TION PAGE

Form Approved
OMB No. 0704-0188

Pul
gat
coll
Dav

/erage 1 hour per response, including the time for reviewing instructions, searching existing data sources, the collection of information. Send comments regarding this burden estimate or any other aspect of this o Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

AD-A221 951

1.

TE

3. REPORT TYPE AND DATES COVERED
Final Report, 1 Dec 88 to 30 Nov 89

4. TITLE AND SUBTITLE
CSRD PARALLEL SERVICE MACHINE ENHANCEMENT

5. FUNDING NUMBERS
AFOSR-89=0166
61104D 3842/A5

6. AUTHOR(S)
Dr. David J. Kuck

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Illinois
Center for Supercomputer
Research and Development
Urbana, IL 61801

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 90-0463

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR?NM
Building 410
Bolling AFB, DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-89-0166

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Ab-

The three major areas in which the Center for Supercomputing Research and Development (CRSD) has performed research involving DOD support are: I) Parallel Computer Systems Design Evaluation and Simulation Environment; II) Automatic Restructuring and Concurrent Evaluation of Lisp III) Parallel Iterative Solver for Sparse Nonsymmetric Linear Systems. This proposal requested funds to procure additional processors for the Cedar computer system [KLSD84]. These funds enhanced the research capability of CSRD to perform research in the above mentioned areas. Also procurred were disk drives to enhance the storage capacity of this system. [illegible]

14. SUBJECT TERMS

15. NUMBER OF PAGES
8

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

90 05 10 089

The three major areas in which the Center for Supercomputing Research and Development (CSRD) has performed research involving DOD support are:

I. Parallel Computer Systems Design Evaluation and Simulation Environment
II. Automatic Restructuring and Concurrent Evaluation of Lisp
III. Parallel Iterative Solver for Sparse Nonsymmetric Linear Systems

This proposal requested funds to procure additional processors for the Cedar computer system [KLSD84]. These processors enhanced the research capability of CSRD to perform research in the above mentioned areas. Also procurred were disk drives to enhance the storage capacity of this system.

Equipment	Quantity	Manufacturer	Unit Price	Amount
Computational Elements	6	Alliant Comp. Sys.	$40,670.00	$244,020.00
Disk Drives	2	Fujitsu	$4,782.50	$9,565.00
shipping				$127.50
				$253.712.50
Transfer to non–federal funds to close–out account				–132.50
TOTAL				$253,580.00

I. Parallel Computer System Design Evaluation and Simulation Environment

One of the major activities of CSRD is evaluating various multiprocessor architectures. Simulation is the only way of predicting the performance of and comparing different multiprocessor organizations. The problem is such simulations take enormous amounts of computer time even when a system model is not very detailed. This is not surprising as uniprocessor simulations using large realistic benchmarks can be extremely time consuming. Multiprocessor simulation complexity grows at least as fast as the number of processors in the system, and in reality quite a bit faster due to interaction between processors. Parallel simulation is one approach to increasing the speed of simulations of computer systems. It requires a parallel system to execute the simulator to achieve the speedup. We have used additional CEs purchased from Alliant to increase the amount of hardware parallelism to improve performance.

Two main parallel simulation activities we have at CSRD are 1) simulating a class of systems represented by Cedar, and 2) developing a parallel execution system for a high-level system description language. The first activity represents a special-purpose parallel simulator that can only be re-configured to model one class of systems. The second one is a part of an environment project that has as its goals ease of description, model reuse, and fast evaluation of different architectures. In both cases the underlying idea is to use a register-transfer level description of a system and exploit parallelism available in hardware at this level. Both systems are hierarchical, however, and do not require everything to be modeled at this level.

The special-purpose simulator has been used to simulate shared memory multiprocessor systems with interleaved memory system and two unidirectional interconnection networks. We have modeled the effect of various parameters for a range of systems sizes concentrating on from 32 to 512 processors. Both synthetic and trace-driven workloads have been used and the performance of the two models correlated. System parameters varied include network organization, memory interleaving and speed, data path and message length, burst traffic characteristics, conflict resolution, etc [Turn89]. We are in the process of verifying the simulator by modeling the current Cedar configuration and comparing the results with actual measurements.

