

[B+92] Braun, H.W., Claffy, K., Chinoy, B., and Polyzos, G.C. "Analysis and Modeling Tools for High-Speed Networks: Project Status Report," Technical Report CS92-237, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, April 1992.

[KP92] Kay, J., and Pasquale, J., "Network Sloth: Where the Time Goes in Network Software," Technical Report CS92-238, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, April 1992.

[P92b] Pasquale, J., "I/O System Design for Intensive I/O," *IEEE Workshop on Workstation Operation Systems (WWOS) Proceedings*, Key Biscayne, FL, April 1992.

[KPP92b] Kompella, V.P., Pasquale, J.C., and Polyzos, G.C., "Multicasting for Multimedia Applications," *Proc. IEEE INFOCOM '92*, Florence, Italy, May 4-8, 1992.

[KPP92c] Karvelas, D., Papamichail, M., and Polyzos, G.C., "Performance Analysis of the Rotating Slot Generator Scheme," *Proc. IEEE INFOCOM '92*, Florence, Italy, May 4-8, 1992.

[GP92] Glockner A., and Pasquale, J., "Co-adaptive Decision-making in Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, accepted for publication.