Department of Computer Science

THE UNIVERSITY OF
ARIZONA®
TUCSON ARIZONA

Gould-Simpson Building
Tucson, Arizona 85721

RL-34

January - December 1994

# Technical Report List

1994

Abstracts of technical reports available from our department are listed below. If you would like to receive a copy of any of these documents, return the form at the end of this list.

TR 94-01 **Towards an Infrastructure for Temporal Databases: Report of an Invitational ARPA/NSF Workshop**

Niki Pissinou, Richard Snodgrass, Ramez Elmasri, Inderpal Mumick, M. Tamer Özsu, Barbara Pernici, Arie Segev, and Babis Theodoulidis

On June 14-16, 1993, the International Workshop on an Infrastructure for Temporal Databases was held in Arlington, TX. Forty-four prominent researchers from ten countries and from ten industrial labs and companies and from many universities participated in this workshop. Much of the workshop comprised intensive meetings of four working groups. This is a summary of the discussions of the working groups.

Common themes ran through the discussions of all four groups. The infrastructure must be based on a base set of desired features, so that most temporal applications receive at least some support from the temporal DBMS. Terminology is critical. Aspects of the conceptual model must be separated from concerns of the representation. The baseline architecture must be extensible, and should identify what is different about a temporal DBMS, and what can vary between TDBMS implementations. (54 pages)

TR 94-02 **Parallel Logic Simulation of VLSI Systems**

Mary L. Bailey, Jack V. Briner, Jr., and Roger D. Chamberlain

Fast, efficient logic simulators are an essential tool in modern VLSI system design. Logic simulation is used extensively for design verification prior to fabrication, and as VLSI systems grow in size, the execution time required by simulation is becoming more and more significant. Faster logic simulators will have an appreciable economic impact, speeding time to market while ensuring more thorough system design testing. One approach to this problem is to utilize parallel processing, taking advantage of the concurrency available in the VLSI system to accelerate the logic simulation task.

Parallel logic simulation has received a great deal of attention over the past several years, but this work has not yet resulted in effective, high-performance simulators being available to VLSI designers. A number of techniques have been developed to investigate performance issues: formal models, performance modeling, empirical studies, and prototype implementations. Analyzing reported results of these techniques, we conclude that five major factors affect performance: synchronization algorithm, circuit structure, timing granularity, target architecture, and partitioning. After reviewing techniques for parallel simulation, we consider each of these factors using results reported in the literature. Finally we synthesize the results and present directions for future research in the field. (65 pages)

TR 94-03 **Output Value Placement in Moded Logic Programs**

Peter Bigot, Saumya Debray, and Dave Gudeman

Most implementations of logic programming languages treat input and output arguments to procedures in a fundamentally asymmetric way: input values are passed in registers, but output values are returned in memory. In some cases, placing the outputs in memory is useful to preserve the opportunity for tail call optimization. In other cases, this asymmetry can lead to a large number of

unnecessary memory references and adversely affect performance. When input/ output modes for arguments are known it is often possible to avoid much of this unnecessary memory traffic via a form of interprocedural register allocation. In this paper we discuss how this problem may be addressed by returning output values in registers where it seems profitable to do so. The techniques described here have been implemented in the jc system, but are also applicable to other moded logic programming languages, such as Parlog, as well as languages like Prolog when input/output modes have been determined via dataflow analysis. (30 pages)

TR 94-04 **TCP Vegas: New Techniques for Congestion Detection and Avoidance**

Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson

Vegas is a new implementation of TCP that achieves between 40 and 70% better throughput, with one-half to one-fifth the losses, as compared to the implementation of TCP in the Reno distribution of BSD Unix. This paper motivates and describes the five key techniques employed by Vegas, and presents the results of a comprehensive experimental performance study—using both simulations and measurements of the actual Internet—of the Vegas and Reno implementations of TCP. (18 pages)

TR 94-05 **Experiences with a High-Speed Network Adaptor: A Software Perspective**

