cu 508845
N89-29784
511-39
88
211152

THE FORCE
A Portable Parallel Programming Language
Supporting
Computational Structural Mechanics

November 18, 1987 Progress Report

Principal Investigator:
Harry F. Jordan (harry@boulder.colorado.edu)
Research Assistant:
Muhammad S. Benten (benten@boulder.colorado.edu)
Associated Personnel:
Juergen Brehm
Aruna Ramanan

Computer Systems Design Group
Electrical and Computer Engineering Department
University of Colorado
Boulder, Colorado 80309-0425

Project Summary

This project supports the conversion of codes in Computational Structural Mechanics to a parallel form which will efficiently exploit the computational power available from multiprocessors. The work is a part of a comprehensive, Fortran-based system to form a basis for a parallel version of the NICE/SPAR combination which will form the CSM Testbed. The software is macro-based and rests on the "force" methodology developed by the principal investigator in connection with an early scientific multiprocessor. Machine independence is an important characteristic of the system so that retargeting it to the Flex/32, or any other multiprocessor on which NICE/SPAR might be implemented, is well supported. The principal investigator has experience in producing parallel software for both full and sparse systems of linear equations using the force macros, and other researchers have used the Force in finite element programs. It has been possible to rapidly develop software which performs at maximum efficiency on a multiprocessor. The inherent machine independence of the system also means that the parallelization will not be limited to a specific multiprocessor.

THE FORCE LANGUAGE

- A Fortran based parallel programming language
- For shared memory multiprocessors

  Flexible Computer Corp. Flex/32   Encore Multimax
  Sequent Balance   Alliant FX/8
- Independent of number of processes
- Parallel execution of loops, cases, subroutines
- Synchronization: barrier named critical sections producer/consumer
- Management of variables:

	Shared	Private
(local)	×	×
Common	×	×

- Dynamic creation of parallel work
- Efficient implementation of primitives
- Language description and manual available
- Shell scripts for compilation, execution

The Force[1] is implemented as a macro preprocessor on the Flex/32[2] multiprocessor located at LaRC. Performance of matrix multiply and Gaussian elimination were reported at the August 25, 1986 CSM Grants Review. The complete documentation is in the Force User's Manual[3] published as a technical report in October 1986 and most recently revised in October 1987.

The software consists of Unix stream editor scripts, macro definitions for m4, and Unix shell scripts to invoke the software and interface it to Flex/32 compilation and execution. The whole is a parallel extension to Fortran, compatible with systems on the Encore Multimax, Sequent Balance 8000 and Alliant FX/8 multiprocessors. Force programs can be run unchanged on any of the machines.

Dense matrix algorithms in the Force show close to linear speed up when run with from one to 18 processors. The maximum speed of matrix multiply was 1.1 MFLOPS while that for the more complex Gaussian elimination was 0.54 MFLOPS. Both were for unoptimized Force programs. Experience shows that higher speeds can be obtained by optimization techniques such as loop unrolling.

Efficiency concerns lead to careful analysis and redesign of existing macros, such as the thorough analysis and optimization of the barrier by Arenstorf[4].

CSM Applications of the Force

- Parallelize RED module from SPAR's INV processor
  - Simple, low-level parallelization; no redesign
  - Speedup of 3+ on 8 Flex/32 processors

Other Groups:

- Gene Poole wrote Conjugate Gradient for Flex/32 in Force

- Charbel Farhat has numerous FE codes written in Force
  - Profile solver
  - Element by element computations
    - . Linear
    - . and Nonlinear
  - Preconditioned conjugate gradient
  - Block asymmetric factorization
  - Homotopy equations
  - Eigenvalue solver

The identical source for these has been run on:

- Encore Multimax
- Alliant FX/8
- Sequent Balance
- Cray 2

The Force on the Flex/32 supports parallelization of SPAR in the testbed system. Parallelization can be attacked two ways: confine parallelization to low level modules or redesign a parallel SPAR. The first was applied as a learning exercise, but as expected, did not yield large performance increases. For significant gain in performance, the whole program must be parallelized. This requires a thorough understanding of the structure of SPAR, which is not well documented. An analysis of data dependencies will be needed to implement SPAR on any multiprocessor.

The RED module from SPAR's INV processor was parallelized using the low-level approach which was primarily characterized by the replacement of sequential DO loops with parallel DOALLs. Without algorithm redesign, this simple approach yielded a speedup of just more than 3 using 8 processors of the Flex/32.

Others have used the Force to implement newly designed parallel algorithms for structural mechanics. One of the major users has been Charbel Farhat of the University of Colorado Center for Space Structures and Controls. Dr. Farhat has found the Force useful to write numerous finite element codes so that they can be run unchanged on several different multiprocessors.

PROGRESS

Since our last CSM Grants/Contract Review
on August 25-26, 1987

- A new Force Manual has been published, including:
  - Askfor DO for dynamic work generation
  - Self-scheduled parallel case macro
  - Continuation lines
- The Force has been ported to the Cray 2
  - Simple test cases run correctly
  - Charbel Farhat has run substantial programs

    ...The last known bug has been corrected

A major improvement in the Force as a complete parallel programming language is support for dynamic generation of parallel work. While many scientific codes can be written without this capability, it is of use in adaptive and search type algorithms. It can be explicitly coded by the user in the original version of the Force, but this is an involved and error prone process. The Askfor DO macro was developed to provide a basic level of support for this capability without altering the structure of the language in any major way.

Efficiency of Force constructs has not only been improved by the revision of the Barrier mentioned previously, but also a new macro for parallel case execution has been introduced. By allowing the work to be self-scheduled instead of prescheduled, this macro can improve efficiency of the parallel case in certain contexts. Convenience of the system has been extended by allowing the Force macros to use continuation lines.

As a result of interest at our last grantees review, the Force has been ported to the Cray 2. The implementation has not been completely tested, but it runs simple test cases correctly and there are no known problems at this time. Some substantial codes written by Charbel Farhat have been run on the Cray 2 using the system, and that part of the Force which he uses seems to be correct.

REFERENCES

[1] H. F. Jordan, "The Force," in *The Characteristics of Parallel Algorithms,* L. H. Jamieson, D. B. Gannon and R. J. Douglass, Eds., Chap. 16, MIT Press (1987).

[2] H. F. Jordan, "The force on the Flex: Global parallelism and portability," to appear in *Parallel Processing and Medium Scale Multiprocessors,* Arthur Wouk, Ed., SIAM, Philadelphia, PA, 1987.

[3] H. F. Jordan, M. S. Benten, N. S. Arenstorf and Aruna V. Ramanan, "Force User's Manual," *ECE Tech. Rept. 86-1-4R,* Computer Systems Design Group, Dept. of Electrical and Computer Engineering, University of Colorado, Boulder, CO, 80309-0425 (Revised Oct. 1987).

[4] N. S. Arenstorf and H. F. Jordan, "Comparing Barrier Algorithms," *ECE Tech. Rept. 87-1-2,* Computer Systems Design Group, Dept. of Electrical and Computer Engineering, University of Colorado, Boulder, CO, 80309-0425 (June 1987).