The general-purpose parallel simulator is based on the ideas and experience derived from the special-purpose case. We currently have a prototype execution system developed and are evaluating its performance. One of the main questions we are trying to answer is the amount of parallelism one can find and exploit on average in simulating a multiprocessor system. The experience with special-purpose simulation and preliminary results with the general-purpose system are very encouraging. The mechanism used for the parallel evaluation and the data structures involved can have a major impact on performance. We are considering alternatives and will evaluate their behavior and performance.

II. Automatic Restructuring and Concurrent Evaluation of Lisp

II.1. Parcel

Parcel is an investigation of the compilation of sequential Scheme programs for execution on a shared-memory multiprocessor. The Parcel system has two components: a parallelizing compiler, that produces from a sequential Scheme program a machine-independent parallel intermediate form, and a code generator/run-time system, via which the intermediate form is expanded into machine instructions for the Alliant FX/8 and linked to a parallel run-time system to form an executable.

II.2. The Parcel Compiler

The Parcel compiler operates in three phases: interprocedural analysis, standard transformations, and parallelizing transformations.

Interprocedural analysis in Parcel takes the form of an abstract interpretation of the program. The abstract domains over which this interpretation takes place are built upon a mechanism for recording the interprocedural movements made by dynamically allocated objects. In terms of these movements, the analysis determines the visibility of any side-effects upon an object, and determines the lifetime of the object, in such a way that it may be placed on a stack or in a hierarchical memory as its required lifetime permits. This analysis is described in detail in [Harr89].

A number of standard transformations are performed upon each procedure of the program by Parcel, prior to parallelization. These transformations are optimizations that are beneficial from the standpoint of both parallel and sequential execution. For example, procedure integration is performed because it both eliminates the expense of procedure invocation, and makes code motion during parallelization more effective, by allowing it to be applied to a larger procedure body. Similarly, invariant floating eliminates needless repetition of an evaluation, and may at the same time remove spurious control dependences from a procedure.

A number of parallelizing transformations tailored specifically to symbolic programs originated in Parcel. For example, we introduced a quite general means of paralleling exit loops (generalized While/Repeat structures) by means of boolean recurrences in conjunction with loop distribution. A transformation called recursion splitting is used to rewrite self-recursive procedures as a complementary pair of loops, one of which performs the downward portion of a recursive computation, the other of which performs the upward portion. These transformations are described in [Harr89, HaPa88].

II.3. The Parcel Run-time System

From every procedure of a program, Parcel produces two transformed versions, one sequential and the other parallel. These two versions are presented, in intermediate form, to the code generator, which expands them into machine code and links them to the run-time system to form an executable.

The run-time system has, in addition to the usual sequential functionality of a Scheme run-time system, a microtasking library for the scheduling of parallel activity, a parallel storage allocator and deallocator (garbage collector), and a library of parallel recurrence solvers, for recurrences over numeric and symbolic data that are recognized and isolated by the compiler.

The demands upon the Parcel microtasking library are quite strenuous, for two reasons. First, the parallel codes produced by the compiler contain deeply nested, often recursive parallelism; second, the shape and granularity of the parallel computation varies dramatically from one program to another, and even from one data set to another. It is the responsibility of the microtasking library to select between parallel and sequential procedure versions to achieve a sufficient degree of parallelism and load balancing while retaining as much of the efficiency of the sequential computation as possible. More stacks that processors are used by the microtasking system, to break the traditional linkage between processors and stacks, for more flexible and efficient scheduling of parallel threads of activity. The microtasking system of Parcel is described in [ChHa90].

II.4. Miprac

Miprac is a successor to Parcel, in which the interprocedural analysis and transformations developed for Scheme programs in Parcel are fused with older techniques for the analysis and restructuring of Fortran programs, to yield a flexible tool for the analysis and restructuring of procedural languages generally. In the past, the intermediate form upon which a parallelizing compiler operated had a primarily syntactic correspondence to the program being compiled. For example, the intermediate form of Parafrase and KAP is essentially a syntax trees of the source program. We propose, however, an intermediate form with a primarily semantic correspondence to the program being compiled. To this end, we have designed MIL (Miprac Intermediate Language) to have three types of constructs: control (iteration, condition, sequence, branch), memory access (read, write, synchronization, file manipulation), and operations (addition, multiplication, etc). These three types of constructs correspond closely to common semantic domains, so that to rewrite a program into MIL is very much like giving the semantics of the program in terms of prescribed semantic domains. We are implementing translators from Fortran 77, C, and Scheme into MIL, in order to demonstrate its flexibility.