Peter Druschel, Larry L. Peterson, and Bruce S. Davie

This paper describes our experiences, from a software perspective, with the Osiris network adaptor. Osiris is an interesting case to study because of the flexibility it provides in defining the host/adaptor software interface. The paper makes two contributions. First, it identifies the problems we encountered while programming Osiris and optimizing network performance, and outlines how we either addressed them in the software, or had to modify the hardware. Second, it describes the opportunities provided by Osiris that we were able to exploit in the host operating system (OS); opportunities that suggested techniques for making the OS more effective in delivering network data to application programs. The most novel of these techniques, called *application device channels*, gives application programs running in user space direct access to the adaptor. The paper concludes with the lessons drawn from this work, which we believe will benefit the designers of future network adaptors. (18 pages)

TR 94-06 **Temporal Granularity and Indeterminacy: Two Sides of the Same Coin**

Curtis Dyreson and Richard T. Snodgrass

*Granularity* is an integral feature of all temporal data. For instance, a person's age is commonly given to the granularity of *years* and the time of their next airline flight to the granularity of *minutes*. *Indeterminacy*, or don't know when information, is a companion to granularity. A birthday, given to the granularity of days, does not reveal *precisely* when a person was born, only that they were born *sometime* during the indicated day. We define a granularity as a calendar-dependent partitioning of the time-line. Pairs of granularities are related in that some granularities are finer or coarser with respect to other granularities. A *granularity lattice* records all the relationships between granularities, even among granularities in different calendars. A time at a given granularity is indeterminate at all finer granularities. We extend the syntax and semantics of SQL-92 to support mixed granularities. This support rests on two operations, *scale* and *align*, that move times within the granularity lattice, e.g., from days to months. We demonstrate that our solution is practical by outlining an efficient implementation. The implementation uses several optimization strategies to mitigate the expense of accommodating multiple granularities. (22 pages)

TR 94-07 **A Fast and General Software Solution to Mutual Exclusion on Uniprocessors**

David Mosberger, Peter Druschel, and Larry L. Peterson

This paper presents a technique to solve the mutual exclusion problem for uniprocessors purely in software. The idea is to execute atomic sequences without any hardware protection and, in the rare case that the atomic sequence is interrupted, to roll forward to the end of the sequence. The main contribution of this paper is to discuss the OS-related issues of this technique and to demonstrate its practicality, both in terms of flexibility and performance. It proposes a purely software-based technique that achieves mutual exclusion *without any memory-accesses*. Experiments show that this technique has the potential to outperform equivalent hardware mechanisms. (17 pages)

TR 94-08 **Cache and TLB Effectiveness in Processing Network I/O**

Michael A. Pagels Peter Druschel, and Larry Peterson

This paper considers the question of how effective caches are in processing network I/O. Our analysis shows that operating system structure plays a key role in the caches behavior, with BSD Unix (a monolithic OS) making more effective use of the cache than Mach (a microkernel OS). Moreover, closer inspection shows that several factors contribute to this result, including how TLBs are managed, how scheduling points are interspersed with data accesses, how data is laid out in memory, and how network functionality is distributed between user space and the kernel. (16 pages)

TR 94-09 **Cluster-C*: Understanding the Performance Limits**

Larry Peterson

Data parallel languages are gaining interest as it becomes clear that they support a wider range of computation than previously believed. With improved network technology, it is now feasible to build data parallel supercomputers using traditional RISC-based workstations connected by a high-speed network. This paper presents an in-depth look at the communication behavior of nine C* programs. It also compares the performance of these programs on both a cluster of 8 HP 720 workstations and a 32 node (128 Vector Unit) CM-5. The result is that under some conditions, the cluster is faster on an absolute scale, and that on a relative, per-node scale, the cluster delivers superior performance in all cases. (14 pages)

TR 94-10 **USC: A Universal Stub Compiler**

Sean O'Malley, Todd Proebsting, and Allen Brady Montz

Abstract USC is a new stub compiler which can be used to generate stubs which perform a wide variety of data conversion operations. USC is flexible and can be used in situations were previously only manually code generation was possible. USC generated code is up to 20 times faster than code generated by traditional argument marshaling schemes such as ASN.1 and Sun XDR. This paper presents the design of USC and a comprehensive set of experiments designed to compare USC performance with the best manually generated code and traditional stub compilers. (14 pages)

TR 94-11 **Distributed Filaments: Efficient Fine-Grain Parallelism on a Cluster of Workstations**

