EXPERIENCE WITH A VECTORIZED GENERAL CIRCULATION WEATHER MODEL ON STAR-100

David B. Soll
Nadim R. Habra
Gary L. Russell

GTE Information Systems
Goddard Institute for Space Studies
2880 Broadway
New York, N.Y. 10025

Symposium on High Speed Computer and Algorithm Organization (HICOM) APRIL 13-15, 1977
ACADEMIC PRESS 1977

Abstract

A version of the Goddard Institute for Space Studies Atmospheric General Circulation Model was vectorized to run on a CDC STAR-100. The numerical model was coded and run in two different vector languages, CDC STAR FORTRAN and LRLTRAN. A factor of 10 speed improvement over an IBM 360/95 was realized. Efficient use of the STAR machine required some re-designing of algorithms and logic. This seems to preclude the application of vectorizing compilers on the original scalar code to achieve the same results. Although vector languages have not yet reached maturity, they permit a more natural and efficient formulation for such numerical codes.

INTRODUCTION

This paper presents the experience gained from converting the Goddard Institute for Space Studies' (GISS) general circulation weather model to run efficiently on a CDC STAR-100. The conversion effort involved both reorganization and redesign of the original scalar code and the use of two vector languages: CDC STAR FORTRAN and LRLTRAN. Included are timing comparisons and suggestions for improving the usability of vector machines. The result of the conversion of this model supports the suggestion by Stone(1) that simple, mechanical adaptations of serial algorithms are not necessarily efficient on non-serial machines.

THE MODEL

The particular model used was a coarse grid version of the nine layer GISS general circulation model(2) which in turn was derived from the three layer weather model developed by Arakawa and Mintz at U.C.L.A. This model performs a time integration of the "primitive equations" on a regular cylindrical grid with different time steps for each of the three principal sections: dynamics (winds, etc.), physics (transport equations), and radiation. The coarse grid consists of 24 meridians each of which contains 16 latitude points. Although the number of grid points is significantly smaller than current production versions of the model, it was felt that this representation was adequate to perform the conversion economically. At GISS, the model is programmed in FORTRAN IV and runs on an IBM 360/95 (a faster version of the 360/91).

CONVERSION TO THE STAR

There were several general considerations in the conversion effort. Foremost was that although the STAR vector pipeline is fast, scalar operations are not; consequently sections of the program had to be substantially reorganized to reduce the ratio of scalar to vector code. In fact, many scalar quantities were maintained in vector form, expanded over the entire grid. Moreover, small arrays were expanded where appropriate into longer vectors to make full use of the streaming efficiency of the vector hardware.

Software compatibility between the two vector languages available dictated the use of singly-dimensioned arrays for all vector operations. Furthermore, the storage order of all vectors was altered so that longitudinal points for each latitude were in contiguous storage. This was done primarily because the dynamics section contains different computations for polar and non-polar points. The result of this reorganization was that the different computations could be performed on continuous as opposed to scattered storage.

A second consideration was that several scalar algorithms did not lend themselves to direct vectorization and had to be rewritten. Circularity in the longitudinal computations is a recurring problem in the dynamics section. This was resolved by introducing two extra longitudinal points at each latitude thus creating an artificial wrap-around.

Another instance of algorithm redesign involved weighted sums of products of quantities defined at neighboring gridpoints. By reordering the computation, weighted sums of the individual quantities could be computed and the products formed afterwards. The latter method is more easily vectorizable than the former.

A simulation of the vectorized code was conducted in scalar FORTRAN to test the validity of the reorganized logic and the modified algorithms.

LANGUAGES AND CODING TECHNIQUES

Since the numerical model is considered to be a research tool, coding proceeded under the assumption that it should be possible to easily make future changes while maintaining readability. Consequently, it was decided that a high level language would be used throughout.