Because MIL allows branching, it has a non-compositional semantics (a continuation semantics). However, in Miprac most transformations apply to NMIL (Normalized MIL) in which branching has been eliminated by control-flow normalization. NMIL has compositional semantics, so that dataflow analysis and transformation of a NMIL program is straightforward to implement and verify. The control-flow normalization used in Miprac derives from that introduced in PAF [TaDF88], and is described in [Amma89].

Interprocedural analysis in Miprac is an extension of the analysis performed in Parcel to permit arithmetic on pointer variables (generalized access in blocks of dynamically allocated storage) as well as closures (higher-order procedures with side-effects). Because it is essentially typeless, MIL easily accommodates programs written in languages like C and Fortran in which type casting and outright type violations are commonplace.

III. Parallel Iterative Solver for Sparse Nonsymmetric Linear Systems

Practical iterative solvers for large sparse nonsymmetric linear systems $Ax=b$ usually require A to satisfy special properties, such as having a positive definite symmetric part or all eigenvalues with positive real parts. One class that guarantees convergence for any nonsingular matrix A are *row projection* (RP) methods, iterative algorithms that partition the rows of A into blocks $A^T=[A_1,A_2,\ldots,A_m]$ and then compute orthogonal projections onto range(A_i) ($i=1,\ldots,m$) on each iteration. Two general classes of RP methods have been examined, the product or Kaczmarz [Kacz39] and sum or Cimmino [Cimm39] forms. Both iterative forms are accelerated using conjugate gradients, as initially suggested by A. Björck and T. Elfving in [BjEl79] and implemented for two-dimensional problems by C. Kamath and A. Sameh in [Kama86].

Several theoretical and numerical results have been obtained for RP methods in the last year. Among them are:

• The limit points of RP methods have been characterized, for arbitrary choices of iteration parameters and block row partitionings [Bram89]. RP methods converge even when the system is singular or rectangular, and for least squares problems it is important to know what 'solution' the method finds.

• The choice of ω=1 has been shown to be optimal for the iteration parameter for both theoretical and computational reasons [KaSa88, BrSa89a].

• The underlying connections between the sum and product RP forms and CG applied to the normal equations have been discovered. This provides a basis of comparison of the methods, as well as one guide for choosing block row partitionings.

• By combining the properties of orthogonal projectors with those of the CG algorithm, one projection per CG iteration can be eliminated by using a special choice of the starting vector [BrSa88].

• A new RP method has been developed [BrSa89a]. CG acceleration of this system provides an error–minimizing algorithm, and the new system also allows an explicit reduction in the problem size.

• An error analysis of the methods showed that it is essential to compute the projections accurately. This in turn restricts the types of block row partitionings allowable.

• An approach to finding row partitionings that allow parallelism in the computations has been developed [BrSa89b]. This approach is based on a domain decomposition idea, and depends only on the discretization operator used and not on any regularity of the domain of the PDE.

The last item yields RP methods implementations that are frequently faster than other solvers, because for problems on an $n \times n \times n$ mesh, e.g., they provide $O(n^2)$ parallelism for speed and work on subproblems of size $O(n)$, giving good data locality.

	N=13824						N=216000					
Method	1	2	3	4	5	6	1	2	3	4	5	6
KACZ												
V–RP			MX						MX			
CIMM									MX			
GMRES			RS	RS					RS	TM		
ILU		RS	RS	RS		RS		UP	UI	UP		
MILU	RS	RS	RS		RS	RS	RS	UP	RS		RS	UP
CGNE			MX	MX					MX	MX		
ILCG		UP	MX	MX		MX		UP	TM	UP		TM