Vincent W. Freeh, David K. Lowenthal, and Gregory R. Andrews

A fine-grain parallel program is one in which processes are typically small, ranging from a few to a few hundred instructions. Fine-grain parallelism arises naturally in many situations, such as iterative grid computations, recursive fork/join programs, the bodies of parallel FOR loops, and the implicit parallelism in functional or dataflow languages. It is useful both to describe massively parallel computations and as a target for compilers. However, fine-grain parallelism has long been thought to be inefficient due to the overheads of process creation, context switching, and synchronization. This paper describes a software kernel, Distributed Filaments (DF), that implements fine-grain parallelism both portably and efficiently on a workstation cluster. DF runs on existing, off-the-shelf hardware and software. DF has a simple interface, with only seven routines, so it is easy to use. It achieves efficiency by using stateless threads on each node, overlapping communication and computation, employing a new low-overhead, reliable datagram communication protocol, and automatically balancing the work generated by fork/join computations. The net result is good speedup on a variety of applications. (16 pages)

TR 94-12 **Computing Temporal Aggregates**

Nick Kline and Richard T. Snodgrass

Aggregate computation, such as selecting the minimum attribute value of a relation, is expensive, especially in a temporal database. We describe the basic techniques behind computing aggregates in conventional databases and show that these techniques are not efficient when applied to temporal databases. We examine the problem of computing constant intervals (intervals of time where the aggregate value is constant), which are needed for temporal grouping. We introduce several algorithms for computing temporal aggregates: the aggregation tree, the chalk board, and the *k*-ordered aggregation tree algorithms. An empirical comparison demonstrates that the choice of algorithm depends in part on the amount of memory available, the number of tuples in the underlying relation, and the degree to which the tuples are ordered. (21 pages)

TR 94-13 **Scheduling Independent Tasks to Minimize the Makespan on Identical Machines**

John Bruno, Edward G. Coffman Jr., and Peter Downey

In this paper we consider scheduling *n* tasks with task times that are i.i.d. random variables with a common distribution function *F* on *m* parallel machines. Scheduling is done by an a priori assignment of tasks to processors. We show that if the distribution function *F* is a Polya frequency function of order 2 then the assignment which attempts to place an equal number of tasks on each processor achieves the stochastically smallest makespan among all assignments. The condition embraces many important distributions, such as the gamma and truncated normal distributions. (14 pages)

TR 94-14 **Exploiting Highly Reliable Networks with Careful Protocols**

David Mosberger, Charles J. Turner, and Larry L. Peterson

Optical interconnects often exhibit excellent reliability. This paper studies the potential for exploiting this reliability via network protocols that no longer assume unreliable links. Such protocols are termed ‘‘careful protocols’’ as they have to be careful to preserve the reliability of the link to the application level. After giving a definition of careful protocols and discussing related work, the paper presents in detail a case study of a networking service that has been implemented, both as a traditional and as a careful protocol. The case study suggests that careful protocols can achieve considerably better performance than traditional ones but also shows that this is not trivial to achieve. (27 pages)

TR 94-15 **Fast Performance Models for Conservative Asynchronous Protocols Using Busses**

(formerly titled "Towards "On the Fly" Performance Models for Conservative Asynchronous Protocols")

Mary L. Bailey and Shane Walker

In the parallel simulation community there are two common synchronization strategies: conservative and optimistic. Formal models exist for estimating the performance of optimistic strategies and conservative synchronous strategies, but few models predict the performance of conservative asynchronous strategies. We are interested in not only creating a performance model for conservative asynchronous simulation, but a model that is fast, so that "on the fly" estimates can be made to assist users in determining whether the conservative strategy is appropriate for their application.

Towards this goal, we present in this paper a model for predicting the cost of communications in conservative asynchronous simulations that use a specific type of communication, a bus structure. Not only is this a useful communication structure in practice, but it places some constraints on communications which make it more amenable for "on the fly" characterizations. Overall, the model shows promise when evaluated using test cases not involved in the model building. Most of the characterizations can be computed using simple formulas parameterized with variables obtained from the input program. A few characterizations are more complex, but these can be estimated by applying a sequence of simple functions. (25 pages)

TR 94-16 **Paving the Road to Network Security or the Value of Small Cobblestones**

