REPORT AD-A262 087

Form Approved OMB No. 0704-0188

12

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 930302	3. REPORT TYPE AND DATES COVERED final 880801-930131

4. TITLE AND SUBTITLE Workform Processing: A New Paradigm for Fault-tolerant Distributed Parallel Computation	5. FUNDING NUMBERS C: N00014-88-K-0619
6. AUTHOR(S) David Cheriton	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Computer Science Department Stanford University Stanford, CA 94305	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 800 N. Quincy St. Arlington, Virginia 22217-5660	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release: distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

See attached report.

93-06447 17p8

98 3 29 098

14. SUBJECT TERMS	15. NUMBER OF PAGES 16
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UL	UL	UL	UL

Final Report to DARPA and ONR on Workform Processing: A New Paradigm for Fault-tolerant Distributed Parallel Computation

David R. Cheriton
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

PI: David R. Cheriton

Institution: Stanford University

Phone: 415-723-1131

E-mail: cheriton@pescadero.stanford.edu

Contract: Workform Processing

Contract: N00014-88-K-0619

Period: 8/1/88-1/31/93

1 Report Summary

This project explored the workform model of parallel processing, looking at its implications for parallel architectures, operating systems and parallel application structuring. The architecture investigation lead to the design and construction of a scalable shared memory multiprocessor, ParaDiGM, that included as novel features: large cache lines, software-controlled caches, locking and messaging support in the caches and hierarchically shared caches. Various detailed studies coming from this work has shown that these features provide major improvements in cost-effective scalability for parallel computation, particularly parallel simulations.

The work on parallel operating system support lead to two major innovations, application-controlled page caches and memory-based messaging. The former allows an application to completely control the allocation of physical memory allocated to it, avoiding unexpected page faults. The latter provides application-level access to the harware-supported memory-based messaging as well as providing an extended abstraction of this facility. These novel approaches were shown to provide major benefits to large-scale parallel applications. They also show the potential of reducing the complexity of so-called microkernels for operating systems by allowing major portions of the virtual memory management and interprocess communications to be simplified or removing from the kernel, thereby reducing kernel complexity and improving reliability. These gains are achieved without compromising applications performance, and in fact allow significant improvements in sophisticated applications, such as the parallel simulations on which the work focused.

The application structuring work investigated the use of object-oriented techniques for supporting cache architecture sensitive parallel applications restructuring (CASPAR). An object-oriented library was developed that supports cache-aligned memory allocation, processor-affinity, spatially-based load balancing and distributed temporal-based synchronization, all oriented towards spatially-based parallel simulation. Extensive study of a large-scale wind tunnel simulation restructured to use this library showed dramatic improvements, not only in raw performance, but in key measures such as cache hit rates and load balancing. Thus, these techniques will provide even greater benefits for future hardware systems, in which the ratio of processor cycle time to memory access latency and communication latency is expected to increase significantly.

Overall, the work of this project showed enormous merit in the original focus of the work, that memory and communication were the key issues to focus on for parallel performance, not processors, as has been the traditional focus. This novel focus lead to major innovations in the hardware architecture, the operating system support and parallel application structuring. The techniques developed under this contract appear applicable to military and commercial systems both of a general-purpose and special-purpose nature.

2 Numerical Productivity Measures

PI: David R. Cheriton

Institution: Stanford University

Phone: 415-723-1131

E-mail: cheriton@pescadero.stanford.edu

Contract: Workform Processing

Contract: N00014-88-K-0619

Period: 8/1/88-1/31/93

- Refereed papers submitted but not yet published: 5
- Refereed papers published: 11
- Unrefereed reports and articles: 4
- Books or parts thereof submitted but not yet published: 0
- Books or parts thereof published: 1
- Patents filed but not yet granted: 0
- Patents granted: 0
- Invited presentations: 4
- Contributed presentations: 28
- Honors received: 2 PYI to D. Cheriton and Anoop Gupta
- Prizes or awards received (Nobel, Japan, Turing, etc.): 0
- Promotions obtained: 1
- Graduate students supported at 25% or more: 13
- Post-docs supported at 25 % time or more: 1
- Minorities supported: 0

PI: David R. Cheriton

Institution: Stanford University

Phone: 415-723-1131

E-mail: cheriton@pescadero.stanford.edu

Contract: Workform Processing

Contract: N00014-88-K-0619

Period: 3/1/88-1/31/93

3 Overview of the Project

This project was fundamentally based on the recognition that memory considerations were as important, if not more important, than processing considerations. Memory access is the performance limiting factor in systems, not processor technology. Also, state is what has to be maintained and recovered to achieve fault-tolerance. Finally, memory or state structuring is a more familiar way of structuring programs for programmers than "processing" structuring. For example, the programmer is scarcely aware of the notion of "process" in conventional sequential programming, yet concerns himself extensively with the data structuring of the program. One of the difficulties in conventional parallel programming approaches is that the programmer must directly and completely concern himself with processes and how they interact.

