ANL-79-68

ANL-79-68

PRODUCTION RUNS ON THE CRAY-1

by

Larry Rudsinski
with
Gail W. Pieper

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Prepared for the U. S. DEPARTMENT OF ENERGY
under Contract W-31-109-Eng-38

DISCLAIMER

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

Distribution Category:
Mathematics and Computers
(UC-32)

ANL-79-68

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

PRODUCTION RUNS ON THE CRAY-1

by

Larry Rudsinski*

with

Gail W. Pieper

Applied Mathematics Division

July 1979

*Current address: 14520 Raneys Lane, Orland Park, Illinois 60462

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

Under a grant from the National Center for Atmospheric Research, users from several Argonne divisions executed programs on the CRAY-1. The objectives were twofold: (1) to determine the feasibility of implementing large codes on the CRAY-1 and (2) to decide what algorithm changes could be made to improve run time on a vector machine. In addition, we hoped to evaluate the code produced by the new CRAY FORTRAN compiler.

Large reactor codes were implemented with relative ease, requiring, for example, approximately two man-weeks to get a 56,000-card program running correctly. Certain difficulties were encountered, however, in transferring the large binary files required by a biomedical code.

We conclude that it is both feasible and desirable to implement large codes on the CRAY-1. Better instruction scheduling in the CRAY compiler has reduced run times by 40 percent. With some rewriting in Assembler language to produce optimal code, the programs tested ran two to four times faster than on the IBM 370/195 or the 3033 at Argonne. Further improvement (up to twofold) was realized in several programs by substituting new algorithms to take advantage of the vector architecture of the CRAY.

PRODUCTION RUNS ON THE CRAY-1

by

Larry Rudsinski

with

Gail W. Pieper

I. INTRODUCTION

Argonne National Laboratory (ANL) is engaged in a project to determine the impact of advanced scientific computers on Argonne's computing workload. The CRAY-1 was chosen to begin our investigation because it was the only machine currently available that qualifies as an advanced scientific computer, or Class VI (one capable of executing 20 to 60 million floating point operations per second). Under a grant from the National Center for Atmospheric Research (NCAR), users from several Argonne divisions executed test programs on the CRAY at NCAR and made performance comparisons with the IBM 370/195. The results of the investigation were published in ANL Report 79-9, entitled *Evaluating Computer Program Performance on the CRAY-1.*[1]

Motivated by the success of the original project, we applied for and received an additional grant from NCAR to continue our study of user programs in greater depth. Our investigation focused on two areas: the feasibility of putting large codes on the CRAY-1 and the implementation of new algorithms to exploit the vector hardware design of the CRAY-1. In addition, we paid careful attention to the code produced by the CRAY FORTRAN (CFT) compiler to evaluate the effectiveness of the new scalar coding instructions.

Production runs were carried out for two large reactor codes, and two other programs were successfully loaded and compiled before the project had to be terminated. We found that relatively little effort was required by Argonne scientists to get these codes running on the CRAY-1. With only minor modifications, the codes ran two to four times faster on the CRAY-1 than on the IBM 370/195; and further improvement (up to twofold) was realized when portions of the codes were rewritten to exploit vector capabilities.

II. USE OF THE NCAR SYSTEM

For the past year, we have been using the NCAR system in Boulder, Colorado, to evaluate computer program performance on the CRAY-1. At NCAR, timesharing is not available; instead, our jobs were transported, and a CRAY version of UPDATE (a batch editor) was used to maintain the source on the NCAR system.

In transporting the files, we depended heavily on the Remote Job Entry Station in Argonne's Applied Mathematics Division. This station made initial communications with NCAR possible within a few days, whereas setting up our own offsite computing station would have taken months.

AMD's Remote Job Entry Station consists of a VARIAN computer with a tape drive. The station uses the UT200 protocol to communicate with NCAR. Rather than reading large source files by punched cards, which is the typical way programs are transmitted on UT200 terminals, we were able to write source files in ASCII on nine-track tapes and use the tape drives on the VARIAN to transmit the file.

Our current project, in particular, put the station and the communication lines to a stringent test, in that we shipped very large files (13,000 to 27,000 records), requiring the link to stay up for one to three hours at a time.