Hilarie Orman, Sean O’Malley, Richard Schroeppel, and David Schwartz

Software subsystems that implement cryptographic security features can be built from small modules using uniform interfaces. The methods demonstrated in this paper illustrate how configuration flexibility can be achieved and how complex services can be constructed, all using the same building block modules. These allow the configuration process to be independent of algorithm details, while the algorithms used in the subsystem are obvious. (17 pages)

TR 94-17 **A Fast Algorithm for Multi-Pattern Searching**

Sun Wu and Udi Manber

A new algorithm to search for multiple patterns at the same time is presented. The algorithm is faster than previous algorithms and can support a very large number - tens of thousands - of patterns. Several applications of the multi-pattern matching problem are discussed. We argue that, in addition to previous applications that required such a search, multi-pattern matching can be used in lieu of indexed or sorted data in some applications involving small to medium size datasets. Its advantage, of course, is that no additional search structure is needed. (11 pages)

TR 94-18 **The Ratio of the Extreme to the Sum in a Random Sequence with Applications**

Peter J. Downey and Paul E. Wright

If $X_1, X_2, \cdots, X_n$ is a sequence of non-negative independent random variables with common distribution function $F(t)$, we write $X_{(n)}$ for the maximum of the sequence and $S_n$ for its sum. The ratio variate $R_n = X_{(n)}/S_n$ is a quantity arising in the analysis of process speedup and the performance of scheduling. O'Brien (1980) showed that $R_n \to 0$ almost surely as $n \to \infty$ if and only if $\mathbf{E}X_1 < \infty$. Since $\{R_n\}$ is a uniformly bounded sequence it follows that $\mathbf{E}X_1 < \infty$ implies $\mathbf{E}R_n \to 0$ as $n \to \infty$.

Here we show that, provided either that (i) $\mathbf{E}X_1^2 < \infty$ or that (ii) $1 - F(t)$ is a regularly varying function with index $\rho < -1$, it follows that

$$\mathbf{E}R_n = \frac{\mathbf{E}X_{(n)}}{\mathbf{E}S_n}\Big(1 + o(1)\Big) \qquad (n \to \infty) .$$

Since the asymptotics of $\mathbf{E}X_{(n)}$ is often readily calculated, this prov ides a useful estimate for the most significant behavior of the ratio $R_n$ in expectation. We apply this result to multiprocessor scheduling and to the behavior of sample statistics. (24 pages)

TR 94-19 **Interrupt Protocol Processing in the x-kernel**

Mats Bjorkman

On many modern processors, context switches are costly in terms of latency. In this report, we argue that latency caused by kernel thread scheduling for incoming network messages can be avoided in a number of common situations, where instead the interrupt handler itself can perform the necessary protocol processing. For processors with high context switch costs, this can give a significant performance improvement. To support our argumentation, we present an implementation of the x-kernel communication protocol execution environment that exploits this execution paradigm. We present measurements on DECstations and on the Intel Paragon, showing the performance gain to expect when performing protocol processing in the interrupt handler. (12 pages)

TR 94-20 **Scout: A Communications-Oriented Operating System**

Allen B. Montz, David Mosberger, Sean W. O'Malley, Larry L. Peterson, Todd A. Proebsting, and John H. Hartman

This white paper describes Scout, a new operating system being designed for systems connected to the National Information Infrastructure (NII). Scout provides a communication-oriented software architecture for building operating system code that is specialized for the different systems that we expect to be available on the NII. It includes an explicit path abstraction that both facilitates effective resource management and permits optimizations of the critical path that I/O data follows. These path-enabled optimizations, along with the application of advanced compiler techniques, result in a system that has both predictable and scalable performance. (18 pages)

TR 94-21 **Network Systems Research Group: An Annotated Bibliography**

Larry Peterson

This paper gives a brief overview of the research done by the Network Systems Research Group over the last three to four years, and references the principle papers published during that time. (6 pages)

TR 94-22 **Approximately Matching Context-Free Languages**

Gene Myers

Given a string $w$ and a pattern $p$, approximate pattern matching merges traditional sequence comparison and pattern matching by asking for the minimum difference between $w$ and a string exactly matched by $p$. We give an $O(PN sup 2 (N+ log ˆP))$ algorithm for approximate matching between a string of length $N$ and a context-free language specified by a grammar of size $P$. The algorithm generalizes the Cocke-Younger-Kasami algorithm for determining membership in a context-free language. We further sketch an $O(P sup 5 N sup 8 8 sup P )$ algorithm for the problem where gap costs are concave, and pose two open problems for such general comparison cost models. (13 pages)

TR 94-23 **Supporting Fault-Tolerant Parallel Programming in Linda**

Dave Edward Bakken

As people are becoming increasingly dependent on computerized systems, the need for these systems to be dependable is also increasing. However, programming dependable systems is diffcult, especially when parallelism is involved. This is due in part to the fact that very few high-level programming languages support both fault-tolerance and parallel programming.