In contrast to this focus in this project, much of the previous and concurrent work in parallel computer systems has focused on the processing aspect. Multiple processes or threads are presented as key abstractions to support parallel programming, with their attendant synchronization and communication primitives. Correspondingly, parallel program structuring work has focused on how to subdivide the work among these processes. Moreover, related work has focused on load balancing and sophisticated operating system support for parallel processes. We felt that sufficient attention to memory-related issues and in fact, memory-driven models of structuring parallel programs, were lacking in this work, and set out to address this deficiency.

3.1 Previous Experience and Background

The previous experience of the PI and his research group was primarily in distributed operating systems and networking. In particular, we had developed a distributed operating system, the V distributed system, that ran across a cluster of workstations connected by a local-area network. The system supported network transparent message-passing, on which was built network file access, remote execution, distributed scheduling, process migration and distributed parallel execution, i.e. running parallel programs across clusters of workstations. We had also done some preliminary work on distributed virtual memory in this environment. Thus, we had experience with loosely-coupled parallelism, message passing and virtual memory structuring in a distributed systems environment. It is relevant to note that the V operating system was considerably more sophisticated than those found on distributed memory parallel machines of the time, and even to a significant extent now. Thus, we felt we had an important basis on which to contribute to the advancement of work in the parallel processing area. Relating to our emphasis on memory within the parallel processing context, we recognized that the operating system is a major component of the effective application memory system because it implements the abstraction of protected, separate virtual address spaces on top of the hardware. In factor, one might view the operating system/hardware collaboration to implement memory systems as analogous to the compiler/processor collaboration to implement high-level languages. In this vein, we were interested exploring possible hardware-software trade-offs in the implementation of parallel processing memory systems.

In this previous work, the highly concurrent structuring of the file server in V provided recognition of the need for, and inspiration for, a model of parallel processing that was decoupled from the notion of process. The basic model, the workform model, was the starting point for this work.

3.2 Workform Model

The basic inspiration for the workform model derives from the office notion of forms specifying (sub)tasks to be performed being passed around between workers. The workform model provides a familiar paradigm of parallelism in which a wide variety of applications can be programmed. As a parallel programming language model, it also recognizes that the key issue in parallel language design is the representation and implementation of state, not concurrency or synchronization primitives per se. Forms-based programming groups logically related portions of the state into physical units of caching and memory transfer, encouraging higher cache hit ratios, faster memory hierarchy transfers and lower levels of contention between processors pursuing unrelated tasks. The workform approach also minimizes subtasking overhead (where a subtask is the unit of potential parallelism and the workform is an implementation of subtask) since it: minimizes the cost of subtask context switching by making subtask context independent of hardware context; minimizes the cost of subtask communication by colocating all subtask information in workforms; and, maximizes locality of reference within a subtask execution as well as separation of reference between subtasks by contiguous allocation of each workform and separation of allocation of workforms based on processor cache characteristics.

Finally, the workform approach facilitates program survivability because a workform, like an office form, consolidates the critical information about a subtask into a simple, identifiable state description. "Critical information" here refers to information that is required to carry out the subtask. It specifically excludes temporary state information that a workform processor might generate while processing a workform. Consequently, the information contained in workforms is adequate to restart a computation after a fault. Thus, it is sufficient to provide redundancy, error detection and recovery for workforms. This reduces the problem of maintaining redundant copies of state to workforms, rather than the entire state. In particular, the rapidly changing temporary state is separated out.

This is analogous to a sophisticated office system where forms are identified, tracked, checked and logged as they proceed through the office, allowing tracing and recovery of missing forms, checking to detect incorrect or unauthorized actions and auditing to determine the source of failures. The forms management techniques also provide a means for synchronization of effort on a particular task.

Our original research plan was to:

1. Design and implement a parallel programming system that supports the workform model with emphasis on performance, program survivability and programmability.

2. Design and implement a multiprocessor cluster with a distributed memory hierarchy with a careful partition of function and optimizations between hardware and software, analogous to issues addressed with the reduced instruction set processor design and compiler work of the last decade. These refinements to support the workform approach are simplifications of the hardware, exploiting software techniques than extra complications for forms-based programming.

3. Implement several example parallel applications that demonstrate the ease of expression, performance attributes and program survivability of this approach.

The multiprocessor effort, the memory hierarchy work in particular, was to complement DARPA-supported work on high-performance processors. It was also seen as exploring a portion of the design space that is complementary to work such as the Connection Machine or iWarp. The distributed and parallel operating systems work provides techniques, protocols and software that can feed into standard industrial systems, such as the various favors of Unix and Mach.

3.3 Evolution of Project Directions

The project evolved into three identifiable sub-efforts, focused on application structuring, operating system work and architecture work.

