

AD-A284 831

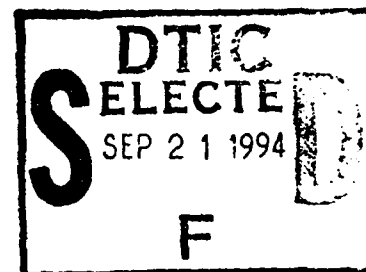


(L)

Final Report to ONR
on Parallel Computation

K. Mani Chandy
California Institute of Technology

September 6, 1994



This document has been approved
for public release and sale; its
distribution is unlimited.

94-30199



1108

94 9 1 32

Mani Chandy
California Institute of Technology
(818) 356-6559
mani@vlsi.caltech.edu
Parallel Computation
Navy Grant No. N00014-89-J-3201
Reporting Period: Final Report

Productivity Measures

1. Refereed Papers Submitted but not yet published: 0
2. Refereed Papers Published: 6
3. Refereed Papers to appear: 1
4. Unrefereed Reports: 1
5. Book Chapters : 0
6. Books published: 1
7. Patents filed: 0
8. Invited Presentations: 6
9. Honors: 0
10. Prizes: 0
11. Promotions Obtained: 0
12. Graduate students supported: 3
13. Postdocs supported: 0
14. Minorities supported: 0

[illegible]

Mani Chandy
California Institute of Technology
(818) 356-6559
mani@vlsi.caltech.edu
Parallel Computation
Navy Grant No. N00014-89-J-3201
Final Report

Technical Progress

Problem Areas

In this project we have worked on three problems: (a) designing a theory of the modular design of parallel reactive systems, and (b) an application of parallel systems of importance to the Navy — parallel discrete-event simulation, and (c) parallel languages that integrate task and data parallelism.

Modular Design and PCN

Consider a reactive system of the following type. Data is fed to the system from many sensors distributed over some geographical region. The system processes the data and then sends responses to actuators that are also distributed. Processing the data requires numeric scientific computation, symbolic computation (such as tree searching) for evaluating alternatives, and use of sizable human interfaces. The system may call programs from existing libraries in C, Fortran and other languages. We developed theory and tools to build such a distributed system in a systematic way so as to ensure that the system is reliable and that it satisfies its specifications.

My specific goal for this grant is to develop a theory that forms the framework for designing compositional systems: systems that can be built by putting subsystems together in a small number of well-defined ways. This theory was used to build a compositional programming notation, called PCN — Program Composition Notation. PCN has been implemented by a group at Caltech and Argonne, funded by ONR, AFOSR and NSF. PCN was developed by Stephen Taylor at Caltech and myself with

support from Ian Foster at Argonne National Laboratories and Carl Kesselman at Aerospace Corp.

PCN is available by anonymous FTP, and it is being used to develop real parallel applications in climate modeling, modeling air-pollution, computational fluid dynamics, and biology. The grant from ONR was used specifically for developing the theoretical foundations underlying PCN.

The theory for PCN was presented in a book written by Steve Taylor and myself.

Lessons from PCN PCN has a simple theory, and the notation is straightforward. Nevertheless, we found that many scientific application programmers were unwilling to try PCN, though computer scientists were quite ready to use the system. The appeal to computer scientists — theoretical foundation, simplicity, transportability — were of less importance to computational scientists. Specific objections from computational scientists, most of whom program using Fortran 77 were:

1. Parallelizing a Fortran program using PCN requires that the Fortran program be decomposed into subroutines, each of which is then called from PCN. Programmers want to parallelize only the subroutines that are compute-intensive, and they want to leave the remainder of their programs untouched. They cannot do this in PCN.
2. PCN does not have iteration; it uses recursion instead. This is a hurdle, though a minor one, for Fortran programmers.
3. The data types of PCN (such as tuples) are borrowed from functional and logic programming, and are strange to Fortran programmers. Likewise, the syntax is strange to computational scientists brought up on Fortran.