This dissertation addresses this problem by presenting FT-Linda, a high-level language for programming fault-tolerant parallel programs. FT-Linda is based on Linda, a language for programming parallel applications whose most notable feature is a distributed shared memory called tuple space. FT-Linda extends Linda by providing support to allow a program to tolerate failures in the underlying computing platform. The distinguishing features of FT-Linda are stable tuple spaces and atomic execution of multiple tuple space operations. The former is a type of stable storage in which tuple values are guaranteed to persist across failures, while the latter allows collections of tuple operations to be executed in an all-or-nothing fashion despite failures and concurrency. Example FT-Linda programs are given for both dependable systems and parallel applications.

The design and implementation of FT-Linda are presented in detail. The key technique used is the replicated state machine approach to constructing fault-tolerant distributed programs. Here, tuple space is replicated to provide failure resilience, and the replicas are sent a message describing the atomic sequence of tuple space operations to perform. This strategy allows an efficient implementation in which only a single multicast message is needed for each atomic sequence of tuple space operations.

An implementation of FT-Linda for a network of workstations is also described. FT-Linda is being implemented using Consul, a communication substrate that supports fault-tolerant distributed programming. Consul is built in turn with the {it x}-kernel, an operating system kernel that provides support for composing network protocols. Each of the components of the implementation has been built and tested. (173 pages)

TR 94-24 **Operating System Support for High-Speed Networking**

Peter Druschel

The advent of high-speed networks may soon increase the network bandwidth available to workstation class computers by two orders of magnitude. Combined with the dramatic increase in microprocessor speed, these technological advances make possible new kinds of applications, such as multimedia and parallel computing on networks of workstations. At the same time, the operating system, in its role as mediator and multiplexor of computing resources, is threatening to become a bottleneck. The underlying cause is that main memory performance has not kept up with the growth of CPU and I/O speed, thus opening a bandwidth gap between CPU and main memory, while closing the old gap between main memory and I/O. Current operating systems fail to properly take into account the performance characteristics of the memory subsystem. The trend towards server-based operating systems exacerbates this problem, since a modular OS structure tends to increase pressure on the memory system. This dissertation is concerned with the I/O bottleneck in operating systems, with particular focus on high-speed networking. We start by identifying the causes of this bottleneck, which are rooted in a mismatch of operating system behavior with the performance characteristics of modern computer hardware. Then, traditional approaches to supporting I/O in operating systems are re-evaluated in light of current hardware performance tradeoffs. This re-evaluation gives rise to a set of novel techniques that eliminate the I/O bottleneck. (107 pages)

TR 94-25 **Development of an Intelligent Monitoring and Control System for a Heterogeneous Numerical Propulsion System Simulation**

John A. Reed, Abdollah A. Afjeh, University of Toledo, Henry Lewandowski, Cleveland State University, Patrick T. Homer, Richard D. Schlichting, University of Arizona

The NASA Numerical Propulsion System Simulation (NPSS) project is exploring the use of computer simulation to facilitate the design of new jet engines. Several key issues raised in this research are being examined in an NPSS-related research project: zooming, monitoring and control, and support for heterogeneity. The design of a simulation executive that addresses each of these issues is described. In this work, the strategy of zooming, which allows codes that model at different levels of

fidelity to be integrated within a single simulation, is applied to the fan component of a turbofan propulsion system. A prototype monitoring and control system has been designed for this simulation to support experimentation with expert system techniques for active control of the simulation. An interconnection system provides a transparent means of connecting the heterogeneous systems that comprise the prototype. (14 pages)

TR 94-26 **The USC Reference Manual**

Sean O'Malley, Todd Proebsting, Allen Brady Montz, and Dorgival Guedes

USC is a new stub compiler that generates stubs that perform many data conversion operations. USC is flexible and can be used in situations where previously only manual code generation was possible. USC generated code is up to 20 times faster than code generated by traditional argument marshaling schemes such as ASN.1 and Sun XDR. This document is the USC reference manual. (11 pages)