The workform processing evolved into work on application structuring. Some initial work on compiler support showed promise but was abandoned after some key personnel were lost. Subsequent focus was on the use of library/run-time mechanism mechanisms to provide program structuring support, with initial work on a general even-driven simulation library (OLPS), and then a full effort leading to the Space Library in support of physically spatially based 3-dimensional particle simulations. The workform concept was been integrated into an object-oriented framework, and the work focused primarily on exploring classes of simulations that are amenable to object-oriented simulation. Techniques were developed for structuring the parallel applications that improve their scalability on typical shared memory multiprocessor architectures, particularly the ParaDiGM architecture developed under this contract. Performance measurements have shown significant performance and scalability benefits (like a factor of 50 reduction in communication costs) of these structuring techniques in some cases. As a general assessment, the software effort has produced a new programming model and implemented a prototype environment that supports that model, plus one application that clearly demonstrates its merits. Thus, although the original workform model has been very inspirational, the work has resulted in incorporating it as an extended object-oriented environment both to better capitalize on existing tools, and to more easily migrate the results to the broader community. The result has been a reduction in the amount of compiler development that has been necessary. The environment developed to date appears very promising, and although more targeted than originally proposed, has shown major benefits. The focus on a single application has provided greater performance results and more sophisticated extensions than had been planned, and is expected to payoff significantly when this software is applied to other applications.

The initial architecture work lead to the development of the VMP multiprocessor (some of this effort being supported by an earlier short contract from DARPA). This multiprocessor explored the use of large cache line sizes and software-controlled cache management, allowing control of memory access at the granularity of cache lines. This software approach allows a significantly simplified approach to multiprocessor design, what might be viewed as the RISC philosophy applied to multiprocessor systems, judicially moving function into hardware to minimize complexity, maximize speed-up and make the remaining hardware resources more reliable. In particular, the innovative cache design we developed based on the use of software management of the cache provides a basic set of hardware "resources" to minimize the software overhead. Besides reducing the amount of hardware required, this design provides significantly extended cache management functionality and flexibility, all without loss of performance.

The subsequent architecture work focused on how to scale to much larger numbers of processors than the VMP. This work, leading to the ParaDiGM architecture, focused on multi-level shared caching, memory-based messaging, cache-based locking support and exploiting the large cache lines and directory-based consistency protocols to implement this schemes.

The operating systems work started with the V distributed system as a basis. The further development of this software was not a research objective per se. Rather, we viewed it as a research vehicle that seemed well-adapted to supporting our investigations. It is smaller and far more familiar to us than Unix or Mach and therefore allowed us to port it to our new hardware. The "overhead" part of this effort was porting the system to the various new architectures we used as part of this work. The overhead was minimized by focusing on a small number of architectures, particularly the MIPS and 680x0 architectures. The latter was supported by the original development of V. The former was adopted as a simple architecture that gave exposure to RISC architecture issues, and provided a route for evolution in the future, viewing the MIPS architecture as having a more promising future than the 680x0.

The operating system research entailed modifications to V that support the issues of interest in this project. Initial work on this area entailed extensions to provide cache management support, integrated with the virtual memory system. Additional extensions were made to support large memory systems, distributed file caching, application-control of virtual memory and operating system support for memory-based messaging.

A final sweeping objective for the project was to have at least one parallel application running on our parallel operating system running on our parallel architecture. At the time of writing, we are short of this goal but close, and attempting to use some gift funds to reach this goal. Currently, we are able to configure a

16-processor ParaDiGM machine but the operating system is not running fully reliably on this configuration. We also have the MP3D parallel application running on ParaDiGM on top of the V operating system, but not in the full parallel form. Thus, we have built much of what we set out to build but did not manage to get it all fully debugging and working within the time and funds of the contract.

The following section provides descriptions of the research results in more detail.

4 Summary of Technical Results

The research work under this contract investigated techniques for highly scalable and fault-tolerant parallel programming, with work ranging from parallel application programming and structuring techniques to operating systems innovations, down to next generation cache-based architecture techniques. These areas are described below. These descriptions are but a summary of the work which is more fully described in the cited research reports.

4.1 Parallel Program (Re)Structuring

Memory bandwidth is the key limiting factor in the performance and scalability of many parallel applications. Similarly, the handling of memory/state is the key to fault-tolerance and recovery. Consequently, a key focus of the work was on techniques for structuring parallel applications to make the best use of memory bandwidth provided by architectures of interest and maximize the potential for recovery from faults. The primary architectural class of interest is generically the cache-based shared memory multiprocessor with homogeneous general-purpose processors designed for symmetric multiprocessing. As described below, we developed the design for a scalable architecture, ParaDiGM, in this generic class with several innovations, including large cache lines, hierarchically shared caches, message-based messaging, cache-directory based lock support, cache-based prefetch support and other features. The rest of the section describes our application-structuring work that improves applications for this class of architecture, and also allows them to take particular advantage of the architectural innovations we are exploring.