We decided to transport those aspects of PCN that computer scientists liked to the Fortran 77 domain. The effort on this project was directed at developing a new theory for a modular Fortran notation — called Fortran M — developed at Caltech and Argonne National Laboratories. Fortran M is a simple extension of Fortran 77, using ideas from object-oriented programming and PCN. It has been implemented on shared-memory systems, and it will be implemented on distributed-memory systems in the coming year. Fortran M is designed so that debugging parallel programs is simplified significantly: in particular, the constructs of Fortran M are

placed into one of two categories — deterministic or nondeterministic — and by using only deterministic constructs programmers can guarantee that multiple runs of the same program with the same inputs will produce the same output whether it is executed on a single node or on a network. Therefore, programmers can debug their programs on a workstation, and then confidently execute the program on a parallel machine.

A theory of communicating processes was developed, that forms the foundation for Fortran M. The theory deals with:

1. Proving that multiple executions of the program with the same input will produce the same output.
2. Verifying parallel programs by extending the theory for sequential program verification.
3. Compositional (or modular) development of parallel programs, where specifications of programs are derived from specifications (and not implementations) of its components.

ONR support was used specifically for the theoretical aspects of Fortran M. The implementation of the compiler was carried out with support from the National Science Foundation Center for Research in Parallel Computing.

Parallel Languages Integrating Task and Data Parallelism

I worked with Prof. Rajive Bagrodia from UCLA to develop extensions of C for task and data parallelism. An integrated task and data parallel language called UC (for Universal C) has been developed. UC compilers for the CM2 are available over the network. UC compilers for networks of workstations will be available in a year.

Universal C The central idea of UC is *index.sets*. Data-parallel operations can be thought as parallel operations on all elements of an array indexed through an index set. A central aspect of UC is its simplicity, and the ease of learning UC for people familiar with C.

Prof. Bagrodia and I extended UC to handle task-parallelism. This is done by introducing channels in a novel way. In conventional approaches, channels are directed

from a process to another process. Since UC does not have processes, channels in UC are shared data structures between virtual processors. UC supports multiple different tasks, each of which is a data parallel program. Task and data parallel integration are important to DoD for applications such as command and control systems, multidisciplinary designs such as designs of launch vehicles, and for complex scientific simulations such as ocean models.

Maisie We also worked on Maisie, a language that can be used either for simulation or for parallel implementation, with a smooth transition from simulation to parallel execution. The central idea behind Maisie is to allow programmers to first simulate a concurrent system and carry out performance optimization and debugging in simulation mode, and then transform the simulation into a parallel implementation in a sequence of small steps.

Distributed Simulation

Space-Time We also developed a unifying theory for parallel discrete-event simulation. The known methods of discrete-event simulation fit into the theory, and the theory can be used to develop new simulation algorithms. We used the theory to develop a new algorithm, called *Space-Time*, implemented the algorithm, and showed that it is efficient.

The unifying theory allows for the implementation of discrete-event simulation systems that can be tailored to specific types of algorithms within the general theory. For instance, if battle management simulations require optimistic algorithms, then space-time can be run in optimistic mode. If circuit simulations perform better using conservative algorithms, then space-time can be used in a conservative mode.

We implemented the space-time algorithm, and several applications have been developed using the space-time system.

Mani Chandy
California Institute of Technology
(818) 356-6559
mani@vlsi.caltech.edu
Parallel Computation
Navy Grant No. N00014-89-J-3201
Final Report

Lists of Publications

- K. M. Chandy "Reasoning About Continuous Systems", *Science of Computing*, Vol. 14, No. 3, Special Issue on Mathematics of Program Construction, pp 117 - 132, Oct. 1990.
- K. M. Chandy and S. Taylor, "The Composition of Concurrent Programs", *Proceedings of Supercomputing 90*, Reno, Nevada, Nov. 13-17.
- R. Bagrodia and S. Mathur, "Efficient Implementation of High-Level Parallel Programs", *Proc. on 4th International Conf. on ACM Conference on Architectural Support for Parallel Languages and Operating Systems*, 1990 pp 142-151
- R. Bagrodia, M.Chandy, Edmund Kwan, "UC: A Language for the Connection Machine" *Proc. Supercomputing 90*, page 525-534. New York.
- K. M. Chandy and S. Taylor *An Introduction to Parallel Programming*, Jones and Bartlett, Boston, Mass, 1991.
- R. Bagrodia, M.Chandy, Wen-Toh Liao, "An experimental study on the performance of the space-time algorithm" *Proceedings, Workshop on Parallel and Distributed Simulation*, Society. for Computer Simulation January 1992.
- R. Bagrodia, M.Chandy, Wen-Toh Liao, "A Unifying Framework for Distributed Simulation," *ACM Transactions on Modeling and Computer Simulation*, October 1992.