The major restriction imposed by the Remote Job Entry Station was on the record block size of the ASCII tapes. Thus the DIF3D code--consisting of 43,000 cards--had to be broken down into two files before being sent to NCAR.

Our experience indicates that remote access to the NCAR system is adequate for developing, testing, and executing both small and large user jobs. The NCAR CRAY-1 has the necessary hardware and software facilities--including a FORTRAN compiler, a relocatable loader, a capability to stage files between the CRAY-1 and a front-end machine, and a system utility for updating an object file by replacing individual routines with new versions.

III. PRODUCTION RUNS

Several large codes were compiled on the CRAY-1, and the performance of two codes was compared with the IBM 370/195 and the IBM 3033.

A. SAS (Reactor Analysis and Safety)

SAS codes are utilized by the Reactor Analysis and Safety (RAS) Division to analyze the consequences of hypothetical accidents in fast breeder reactors. Currently, limitations in computer capability are inhibiting SAS code extensions to two- or three-dimensional models. Because these codes account for 5 to 10 percent of Argonne's computing usage, RAS scientists were interested in determining whether the CRAY-1 would be significantly faster than the IBM 370/195. Timing studies of one module from the SAS family indicated that a more substantial test with the whole SAS3D code (56,000 cards) would be desirable.

To put SAS3D on the CRAY-1 at NCAR, we first had to write a FORTRAN source tape using the ASCII character set required by NCAR. This tape contained three files: the source for the UPDAT program needed to modify SAS3D, the SAS3D common blocks, and the SAS3D source file. The resultant code was compiled, and both the source file and the object file were stored on CRAY-1 disks.

Getting the code to run on the CRAY-1 required a few changes to SAS3D. For example, the CDC version has to be overlaid; the CRAY, however, has ample memory to store the entire program. Consequently, all of the overlay cards were removed, as well as a special subroutine OVERLAY, and the program executed as a non-overlaid job. Other modifications involved replacing the timing routine and changing the routines for data-pack storage and retrieval; in addition, a few non-standard separators were found and replaced in FORMAT statements. Approximately one man-week was needed to complete these changes and to execute the tape successfully at NCAR.

Two different SAS3D cases were run on the CRAY-1. The first involved the initial 300 time steps of a one-channel low-power boiling case (LOWBLA). This run tested only the pre-boiling parts of the code. The second case was a more extensive one-channel case: 1000 time steps for channel 1 of a 33-channel transient undercooling case. This case extended into sodium boiling, clad relocation, and fuel relocation.

After the initial runs with SAS, we determined that further improvement could be realized by rewriting three linear interpolation routines. The table scanning loop in these routines had been written in a somewhat convoluted manner to achieve loop-mode on the IBM 370/195. The logic required, however, degraded the performance on the CRAY, which will run a simple FORTRAN DO-loop version as a simple in-stack loop. Also, because the CRAY FORTRAN compiler uses only scalar instructions to compile the loops in the interpolation routines, further speed improvement was achieved with CRAY Assembler Language (CAL) versions of these routines, which were written to use the vector compare instructions on the CRAY-1.*

Table I gives the times required to run both the LOWBLA and the one-channel case on Argonne's IBM 370/195 and on the NCAR CRAY-1. The last column, CRAY-1 CAL, refers to runs using CAL versions of the three interpolation routines.

*SAS3D modifications and timings were provided by F. Dunn

TABLE I

SAS3D Timing Comparisons (in seconds)

Case	IBM 370/195	IBM 3033	CRAY-1 CFT	CRAY-1 CAL
LOWBLA, 1 channel, 300 steps, no boiling	44.1	43.8	11.8	9.7
1-channel test, 1 of 33, 1000 steps	333.3	309.1	129.6	95.5

The FORTRAN version of SAS3D runs slightly faster on the IBM 3033 than on the 370/195 and a factor of 2.5 to 4 times faster on the CRAY-1 than on the 370/195. The use of the vector compare instructions, as well as generally tighter coding, in the CAL versions of the linear interpolation routines improves the overall running time of SAS3D by 20-25 percent.