Our work focused on MP3D, parallel hypersonic flow wind tunnel simulation based on the particle technique. Our original work focused on what we call Cache Architecture Sensitive Parallel Application Restructuring (CASPAR). (These results are described in the report "Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: A First Experience", which was published in April 1991.) The restructured program exhibited significantly better performance as a result of the improved cache and bus behavior. For example, the cache miss traffic on a ParaDiGM machine was reduced by a factor of 30 to less than 1 percent, and the bus load on the Sequent, a smaller cache line machine, is reduced by a factor of 7. An important aspect of this work is that the speedup exhibited by the revised program is significantly better for machines with a large ratio of processor cycles to cache miss/memory access times, which are becoming common with high-performance RISC processors, and this effect can be expected for restructuring programs in general.

Second, the program was restructured with almost a textbook-like object-oriented structure which was entirely in concert with the goals of reducing cache misses, false sharing and unnecessary contention. Thus, we are encouraged that our performance objectives can not only be met, but aided, within the framework of object-oriented programming, which appears to be a significant programming methodology for the future. This structure contrasts with the original structure of the program, which used a FORTRAN-like array organization. It became apparent that the array structure was responsible for the extremely bad cache behavior of the original program but was also a necessary aspect of the original program for its efficient execution on vector processors, such as the Cray architecture. We conjecture from this experience that the significant separation between vector processors and object-oriented programming as well as shared memory multiprocessors does not bode well for vector processing.

Finally, the basic approach used in restructuring this program suggests a general applicability for a wide variety of physical system simulations. In essence, we recognize that simulations exhibit locality and clustered communication because each object in the simulation can only directly interact with its immediate neighbors. Our restructured program exploits that locality and maps this locality onto cache locality. For example, a processor is assigned a region of space and processes all particles in that region of space. Thus, processors need only interact when processing particles on the boundaries.

It seems feasible to provide relatively general-purpose programming support to exploit this "natural" locality by mapping it onto cache locality. Our report on ParaDiGM contains additional examples of the locality and some basic analysis of its scalability with ParaDiGM.

The work also partitioned this particular application program into a general-purpose class library and the application-specific code for MP3D. The library is designed to support a range of physical simulations for which the spatial division of processing is appropriate. This library incorporates the techniques we have developed to provide good cache behavior, including aligned memory allocation, space-based processor allocation and object-oriented (clustered) memory allocation for locality. The class library provides a set of generic classes that incorporate support for these mechanisms, and allows applications to derive and specialize application-specific classes of objects from these generic objects using the object-oriented technique of inheritance.

The work separating the library portion from the MP3D application has also lead to considerable refinement in software and techniques to make them more application-independent. However, a deficiency at this stage is still a lack of additional applications using this library. We had planned to develop additional applications using this library to give some credibility to our claim of generality to the library's facilities. However, in keeping the requirements of other applications in mind during the development of this library, we feel that confident that the basic library design is sound for wider applicability, but expect some refinements and extensions may be necessary, especially in supporting applications significantly outside basic particle-based simulation. The progress in developing applications was been slowed by the exciting number of techniques we have discovered and wished to investigate as part of this library support, the latest of which are described briefly below.

Distributed temporal synchronization was incorporated into the library, replacing conventional barrier-based global synchronization. This step has significantly improved performance (roughly by a factor of 2). Furthermore, extensive performance evaluation was done of the benefits of allowing time skew within the computation, which allows, in effect, a processor to execute on data it has in its cache for multiple time steps, rather than a single time step, further improving the utilization of the cache. Although our measurements showed some small amount of speedup (around only 10 percent, unfortunately), another factor was identified that appears to be negating must of the benefit we expected to see. That is, the library provides locality of cache line reference but does not ensure virtual page locality. On machines such as the SGI 4D/380 we have used for this work, with TLBs (translation lookaside buffer) for translation to a physically addressed cache, lack of page locality is causing an excessive degree of TLB misses, considerably slowing the computation speed. A recently extended version of the software maintains page locality by copying certain data being transmitted between processors. The result has been a considerable improvement in performance. However, this work was not tuned before the end of the contract to provide satisfying performance measurements.

The copying required to solve the page locality problem is effectively a form of memory-based messaging between processors. We started to look at using the hardware and OS-support memory-based messaging in the ParaDiGM hardware in MP3D, but did not complete this work before the end of the contract.

Finally, we have carried out some basic work in providing better support for per-thread data within a multi-threaded address space. For example, each thread requires a record of its identifier, stack area and resources it holds. Conventional languages provide local-to-procedure variables and global to address variables but generally nothing in between. We determined that having a default of thread-private provided the simplest semantics but required the most of the implementation. Our implementation has demonstrated that these semantics can be provided at a reasonable cost. This work, largely carried out by Peter Ham, a Masters student, was not be written up extensively, but was been made available to other compiler groups for incorporation.

4.2 Operating Systems Support

Our operating systems work is focused on support for a ParaDiGM-like architecture, where the key problems arise from the scale and complexity of parallelism. A basis of our work has been the V kernel, which we continue to extend and refine for ParaDiGM architectures. The V kernel is designed to be a minimal-sized kernel that provides the key abstractions for building parallel and distributed applications and services. The goal of minimizing the size of the kernel is motivated by performance, reliability and maintainability concerns. With the complexity of cache management, parallelism and fault-handling, the kernel needs to be as small as possible to get right and to tune.