- K. M. Chandy " A Theory of Program Composition" Chapter in *Nato Summer Schools Series on Mathematics of Program Construction*, Springer Verlag, 1991.
- V. Austel, R. Bagrodia, M. Chandy, and M. Dhagat, "A Data-Parallel Extension of C", to appear *Journal of Parallel and Distributed Computing*, 1995.

Mani Chandy
California Institute of Technology
(818) 356-6559
mani@vlsi.caltech.edu
Parallel Computation
Navy Grant No. N00014-89-J-3201
Final Report

DoD Interactions

My research group has been working closely with Aerospace Corporation, a nonprofit organization that works for the Air Force Space Division, and with Argonne National Laboratory and the University of California at Los Angeles.

Our joint work with Aerospace Corporation has been successful. This work has two parts. Firstly, the conversion of Aerospace sequential applications in Fortran or C to parallel programs in PCN, and secondly the development of tools by Aerospace Corp. to aid in the development of PCN programs. The specific Aerospace sequential applications that were developed are:

1. Computing optimal trajectories of the space shuttle and other spacecraft. This is a problem in optimization and differential equations. Subroutines of the original Fortran program were composed in PCN and executed with substantial speedups on the Sequent Symmetry. A paper on this work has been published by Aerospace employees:

Twain K. Summerset and Raymond M. Chowkwanyun, "Trajectory Optimization on Multiprocessors" Technical Report, Aerospace Corporation, 1990.

2. Tracking multiple space vehicles. The Aerospace Corporation had a program, mostly in C, for the tracking application. Subroutines from this program have been composed in PCN to get a multicomputer implementation.

Aerospace scientists have also been developing a parallel program in PCN for a computational fluid dynamic model of a Titan rocket motor. The Aerospace contact for the joint work is Dr. Mel Cutler.

Carl Kesselman of Aerospace Corp. developed a performance monitoring tool called Gauge, which is critical for making PCN a useful parallel programming environment.

Cooperation with Argonne National Laboratories has been successful in developing a runtime system for PCN, as well as applications in weather modeling. Sequential code for weather modeling has been successfully parallelized in PCN. The Argonne contact for the joint work is Dr. Ian Foster.

Cooperation with UCLA has dealt with extending PCN to data parallel machines such as the Connection Machine. This cooperation has resulted in an implementation on a Connection Machine; performance experiments show that the implementation is at least as efficient as C* for a variety of problems. The UCLA contact for the joint work is Dr. Rajive Bagrodia.

My research group has been working with Aerospace Corporation, a nonprofit organization that works for the Air Force Space Division, and with Argonne National Laboratory and the University of California at Los Angeles.

The joint work with Aerospace Corp., in the last year has dealt with using PCN to develop parallel programs for a computational fluid dynamics application. The particular application deals with studying shocks in rocket motors. A PCN code for this application is now running at Aerospace on a Sequent Symmetry. Copies of visuals produced by Aerospace of simulations of different shapes can be obtained from Dr. Craig Lee of Aerospace Corporation, El Segundo, California.

Cooperation with Argonne National Laboratories has resulted in the development of parallel climate models in PCN that run on the Intel Touchstone Delta. The Argonne contact for the joint work is Dr. Ian Foster.

The theory of communicating processes was developed jointly with Ian Foster of Argonne National Laboratories.

The work on distributed simulation forms the theoretical foundation for work at ATT on parallel simulation of circuit switched networks. This work was carried out by Eick, Greenberg, Lubachevsky and Weiss.

Mani Chandy
California Institute of Technology
(818) 356-6559
mani@vlsi.caltech.edu
Parallel Computation
Navy Grant No. N00014-89-J-3201
Final Report

Prototypes

PCN is available by anonymous FTP from Argonne National Laboratories at [mcs.anl.gov](ftp://mcs.anl.gov). Details are obtained from tuecke@mcs.anl.gov.

PCN has been used for a variety of applications. Because of its simplicity, PCN has been widely used as a language for teaching introductory parallel programming.

Fortran M is also available by anonymous FTP from Argonne. Fortran M has been used for climate models and other applications, but it is too early for commercial development.