TR 94-27 **Towards Understanding the Relationship Between Pipelining and Circuit Parallelism**

Mary L. Bailey

Pipelining increases the performance of sequential circuits by allowing different parts of the circuit to concurrently work on different input sets. Surprisingly, this increased concurrency fails to translate into increased circuit parallelism, the avergae number of events executed per non-empty time step during a simulations. We show that this *pipeline anomaly* occurs because the non-pipelined circuit performs extraneous computations that are reduced by pipelining. (16 pages)

TR 94-28 **Constructing a Configurable Group RPC Service**

Matti A. Hiltunen and Richard D. Schlichting

Current Remote Procedure Call (RPC) services implement a variety of semantics, with many of the differences related to how communication and server failures are handled. The list increases even more when considering group RPC, a variant of RPC often used for fault-tolerance where an invocation is sent to a group of servers rather than one. This paper presents an approach to constructing group RPC in which a single configurable system is used to build different variants of the service. The approach is based on implementing each property as a separate software module called a micro-protocol, and then configuring the micro-protocols needed to implement the desired service together using a software framework based on the *x*-kernel. The properties of point-to-point and group RPC are identified and classified, and the general execution model described. An example consisting of detailed pseudocode for a modular implementation of a group RPC service is given to illustrate the approach. Dependency issues that restrict configurability are also addressed. (20 pages)

TR 94-29 **Performance Problems in BSD4.4 TCP**

Lawrence S. Brakmo and Larry L. Peterson

This report describes problems in the BSD 4.4-Lite version of TCP (some of which are also present in earlier versions) and proposes fixes that result in a 21% increase in throughput under some conditions. (17 pages)

TR 94-30 **Valid Time Integrity Contraints**

Michael Boehlen

This paper investigates temporal integrity constraints in valid time databases, i.e., databases that capture the time-varying nature of the part of reality being modeled. We first provide a taxonomy of integrity constraints in (temporal) databases in order to establish a common terminology. The taxonomy identifies two classes of valid time integrity constraints: intrastate and interstate integrity constraints. Intrastate integrity constraints result from generalizing nontemporal integrity constraints. They guarantee the consistency of every snapshot of a valid time database. Interstate integrity constraints relate and constrain different valid time snapshots and, therefore, they are unique to the temporal dimension. ChronoLog, a query language based on first order predicate logic, can express both types of integrity constraints. Furthermore, time-restricted integrity constraints may be expressed in ChronoLog. Finally, we discuss the efficient checking of valid time integrity constraints, which is much harder than the checking of transaction time integrity constraints because the user can update all states of a valid time database. (22 pages)

TR 94-31 **On the Semantics of ‘‘Now’’ in Temporal Databases**

James Clifford, Curtis Dyreson, Tomaś Isakowitz, Christian S. Jensen and Richard T. Snodgrass

Most databases record time-varying data, and significant efforts have been devoted to the convenient and efficient management of such data. Perhaps most prominently, numerous data models with varying degrees of built-in support for the temporal dimension of data have been proposed. Some models are quite restricted and simply support uninterpreted attribute domains for times and dates. Other models incorporate either a valid-time dimension, recording when the stored data is true, or a transaction-time dimension, recording when the stored data is current in the database. Bitemporal data models incorporate both valid and transaction time. The special temporal notion of an ever-increasing current-time value has been reflected in some of these data models by inclusion of current-time variables, such as now, until-changed, ∞, @ and –. As timestamp values associated with facts in temporal databases, such variables may be conveniently used for indicating that a fact is currently valid. Although the services of time variables are very desirable, their use leads to a new type of database, consisting of tuples with variables, termed variable databases.