A Ph.D. student, Cary Gray, and the PI looked at the problem of maintaining file caches consistent in a fault-tolerant way. This mechanism ties directly to consistent virtual memory in V because the shared memory is effectively implemented as shared (cached) files that are memory-mapped into the address space. The result of this work was a new cache consistency protocol, as described in SIGOPS'89, and implemented in V. The protocol also shows the benefit of recognizing different types of files and their access patterns, a distinction which also arises with memory. This work was more fully described in Gray's Ph.D. thesis.

A Ph.D. student, Kieran Harty, and the PI looked at the problem of managing very large memory systems, as arises with a large-scale ParaDiGM configuration. With large memory systems, page faults make be infrequent, but their effects can be catastrophic on performance in parallel systems. For example, if the operating system releases a page that is part of a shared data structure as part of page replacement, multiple processes may queue up on the page-in, bringing the program almost to a halt for thousands of processor cycles. Worst yet, some of these processes may be holding locks, which cause further (indirect) blocking by other processors. We explored the approach of moving control of the page cache outside the kernel to the application and shared cache managers. In this approach, the application or a shared cache agent can determine which pages to replace and also control prepaging strategies. We refined the implementation of this scheme and gathered more in-depth performance numbers. A report, *Application-controlled Physical Memory using External Page Cache Management* was published in ASPLOS'92. Harty is making this work into Ph.D. dissertation, which should be complete by June 1993.

We also developed a decentralized approach to global memory management, supporting the above application-level page cache managers. A draft paper, *A Market Approach to Operating System Memory Allocation*, has been written along with a simulation program that has provided a performance evaluation of this approach, and more specifically, some particular algorithms. At the end of the contract, we were still working to incorporate this mechanism into our running experimental operating system.

Our second major effort in the operating system (and hardware) was the provision of OS and hardware support for memory-based messaging. A second Ph.D. student, Robert Kutter, extended the V++ kernel to support memory-based messaging and ported the kernel to the ParaDiGM hardware. It was originally implemented on DECstation 5000s pending the availability of the ParaDiGM hardware. A paper was been written and submitted for publication describing the design, implementation and a simulation-based performance evaluation of its benefits of the approach on machines of increasing scale, *Optimizing Memory-Based Messaging for Scalable Shared Memory Multiprocessor Architectures.* Kutter is pursuing this topic as his Ph.D. thesis and expects to complete in the July 1993 timeframe.

In some earlier work on this contract, Gupta and Tucker looked at the problem of having applications automatically adapt to a varying number of processors. In particular, a parallel program running on a multiprocessor concurrently with other applications has to share the pool of processors. The number of processors available changes as old applications terminate and new applications begin execution. They show that a number of parallel applications perform quite badly if they do not adapt to changing number of processors. In their work, they extend operating system services so that an application is alerted by the operating system when it loses a processor. They then modified these applications to adapt to increases and decreases in the number of processors available to them. Their results (reported in "Process Control and Scheduling Issues for Multiprogrammed Shared-Memory Multiprocessors", Symp. on Operating Systems, Dec. 1989) show the modified operating system and programs leads to significantly improved performance.

A consequence of our focus on optimal minimal kernel design, the V kernel provides an interface that is incompatible with Unix or any other existing standard operating system. To minimize the software development outside of the critical path for kernel research and maximize the software we can exercise on the system, we developed emulation support in the V kernel that allows an emulation segment to implement arbitrary system calls for other operating system interfaces. We also developed a version of this emulation segment that supports a significant portion of the Unix operating system interface, allowing a wide variety of Unix programs to be executed under emulation using the V software. Performance measurements of this software (as reported in "Binary Emulation of Unix using the V Kernel" in Usenix, June 1990) indicate that many of the programs and system services execute *faster* under emulation than under the native system. The primary explanation for this surprising situation is that data copies of I/O, a performance-critical aspect of most Unix programs, can be handled by memory mapping and unprotected block data copies between the emulator and Unix program data space.

We also explored this approach as a way of achieving greater parallelism in the operating system implementation even for standard systems such as Unix. Because the emulator segment is not globally shared (as is the Unix kernel on most shared memory implementations), the degree of parallelism within the emulator segment is roughly limited to the degree of parallelism within the corresponding Unix address space. With large-scale configurations of machines like ParaDiGM, partitioning the parallelism to minimize contention on locks and avoid the need for fine-grain and complex locking protocols seems critical. Our emulation approach seems very successful in supporting a standard operating system interface such as Unix without the monolithic kernel that it seems to otherwise imply.

4.3 ParaDiGM Architecture

ParaDiGM is a highly scalable shared-memory multi-computer architecture. It was being developed to demonstrate the feasibility of building a relatively low-cost shared-memory parallel computer that scales to large configurations yet provides the single-stream performance of high-end microprocessors. A key problem is building a scalable memory hierarchy.