B. DIF3D (Applied Physics)

DIF3D is a multidimensional multigroup finite-difference diffusion theory code which is used by the Applied Physics (AP) Division for reactor analysis. In the near future, its use is expected to escalate to consume between 5 and 10 percent of the ANL batch computer usage. Consequently, we were interested in determining the performance of DIF3D on the CRAY-1.

DIF3D has two major compute sections--an outer fission source iteration and a within-group iteration which uses optimized successive line overrelaxation (SLOR). The modular nature of DIF3D made it convenient to create a small kernel which accounts for 75 percent of all scalar execution time. The kernel largely consists of two routines (SORINV and ROWSRC) from the inner iteration section, which are easily implemented on any machine. Previous investigations showed that SORINV compiled with CFT cannot effectively exploit the CRAY-1 features because of the recursive nature of the SLOR algorithm employed.

Because SORINV accounts for about 60 percent of the computing time in DIF3D on the IBM 370/195 or CDC computers, scientists in the Applied Physics Division have been using an optimized Assembler language version of this routine on IBM and CDC computers. To obtain a reliable assessment on the CRAY-1, an optimized CAL version of SORINV was written.+ The CAL and IBM versions used identical SLOR algorithms.

The case of DIF3D that was timed involved 10 passes through a 50x50 mesh with 25 inner iterations. Table II gives the times obtained.

+The CAL routine and timings are the work of F. Dunn.

TABLE II

Run Times (in seconds) for a DIF3D Kernel

IBM 370/195	IBM 3033	CDC 7600 FTN4	CRAY-1 CFT	CDC 7600 COMPASS	CRAY-1 CAL
1.93	4.51	1.74	1.16	1.41	0.59

The CFT version of the DIF3D kernel runs 1.7 times as fast on the CRAY-1 as on the IBM 370/195. With the CAL version of subroutine SORINV, the code runs 3.8 times as fast on the CRAY-1 as on the 370/195. It is thus evident that substantial performance improvement can be made with inherently scalar code. The relatively poor performance on the IBM 3033 results from the fact that the 3033 has a relatively slow floating-point arithmetic unit, which limits the execution speed of SORINV.

C. **Bio-Medical Code** (Biological and Medical Research)

The Bio-Medical Code is a program originally developed for the Texas Instruments Advanced Scientific Computers, modified by Argonne's Biological and Medical Research (BIM) Division, and subsequently optimized by the Applied Mathematics Division to improve the run-time efficiency. The code required very little effort to compile correctly on the CRAY-1. However, problems were encountered in transferring two large binary files utilized by the code.

For the first binary file, we wrote a small FORTRAN program* for the IBM 370/195 that performed the appropriate binary reads followed by the formatted writes to tape; the process was then reversed on the CRAY-1, with the appropriate formatted reads followed by binary writes. The file required an 800-BPI ASCII tape with 27,000 records.

To generate the remaining file, which was more complex, we obtained from BIM the program that they use to generate the file. After several minor changes, it executed on the CRAY, giving slightly different results from those obtained on the 370/195; consequently, when the original program was executed, the job aborted with an error.

Thus, before we can carry out production runs, careful analysis of the second program will be necessary. This effort is beyond the time constraints of the current project.

D. **TRAC-P1A** (Engineering)

A program TRAC-P1A developed by Los Alamos Scientific Laboratory was obtained by Argonne's Engineering Division. To enable editing on the WYLBUR system at Argonne, the program (comprising 29,000 cards) had to be broken down into three parts. We then modified the CDC UPDATE directives and changed all deck names to be eight characters or less, as required by the CRAY 1 UPDATE program. Additionally, we deleted all the overlay structure, as we had with SAS3D.

The three files were then written onto three separate tapes, set up on the Remote Job Entry Station, and submitted to the CRAY-1, where UPDATE recombined them into a single program library.

*Work on the Bio-Medical Code was carried out in conjunction with Marianne Schiffer.

The code was compiled and loaded, and execution runs were attempted. Initial tests terminated with an error message from one of the TRAC-P1A error-processing routines. Work on correcting the error is in progress.*

*Work on TRAC-P1A was conducted in conjunction with Habib Ahmed.