This paper proposes a framework for defining the semantics of the variable databases of temporal relational data models. A framework is presented because several reasonable meanings may be given to databases that use some of the specific temporal variables that have appeared in the literature. Using the framework, the paper defines a useful semantics for such databases. Because situations occur where the existing time variables are inadequate, two new types of modeling entities that address these shortcomings, timestamps which we call now-relative and now-relative indeterminate, are introduced and defined within the framework. Moreover, the paper provides a foundation, using algebraic bind operators, for the querying of variable databases via existing query languages. This transition to variable databases presented here requires minimal change to the query processor. Finally, to underline the practical feasibility of variable databases, we show that variables may be represented and manipulated efficiently, incurring little space or execution time overhead. (49 pages)

TR 94-32 **Monitoring and Controlling Remote Parallel Computations Using Schooner**

Zhanliang Chen and Richard D. Schlichting

Scientific visualization systems such as AVS have the potential to help users of parallel systems monitor and control their computations. Unfortunately, the machines most suitable for visualization systems are not the parallel systems on which the computation executes, often leading to the use of two distinct machines and the viewing of results only after the computation has completed. Here, an approach to solving this problem is presented in which AVS and a remote parallel computation are incorporated into a single metacomputation using the Schooner software interconnection system. This scheme gives the user enhanced control, including the ability to dynamically select the parallel platform to be used, monitor the progress of the computation, and modify parameters. An example application is described in which AVS and a parallel neural net code executing on either an Intel Paragon, Sequent Symmetry, or PVM-based Sun Sparcstation cluster are interconnected. These experiments demonstrate not just the feasibility of structuring parallel computations as part of a larger metacomputation using Schooner, but also that these benefits can be achieved with only modest cost. (18 pages)

TR 94-33 **Constructing Scientific Applications from Heterogeneous Resources**

Patrick T. Homer

The computer simulation of scientific processes is playing an increasingly important role in scientific research. For example, the development of adequate flight simulation environments, numeric wind tunnels, and numeric propulsion systems is reducing the danger and expense involved in prototyping new aircraft and engine designs. One serious problem that hinders the full realization of the potential of scientific simulation is the lack of tools and techniques for dealing with the heterogeneity inherent in today’s computational resources and applications. Typically, either ad hoc connection techniques, such as manual file transfer between machines, or approximation techniques, such as boundary value equations, are employed.

This dissertation develops a programming model in which scientific applications are designed as

heterogeneous distributed programs, or meta-computations. The central feature of the model is an interconnection system that handles the transfer of control and data among the heterogeneous components of the meta-computation, and provides configuration tools to assist the user in starting and controlling the distributed computation. Key benefits of this programming model include the ability to simulate the interactions among the physical processes being modeled through the free exchange of data between computational components. Another benefit is the possibility of improved user interaction with the meta-computation, allowing the monitoring of intermediate results during long simulations and the ability to steer the simulation, either directly by the user or through the incorporation of an expert system into the meta-computation.

This dissertation describes a specific realization of this model in the Schooner interconnection system, and its use in the construction of a number of scientific meta-computations. Schooner uses a type specification language and an application-level remote procedure call mechanism to ease the task of the scientific programmer in building meta-computations. It also provides static and dynamic configuration management features that support the creation of meta-computations from components at runtime, and their modification during execution. Meta-computations constructed using Schooner include examples involving molecular dynamics and neural nets. Schooner is also in use in several major projects as part of a NASA effort to develop improved jet engine simulations. (145 pages)

TR 94-34 **A Simple Scheme to Make Passwords Based on One-Way Functions Much Harder to Crack**

Udi Manber

We present a simple scheme that makes guessing passwords based on one-way functions 100 to 1000 times harder. The scheme is easy to program and easy to incrementally add to existing schemes. In particular, there is no need to switch to it all at the same time. Old passwords will still work and have the same security as before (one will not be able to distinguish them from new passwords); newly-entered passwords will become much more secure. The new scheme is independent of the one-way function used and does not require changing any part of the encryption mechanism. (6 pages)

TR 94-35 **Adaptive Data Placement for Distributed-Memory Machines**

David K. Lowenthal and Gregory R. Andrews

Programming distributed-memory machines requires paying careful attention to where the data is placed. This is because for efficiency, it is important to balance the computational load among the nodes and to minimize excess movement of data between the nodes during the computation. Most current approaches to data placement are static, requiring either the programmer or compiler to explicitly place or move the data. This paper describes a new adaptive approach. It is implemented in the Adapt system, which finds the best data placement at run time. When necessary, Adapt automatically changes the data placement in mid-application, so that it adapts to the needs of the application. Adapt can be used to simplify both programming in parallel languages such as HPF and in compilers for these languages. We test the performance of Adapt on three applications: Jacobi iteration, LU decomposition, and Dynamic Jacobi, which mimics the behavior of applications such as adaptive mesh refinement. Using Adapt, the first two attain performance very close to that of programs using the best static data placement; the third application cannot be analyzed statically and ran faster using Adapt than with any static data placement. (13 pages)