We believe that the shared memory architecture is the preferred paradigm on which to unify and focus parallel software efforts because of its ease of programming, compatibility with standard programming languages and systems, and because shared memory multiprocessors are clearly becoming the dominant architecture for small-scale parallel computation. Shared-memory architectures developed to date have not been designed to scale beyond a few tens of processors because of the memory bandwidth requirements, prompting a wide-spread belief that they cannot be designed to scale. ParaDiGM is targeted at demonstrating the feasibility of a shared-memory parallel computer system that scales from a small number of processors to hundreds of processors or more, providing cost-effective performance over that entire range. That is, users should find ParaDiGM configurations, from small to large, competitive with other machines with comparable processing capacities. All configurations run the same software and are incrementally upgradeable from the smallest to the largest configurations. The availability of high-performance low-cost microprocessors makes this scaling feasible from the standpoint of raw processing power. The problem lies in the interconnection.

Parallel computation places demands on the interconnection mechanism either because programs are *consistency-intensive* (the processors exchange significant amounts of information among themselves), or else they are *I/O-intensive* (the processors read or write large amounts of information between themselves and the outside world). These two types of traffic have different characteristics. Consistency traffic is characterized by rapid and repeated transfer of the same small data units between processors. For example, the head of a task queue can be the focus of continuous contention in some applications as processors add and remove tasks. I/O traffic is characterized by the sustained transfer of large blocks of data. Typically the data is discarded after processing. For example, trace data driving a simulation can be most efficiently read in large blocks and is discarded after reading.

ParaDiGM achieves high-performance scalability by exploiting distributed operating system techniques for very large-scale, widely distributed operation, optimized shared cache and bus multiprocessor techniques among local clusters of tightly-coupled processors, and parallel application structuring techniques that maximize locality and minimize contention within the resulting architecture. A overview paper on ParaDiGM was published in IEEE Computer, February 1991, (ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, D.R. Cheriton, H.A Goosen and P.D. Boyle), describing the architecture, a number of innovations, and some solid simulation and analytic evaluation. It also incorporates a number of significant innovations, and also acts as a guiding target for our software effects.

Considerable progress was made in experimentally exploring and evaluating the ParaDiGM architecture. First, a Ph.D. student, Hendrick Goosen, completed his thesis on shared caching as exploited in the ParaDiGM architecture, providing extensive direct-execution and trace-driven simulations of a wide-range of cache designs and parameters, showing both the strong benefits of this approach and limitations. In particular, his temporal simulation showed the effects of secondary cache queuing as the number of client caches are increased, particularly if the cache does not support concurrent hit handling during miss processing.

We completed the construction of 4 4-processor boards, which can be configured as 4 separate machines or one 16-processor machine, else combinations in between. The multi-processor board includes sophisticated support for cache-based locking, messaging and hierarchical cache sharing. A daughter board providing a bus interface to a third level of memory has also was designed, implemented and is now in used. The V++ operating system has also been largely ported to this hardware, and it is feasible to run simple multi-process application programs in parallel.

This hardware has enabled us to experimentally evaluate the behavior of hierarchical shared caching, hardware support for memory-based messaging and cache-based locking support. Some preliminary results in this area have been gained and we are hoping to continue the work in this area.

The precursor of the ParaDiGM design was originally described and reported in ISCA'89 in Israel. Paper was entitled Multi-level Shared Caches in the VMP-MC. A basic trace-drive simulator for this architecture was developed and used to perform an initial evaluation of the sharing behavior of parallel programs on this architecture. We also developed PC version of the memory board for the VMP-MC which included a sophisticated consistency directory, plus support for memory-based messaging and locking. The implementation clearly demonstrated the feasibility of this approach. However, we subsumed the use of this board by developing the ParaDiGM multiprocessor board, and a change in personnel delayed the development of the memory board.

The ParaDiGM hardware developed was preceded by the development of the VMP architecture. This work, partially supported by this contract, produced a working small-scale multiprocessor and demonstrated the feasibility of software-controlled caching, as reported in the ISCA'88 conference. This work also covered the initial effort to support this type of architecture in the operating system, reducing the time and effort required to port to the ParaDiGM hardware, which has the same properties.

PI: David R. Cheriton

Institution: Stanford University

Phone: 415-723-1131

E-mail: cheriton@pescadero.stanford.edu

Contract: Workform Processing

Contract: N00014-88-K-0619

Period: 8/1/88-1/31/93

5 Publications and Presentations

Reports:

- D.R. Cheriton and A. Gupta and P. Boyle and H. Goosen, The VMP Multiprocessor: Initial Experience, Refinements and Performance Evaluation, Proc. 15th Int. Symp. on Computer Architecture, ACM, May, 1988
- H. Kanakia and D.R. Cheriton, The VMP Network Adapter Board NAB: High-performance Network Communication for Multiprocessors, SIGCOMM 88, ACM, August, 1988
- D.R. Cheriton, The V Distributed System, Comm. ACM, March, 31(3), 1988, p. 314-333
- M. Abrams, The Object Library for Parallel Simulation (OLPS), Proc. Winter Simulation Conference, San Diego, 1988.
- D.Cheriton, H. Goosen and P. Boyle, Multi-Level Shared Caching in the VMP-MC, Int. Symp. on Computer Architecture, June 1989.
- A. Tucker and A. Gupta, Process Control and Scheduling Issues for Multiprogrammed Shared-Memory Multiprocessors, Symp. on Operating Systems, Dec. 1989.
- C. Gray and D. Cheriton, Leases: An Efficient Fault-tolerant Mechanism for Distributed File Cache Consistency, Symp. on Operating Systems, Dec. 1989.
- D.R. Cheriton, G.W. Whitehead and E.W. Sznyter, Binary Emulation of Unix using the V Kernel, Usenix, Anaheim, June 1990.
- C.G. Gray, Performance and Faul-tolerance in a Cache for Distributed File Service, Stanford Technical report STAN-CS-91-1363, December 1990.
- ParaDiGM: A Highly Scalable Shared-Memory Multi-computer Architecture, D.R. Cheriton, H.A Goosen and P.D. Boyle, IEEE Computer, Feb. 1991.
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: A First Experience, D.R. Cheriton, H.A. Goosen and P. Machanick, Shared Memory Multiprocessing Conference, Tokyo, April 1991. (Also to appear in a book from MIT Press with the same title.)
- Hendrick Goosen, Shared Multi-level Caches for Scalable Multiprocessors, Ph.D. Thesis, Stanford University, July 1991.
- P.Machanick, The Space Library, internal report, September 1992.

- Application-controlled Physical Memory using External Page-Cache Management, K. Harty and D.R. Cheriton, ASPLOS'92
- A Market Approach to Operating System Memory Allocation, D.R. Cheriton and K. Harty, draft paper.
- H. Goosen and D.Cheriton, The Performance of Shared Multilevel Caches, submitted for publication, September 1992.
- Optimizing Memory-Based Messaging for Scalable Shared Memory Multiprocessor Architectures, D.R. Cheriton and R.A. Kutter, submitted for publication.
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: The Value of Distributed Synchronization, P. Machanick and D.R. Cheriton, accepted for publication.

Presentations:

- Parallel Operating Systems and Language Techniques, M. Abrams, K. Harty and J. Pallas. DEC Systems Research Center, Palo Alto CA, July 1989 (invited).
- Multi-level shared caching in the VMP-MC, Int. Symp. on Computer Architecture, Israel, June 1989
- Second-level cache behavior, SigArch Cache Workshop, Eilat, Israel, June 1989.
- Object Library for Parallel Simulation, Stanford Dec. 1988
- Parallel Discrete Event Simulation: A Survey, Stanford, January 1989.
- ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, Sequent Computer Corporation seminar, Oct, 1989.
- ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, Seminar, Bellcore, Oct. 1989
- ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, Computer Systems Colloquium, Princeton, Oct. 1989
- ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, Computer Systems Colloquium, MIT, Oct. 1989
- ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, Computer Systems Colloquium, CMU, Oct. 1989
- ParaDiGM; A Highly Scalable Shared-Memory Multi-computer Architecture, Computer Systems Colloquium, UC Berkeley, Nov 1989.
- Process Control and Scheduling Issues for Multiprogrammed Shared-Memory Multiprocessors, Symp. on Operating Systems, Dec. 1989.
- The V Server Architecture for the 1990's, UniForum, Jan 1990.
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: A First Experience, MIPS Research Meeting, May 1990
- Binary Emulation of Unix using the V Kernel, Usenix, Anaheim, June 1990.
- Binary Emulation of Unix using the V Kernel, seminar, Bellcore, Sept 1990.
- Distributed Computer Systems in the 90's, Hawaii Governor's Conference, Nov, 1990.

- Distributed Systems: The Next 100 Years. U. of Arizona, Nov,. 1990
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: A First Experience, H.A. Goosen. Shared Memory Multiprocessing Conference, Tokyo, April 1991.
- ParaDiGM: A Highly Scalable Shared-Memory Multi-computer Architecture, DARPA High-performance Computing Workshop, Oct. 1990.
- Application-controlled Physical Memory using External Page-Cache Management, K. Harty, Digital Equipment Corporation, June 1991.
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: A First Experience, P. Machanick, Digital Equipment Corporation, June 1991.
- The Group Approach to Experimental Distributed Systems Research, Experimental Computer Science Workshop, Oct. 1991, Palo Alto.
- Application-controlled Physical Memory using External Page-Cache Management, K. Harty, SOSP, Oct.'91, work-in-progress session.
- Dissemination-oriented Communication Systems, D.Cheriton, Computer Forum, Stanford University, Feb 1992.
- Dissemination-oriented Communication Systems, D.Cheriton, Apple Computer, March 1992.
- Dissemination-oriented Communication Systems, D.Cheriton, UCSD, April 1992.
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor, D. Cheriton, Hal Computer Systems, May 1992.
- Future Directions in Computer Communications, D. Cheriton, Teknekron Software Systems, May 1992.
- Dissemination-oriented Protocols, D.Cheriton, Stanford University, June 1992.
- Dissemination-oriented Communication Systems, D.Cheriton, University of Waterloo, June 1992.
- Support for Object-oriented Distributed Simulation, D. Cheriton, IDA, June 1992.
- Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared Memory Multiprocessor: The Value of Distributed Synchronization, P. Machanick and David Cheriton Stanford University, August 1992.
- Problem-oriented Shared Memory Revised, D. Cheriton, SIGOPS European Workshop, Sept. 1992.