Table 1: Failures Among Methods

RP methods have proven to be more robust numerically than other commonly used nonsymmetric iterative algorithms, including the theoretically robust method of conjugate gradients (CG) applied to the normal equations $A^T Ax = A^T b$ and Krylov subspace methods. Table 1 shows the robustness results of testing with six three–dimensional non–selfadjoint elliptic PDE's from [BrSa89a], and that source should be consulted for the particulars of the equations and discretizations used. KACZ is the product form RP method, CIMM the sum form, and V–RP the newly developed RP method. GMRES refers to GMRES(10), CGNE to CG on the normal equations, (M)ILU to GMRES(10) preconditioned with (M)ILU, and ILCG to an ILU preconditioned CGNE. All were tested on systems of order N, with N given in the table. Error conditions are given by the following codes:

- MX: Maximum iterations reached
- UP: Unstable preconditioner
- RS: Residual stagnates
- TM: Maximum allowed CPU time reached

Clearly the RP solvers have a robustness unmatched by the other methods tested.

References

[Amma89] Zahira Ammarguellat, *Control-Flow Normalization of Programs,* CSRD Technical Report #885, Center for Supercomputing Research and Development, University of Illinois – Urbana, 1989.

[Bram89] R. Bramley, *Row Projection Methods for Linear Systems,* CSRD Technical Report #881, Center for Supercomputing Research and Development, University of Illinois – Urbana, 1989.

[BjEl79] A. Björck, T. Elfving, *Accelerated Projection Methods for Computing Pseudo-inverse Solutions of Systems of Linear Equations,* BIT, 145–163, 19(1979).

[BrSa88] R. Bramley and A. Sameh, *A Robust Parallel Solver for Block Tridiagonal Systems,* CSRD Technical Report #806, Center for Supercomputing Research and Development, University of Illinois – Urbana, 1988.

[BrSa89a] R. Bramley and A. Sameh, *Row Projection Methods for Large Nonsymmetric Linear Systems,* submitted to SIAM J. Sci. Stat. Comp., September 1989.

[BrSa89b] R. Bramley and A. Sameh, *Domain Decomposition for Parallel Row Projection Algorithms,* submitted to Applied Numer. Math., October 1989.

[ChHa90] Jyh-Herng Chow and Williams Ludwell Harrison III, *Microtasking for Recursive,* Parallel Programs, CSRD Technical Report, 1989 (work in progress).

[Cimm39] G. Cimmino, *Calcolo Approssimato per le Soluzioni dei Sistemi di Equazioni Lineari,* Ric. Sci. Progr. tecn. econom. naz., 326–333, 9(1939).

[HaPa88] Williams Ludwell Harrison III and David A. Padua, *PARCEL: Project for the Automatic Restructuring and Concurrent Evaluation of Lisp,* Proceedings of the 1988 International Conference on Supercomputing, ACM, July 1988.

[Harr89] Williams Ludwell Harrison III, *The Interprocedural Analysis and Automatic Parallelization of Scheme Programs,* Lisp and Symbolic Computation: an International Journal, Vol 2 No 3, 1989.

[Kacz39] S. Kaczmarz, *Angenä*herte Auflösung von Systemen Linearer Gleichungen, Bull. intern. Acad. polonaise Sci. lettres (Cracouie); Class sci. math. natur.: Seira A. Sci. Math., 355–357 (1939).

[Kama86] C. Kamath, *Solution of Nonsymmetric Systems of Equations on a Multiprocessor,* CSRD Technical Report #591, Center for Supercomputing Research and Development, University of Illinois - Urbana (1986).

[KaSa88] C. Kamath and A. Sameh, *A Projection Method for Solving Nonsymmetric Linear Systems on Multiprocessors*, Parallel Computing, 291–312, 9(1988/1989).

[KLSD84] David Kuck, Duncan Lawrie, Ahmed Sameh and Edward Davidson, *Construction of a Large-Scale Multiprocessor*, September 1984.

[TaDF88] N. Tawbi, A. Dumay, P. Feautrier, *PAF: un Paralleliseur Automatique pour Fortran*, Laboratoire MASI - University of Paris 6, #185, May 1987.

[Turn89] Stephen Wilson Turner, *Shared Memory and Interconnection Network Performance for Vector Multiprocessors*, CSRD Technical Report #876, Center for Supercomputing Research and Development, University of Illinois - Urbana, May 1989.