IV. ALGORITHM CHANGES

An obvious question once a code is running on the CRAY-1 is what, if any, changes can be made to enhance the performance. Subroutine modifications such as those described above for the SAS code are often effective. Occasionally, however, improvement in run time can be realized only by changing the algorithm that the program is using. Below, we describe algorithm changes to the DIF3D and PIC codes, as well as to the square root function.

A. **DIF3D** (Applied Physics)

The nature of DIF3D is such that about 75 percent of all scalar execution time on the IBM 370/195 and the CDC 7600 computers is accounted for by the Successive Line Overrelaxation (SLOR) algorithm. This algorithm solves a pre-inverted tridiagonal system of equations by means of L-U decomposition, a procedure requiring numerous recursive operations that do not lend themselves to vectorization. Replacing the algorithm with a vectorized SLOR, involving odd/even line ordering on a plane,[2-3] resulted in an appreciable improvement in running speed, with a 10 percent increase in storage requirements.+

Table III shows the results of a benchmark test with a 50x50 mesh, using 25 inner iterations and repeated 10 times, for a total of 625,000 mesh cell iterations with 13 floating point operations per cell.

TABLE III

Relative Execution Rates (in units of 3.9 MFLOPS)
for a DIF3D Kernel

Method	CRAY-1	IBM 370/195
Scalar FORTRAN	1.8	1.0
Scalar FORTRAN & CAL	3.5	-
Vector FORTRAN	5.8	0.8

The encouraging CRAY-1 performance results attained with the DIF3D kernel indicated that an implementation of the entire DIF3D code (43,000 cards) was of interest. This implementation was accomplished with relative ease because DIF3D is designed and coded with portability to large-scale machines in mind. Major hardware characteristics such as long-word CDC-type machines or short-word IBM-type machines and one- or two-level memory hierarchy machines are accommodated by activating/deactivating coding that is bracketed by appropriate comment card keywords via a small preprocessing program. The coding invoked for the CRAY-1 was the one-level memory hierarchy storage coding and the CDC long-word coding. Because the CDC long-word brackets also invoke CDC-style entry points and the CDC overlay calling sequence, manual modifications with the WYLBUR text editor were required to obtain the correct entry point formats and to remove the overlay calls (as they were also with SAS). Several other minor changes were also needed to obtain the CRAY-1 library utility functions and to implement the machine-dependent segment of the dynamic storage allocation.

+Algorithm development and timings reported here were carried out by K. Derstine.

A two-dimensional two-group problem with a mesh of 170x170 (57,800 unknowns) was subsequently run to evaluate the entire DIF3D performance. Results are displayed in Table IV.

TABLE IV

DIF3D Problem Showing Relative Execution Rates
(in units of 3.9 MFLOPS)

Method	CRAY-1	IBM 370/195
Scalar FORTRAN	1.6	1.0
Scalar FORTRAN & CAL	3.7	-
Vector FORTRAN	5.3	0.8

Note that the vectorized SLOR required 20 percent fewer outer iterations than the scalar algorithm for the same convergence criteria in the two-dimensional problem.

Similar results were obtained with a three-dimensional variant of the problem. In this problem, the vectorized algorithm required 15 percent more outer iterations than the scalar algorithm. The vectorized SLOR code still ran 3.5 times faster on the CRAY-1 than the scalar algorithm on the IBM 370/195.

B. PIC (Chemistry)

Particle-in-cell trajectory (PIC) codes are used in Argonne's Chemistry Division to study ion-ion plasmas in applied external fields. Since our initial use of these codes on the CRAY-1 at NCAR, we were able to examine in greater depth ways to improve performance timings. Two algorithm changes have been implemented: the exploitation of symmetry and the elimination of out-of-bounds particles.*

1. Symmetry

In plasma simulation studies, if the initial conditions, boundary conditions, and applied field are symmetric, then the simulation particle distributions will retain the symmetry during the simulation. Thus, only a certain fraction of particles need be explicitly followed to generate the full particle distribution and the space charge forces. To take advantage of this feature in the problems under study at Argonne, which have either no symmetry or reflection symmetry about the x axis or the x and y axes, we modified the PIC code to track particles in the upper half or the upper right quadrant of the initial distribution. An example for x- and y-reflection symmetry is detailed below:

Given that QE(,J) is the charge distribution in the (I,J) cell for the explicitly followed particles and that MESHI and MESHJ are the maximum values of I and J, then the full charge distribution Q(I,J) is generated by the code

*We are indebted to Al Wagner for the documentation and data presented here.

```
      MESHJ1 = MESHJ+1
      MESHI1 = MESHI+1
      DO 1 J=1,MESHJ
      DO 1 I=1,,MESHI
      Q(I,J) = QE(MESHI1-I,J) + QE(MESHI1-I,MESHJ1-J)
               + QE(I,MESHJ1-J)+ QE(I,J)
    1 CONTINUE
```

For each time step in the example shown, the operations necessary to advance three quarters of the particles are replaced by simple additions for each charge cell. The savings are thus substantial if the full number of simulation particles is considerably larger than the number of cells.

Table V gives the times obtained on the CRAY-1 for a 40,000-particle system run over 50 time steps using the modified PIC, which readily vectorizes. As the table shows, taking advantage of symmetry improves execution time by 45 to 70 percent.

TABLE V

CRAY-1 Timings (in seconds per time step) for PIC
(40,000-particle system, 64x64 mesh)

Symmetry	Time
none	0.171
x reflection	0.093
x,y reflection	0.054

Further exploration of symmetry is possible in the Poisson equation solver. For example, the imposition of a zero slope boundary condition along the x axis or the x and y axes would shrink the region over which the solution must be found. Thus the charge distribution in only a fraction of the full space would need to be known, and the resulting solution in the reduced region could simply be reflected into the full space. The fast Fourier transform method currently used in PIC to solve Poisson's equation for the full space could be applied to solve Poisson's equation in the upper half of the space, but modifications of the code would be required; the method might also be applicable to solve Poisson's equation in the upper right quadrant of the space. Work in this area is under consideration.

2. Out-of-bounds Particles

In PIC simulation studies, particles that go out of bounds during a particular time step are flagged by the subroutine PARMOVE in such a way that, for subsequent time steps, operations on these particles are masked. If the vectors of particle positions and velocities were periodically condensed to include only in-bounds particles, these masking operations could be eliminated. Unfortunately, such condensation cannot be vectorized, because, in the loop over the index of the full vector, the index of the condensed vector is augmented only if the in-bounds test is positive. Thus, the savings from condensation are questionable.

To test the possible advantages, we modified PARMOVE to provide a current count of the total number of out-of-bounds particles and to condense the vectors of particle positions and velocities when the count exceeded a threshold value. The results are listed in Table VI for a 10,000-particle system run over 100 time

steps. The average execution time per time step is given as a funtion of IOUT, the minimum number of out-of-bounds particles accumulated before condensation. For this case, 9940 particles pass out of bounds during the simulation. Table VI indicates no savings realized; a considerably longer run may show small advantages.

TABLE VI

CRAY-1 Timings (in seconds per time step) for PIC
(10,000-particle system; 64x64 mesh)

IOUT	Time
64	0.043
128	0.043
512	0.042
1024	0.042

C. VSQRT Function (Applied Mathematics)

An analysis of the vector square root (VSQRT) function, which is supplied by CRAY, determined that its execution time could be significantly decreased without losing any accuracy. Traditionally, the square root is expressed as $X = 2 \cdot f$, where $1/4<f<1$. We chose instead to represent X as $X = 2 \cdot F$, where $1/2<F<2$. Then $x = 2 \cdot F$, and F can be calculated by either of the following methods: (1) a second-degree minimax polynomial approximation followed by three Newton iterations, the first two iterations using the 30-bit reciprocal approximation on the CRAY-1 and the final iteration using the full-precision divide; or (2) a fourth-degree polynomial followed by two Newton iterations, the first using the partial-precision divide and the second using the full-precision divide.