Awards:

- Presidential Young Investigator Award - Anoop Gupta.
- IBM Fellowship for Graduate Study - Hendrick Goosen.

PI: David R. Cheriton

Institution: Stanford University

Phone: 415-723-1131

E-mail: cheriton@pescadero.stanford.edu

Contract: Workform Processing

Contract: N00014-88-K-0619

Period: 8/1/88-1/31/93

6 Research Transitions/Interactions

Some of our technology was provided to the committee for the Next Generation Computer Resources (NGCR). The PI also had several extended discussions with the IDA representative Dr. Karen Gordon on the technology, supplying papers and proposing some directions, including serious consideration of the Posix interface, the final selection. With our work with binary Unix/Posix emulation, we see our work as strongly applicable to the next generation "implementation" of operating systems, rather than an immediate reason to go to a new non-standard interface.

The PI participated in a panel discussion at UniForum in Washington in January 1988 on the next generation of server architectures.

A software prototype emulation of the first two levels of the memory hierarchy in the ParaDiGM architecture was demonstrated at the DARPA High-Performance Computing workshop, Oct 2-5th, 1990. This program runs parallel programs using the Tango execution environment, developed by the DASH Project at Stanford University, and supports the analysis of cache behavior, including timing and queuing on critical shared resources, such as busses and shared cache lines and includes a visualization interface to monitor and debug cache behavior. We have made this program freely available to other researchers.

We have had informal interactions with various members of the DARPA community plus interactions with industry, including interest in the ParaDiGM architecture from DEC, Encore, Sequent, Apple, Convex and Motorola. In addition, the PI participated in a DARPA ISAT study on distributed interactive simulation culminating in a presentation at the August 1992 Woodshole ISAT Plenary Meeting and a presentation to the Defense Sciences Board. The work under this contract, both on parallel architectures and on parallel and distributed simulation, was the basis for the PI's contribution to this study.

Finally, a significant number of students have been supported under this work, at the Ph.D., Masters and undergraduate levels. Most of these students are now working for major American Computer companies, including SGI, DEC, Apple, SUN. It is the training of students in the real experimental systems prototypes that are made possible by DARPA sponsorship that we expect are most major impact to come.

PI: David R. Cheriton

Institution: Stanford University

Phone: 415-723-1131

E-mail: cheriton@pescadero.stanford.edu

Contract: Workform Processing

Contract: N00014-88-K-0619

Period: 8/1/88-1/31/93

7 Software and Hardware Prototypes

In 1987, a prototype version of the VMP processor board was produced, initially as a wirewrap version and then a PC board version.

A software prototype of the V distributed operating system was produced for the VMP node.

A software prototype of the V operating system running on the DEC Firefly workstation (5 processors) was produced.

A PC version of the memory board for the VMP-MC designed was developed, laid out and debugged. This board includes a sophisticated consistency directory including novel support for memory-based messaging and locking.

A prototype compiler was produced that supports per-process variables for multi-process programs within a common address space. This work involved issues at the syntactic level, semantic level and linking level of compilation.

An object-oriented simulation package called OOPLS was developed (by Marc Abrams) which allows experimentation with the two common parallel simulation approaches (time warp and bryant-misra-chandy) without changes to the application program.

A prototype of the V distributed system was produced for the MIPS architecture (the DECstation 3100 and 5000's), allowing faster program development, simulation and execution. This prototype tested out aspects of the virtual memory system designed for RISC-processor machines.

A prototype basic Unix binary (software) emulator was developed to run in conjunction with the V-System, to demonstrate this technology and its performance using the VMP-MC architectural ideas.

The *SpaceLib* library we have developed to provide CASPAR support for parallel applications exists as a prototype. This library suite is available to other researchers. It is applicable to a range of physical system simulations for executing on scalable shared memory multiprocessors.

A software prototype of a parallel wind tunnel simulation was developed, MP3D, which represents a sufficiently realistic physical model to be of interest in researchers in the areas of aerodynamics and possibly even fluid dynamics.

The revised version of V, called V++, has been partially developed, and represents a running but not fully revised distributed operating system. We are currently distributing an older version of this software through the Stanford Office of Technology Licensing. We expect to release a new version of this software when it is available, although that may take more than a year of further development.

A 4-processor multiprocessor module was produced for the ParaDiGM architecture as a large printed circuit board. This board includes 4 processors, second-level cache, directory-based consistency, loading support and read-ahead support, as is fully operational.