TR 94-36 **Identifying Features in Biological Sequences: Fifth Workshop Report (Aspen Center For Physics, May 30 - June 19, 1994)**

Gene Myers, Christian Burks, and Gary Stormo

This report is for the fifth of an annual series of workshops held at the Aspen Center for Physics concentrating on the identification of features in DNA sequence, and more broadly on related topics in computational molecular biology. Over the last five years the workshop has been cited as supporting and/or inspiring over 40 papers, has provided training and inspiration to Ph.D students and post-doctoral fellows early in their careers, and provided senior scientists the unique opportunity to seriously engage in interdisciplinary collaborations over the 3 weeks of the workshop. (8 pages)

TR 94-37 **A Configurable Membership Service**

Matti A. Hiltunen and Richard D. Schlichting

A membership service is used to maintain information about which processes are functioning in a distributed system at any given time. Many such services have been defined, with each implementing a unique combination of properties that simplify the construction of higher levels of the system. Despite this wealth of possibilities, however, any given service only realizes one set of properties, which makes it difficult to tailor the service provided to the specific needs of the application. Here, a configurable membership service that addresses this problem is described. This service is based on decomposing membership into its constituent abstract properties, and then implementing these properties as separate modules that can be configured together to produce a customized membership service. In addition to supporting flexible configurations, this membership service allows new properties to be added or existing properties redefined if desired. The constituent properties of membership are first surveyed, followed by a description of an implementation strategy. This strategy is based on building modules as individual *micro-protocols* and then configuring these micro-protocols into a standard event-driven software framework to form a *composite protocol* realizing the desired service. An *x*-kernel based implementation of this scheme is underway. (16 pages)

Technical Reports are available via anonymous FTP from cs.arizona.edu in the **reports** directory or via electronic mail by sending a message to ftpmail@cs.arizona.edu containing the word **help**. Questions may be directed to tr_libr@cs.arizona.edu.

Because of the costs of printing and mailing, only one free hard copy of each report will be sent to an address. Additional copies may be purchased for the amount indicated. Requests that require payment must be accompanied by a check or a prepaid purchase order in U.S. dollars for the proper amount payable to The University of Arizona. Free copies may also be ordered via electronic mail to tr_libr@cs.arizona.edu.

Please send me the reports checked below: (RL-35)

| *report* | *additional copies* | | |
|---|---|---|---|
| ☐ TR 94-24 | 5.50 | ☐ TR 94-31 | 3.50 |
| ☐ TR 94-25 | 2.00 | ☐ TR 94-32 | 2.00 |
| ☐ TR 94-26 | 2.00 | ☐ TR 94-33 | 6.50 |
| ☐ TR 94-27 | 2.00 | ☐ TR 94-34 | 1.50 |
| ☐ TR 94-28 | 2.00 | ☐ TR 94-35 | 2.00 |
| ☐ TR 94-29 | 2.00 | ☐ TR 94-36 | 1.50 |
| ☐ TR 94-30 | 2.50 | ☐ TR 94-37 | 2.00 |

**Icon Newsletter**

This newsletter contains information about the Icon programming language. Typical topics include reports on language design, implementation, and notices of new publications. It is issued aperiodically, two or three times a year.

**Software Distributions**

The Department distributes a variety of software in machine-readable form. Examples are the Icon programming language, SB-Prolog, SR, *x*-kernel, and the Scorpion System. Information about the contents and availability of specific distributions can be obtained by checking the boxes listed below. Most distributions are available for a nominal fee, which includes media and the associated documentation, or via anonymous FTP from cs.arizona.edu.

☐ Please add my name to the distribution list for the Icon Newsletter

Please send me information on the following software distributions:

☐ The Icon programming language
☐ The SR programming language
☐ The SB-Prolog System
☐ The *x*-kernel
☐ The Scorpion System

Technical Librarian
Department of Computer Science
The University of Arizona
Gould-Simpson 721
Tucson, Arizona 85721
USA