Based on hand timings, we estimate that the new algorithm will reduce execution time on the CRAY-1 by 40 percent.+

+The algorithm modifications were made by W. J. Cody.

V. CFT COMPILER

In our initial studies on the CRAY-1 in the summer and fall of 1978, we noted that the CFT compiler used at NCAR was not so sophisticated for scalar optimization as the FORTRAN H extended opt 2 but was being improved.

To test these improvements, RAS scientists reran the SAS3A heat transfer routine TSHTR with the new FORTRAN compiler on the CRAY-1.* Table VII shows the timing results.

TABLE VII

Timings (in seconds) of TSHTR

IBM 370/195	CRAY-1 CFT (1978)	CRAY-1 CFT (1979)
0.261	0.139	0.081

In this example, a 40 percent reduction in execution time was realized by improved scalar instruction scheduling. One cannot, of course, guarantee that all codes will improve by this amount.

Even this current version of the compiler, however, does not produce optimal code. Timing studies with the DIF3D routine SORINV indicate that the CAL version of SORINV runs more than twice as fast as the CFT version (see Table VIII).

TABLE VIII

CRAY-1 Timings (in seconds) of SORINV

CAL Version	CFT Version
0.46	0.98

Nevertheless, because of its speed, the CRAY-1 can outperform an IBM 370/195, even if CFT does not produce very efficient code. CFT compiles programs very quickly: It takes only 13.5 seconds of CPU time to compile the whole SAS3D code, whereas the IBM 370/195 FORTRAN extended opt 2 requires 364.1 CPU seconds.

*TSHTR timings were provided by F. Dunn.

VI. SUMMARY AND CONCLUSIONS

The primary objective of this investigation was to determine the feasibility of running large FORTRAN codes on the CRAY-1. We have demonstrated that the CRAY can compile, load, and execute codes such as SAS3D and DIF3D with relative ease. However, the difficulties encountered with the Bio-Medical Code suggest that certain programs which rely upon other data sets and are not designed to be portable may require more extensive work to run successfully.

A second objective was to develop alternate solution strategies that utilize the vector and parallel hardware design of the CRAY-1. The codes translated to the CRAY ran two to four times faster than on the IBM 370/195. With extensive rewriting of the algorithms, execution was reduced further by a factor of two.

The ability of the CRAY-1 to handle complex codes has generated predominantly favorable response from Argonne scientists. The investment of effort required was generally less than anticipated, and the improvement in program running times was substantial.

REFERENCES

1. L. Rudsinski with G. W. Pieper, *Evaluating Computer Program Performance on the CRAY-1*, Argonne National Laboratory Report ANL-79-9 (January 1979).

2. D. Boley, B. Buzbee, and S. Porter, *On Block Relaxation Techniques*, University of Wisconsin, Mathematics Research Center Report 1860 (June 1978).

3. D. Boley, *Vectorization of Block Relaxation Techniques: Some Numerical Experiments*, Proceedings of the 1978 LASL Workshop on Vector and Parallel Processors, LA-7491 (September 1978).

ACKNOWLEDGMENTS

Most of the documentation and data for this report were provided by the following people: Keith Derstine (Applied Physics), Floyd Dunn (Reactor Analysis and Safety), Al Wagner (Chemistry), and Jim Cody (Applied Mathematics); we are grateful for their assistance. We also commend the operating staff of AMD's Remote Job Entry Station for facilitating our communications with NCAR.

Distribution for ANL-79-68

Internal:

G. W. Pieper (73)
M. Gibson (14)
A. B. Krisciunas
ANL Contract File
ANL Libraries (5)
TIS Files (6)

External:

DOE-TIC, for distribution per UC-32 (191)
Manager, Chicago Operations and Regional Office, DOE
Chief, Office of Patent Counsel, DOE-CORO
President, Argonne Universities Association
Applied Mathematics Division Review Committee:
P. J. Eberlein, SUNY at Buffalo
G. Estrin, U. California, Los Angeles
W. M. Gentleman, U. Waterloo
J. M. Ortega, North Carolina State U.
E. N. Pinson, Bell Telephone Labs.
S. Rosen, Purdue U.
D. M. Young, Jr., U. Texas at Austin