Two STAR vector languages were selected: (1) LRLTRAN,(3) written at the Lawrence Livermore Laboratory (LLL), University of California, and (2) CDC's STAR FORTRAN,(4) a standard program product. LRLTRAN is a programming language derived from FORTRAN containing most of the features of FORTRAN IV and, in addition, vector extensions.

Both languages offer an explicit vector syntax which permits the direct use of the STAR vector hardware. Even though both languages contain a syntax for embedded machine instructions (Q8INLINE), our readability constraint precluded their use. This constraint was slightly relaxed when all operations were coded as explicit dyads. This technique was introduced in order to prevent the generation of unnecessary temporary vectors by the compilers. Whereas LRLTRAN contains a more concise notation for vector addressing, it requires that all vectors be explicitly declared, singly-dimensioned, and have a starting index of zero. This tends to create some confusion when

EQUIVALENCE statements are used to map vectors onto other vectors or multiply-dimensioned scalar arrays. STAR FORTRAN, on the other hand, adheres more closely to ANSI FORTRAN standards. It allows multiply-dimensioned arrays to be used as vectors without an explicit declaration. Although STAR FORTRAN tends to be more verbose, the overall flexibility offered by LRLTRAN introduces additional "overhead" which naturally leads to longer execution times.

STAR EXECUTION AND TIMINGS

The dynamics section was coded in both languages, whereas the physics section was written only in STAR FORTRAN. This paper discusses primarily those two sections, as vectorization of the radiation section is not yet complete. Several runs of the model were made at three separate installations. At LLL, the LRLTRAN version of the dynamics was used, while the entire code in STAR FORTRAN was run at CDC's Data Center in Arden Hills, Minnesota. The timings for the original scalar code were measured on the 360/95 at GISS. A comparison of the different timings is presented in Tables I and II. The scalar code for the 360/95 was compiled under FORTRAN H Extended Plus, Optimization Level 2, and used 64 bit floating point arithmetic. The vector code was compiled under STAR FORTRAN version 2.0, cycle 115P without optimization.

As seen in Table I, the speed improvement for one call to the dynamics sections is almost twice that obtained from one call to the physics section. This is because the dynamics algorithms lend themselves more readily to vectorization. A run of the model for 14 simulated days yielded a net speedup factor of 10.23. While this figure includes several sources of overhead such as I/O and housekeeping, the effect of the slower physics is offset by the fact that the dynamics routines are called six times as often. A comparison of the dynamics written in LRLTRAN (CHAT STAR version 98D) versus STAR FORTRAN (Table II) confirms that the increased flexibility offered by LRLTRAN is achieved with some loss of performance.

OBSERVATIONS AND RECOMMENDATIONS

As a result of our experiences, several observations can be made. Vector codes generally require more memory than their scalar counterparts, primarily because intermediate quantities are now vectors rather than scalars. Since vector machines offer greater speeds, solving larger problems and increasing the resolution of current problems become feasible. It appears, therefore, that large main storage configurations are desirable, if not necessary. Experience with a similar model on ILLIAC IV(5) confirms that the performance potential of a fast machine may not be fully realized when the user must manage his own peripheral storage as an extension of main storage. This problem is solved on the STAR-100 by the use of a virtual memory operating system and peripheral stations(6) which perform most of the detailed input/output-related computing functions. The combination of large main memory and an efficient virtual operating system, then, seems to be the best way to utilize a fast parallel or vector machine.

It is critical that the underlying input/output control software handles the user I/O requests specified in higher level language statements as efficiently as it handles system I/O requests. In this respect and in spite of the available I/O facilities, unformatted FORTRAN I/O for this model on the STAR was found to be a factor of 3.5 times slower than on the 360/95.

Not only are machine/systems improvements necessary, but vector languages and compilers are also underdeveloped. As previously mentioned, both LRLTRAN and STAR FORTRAN suffer from a rigidity of notation which tends to make both languages more verbose. A combination of features offered by both languages would appear to be more appealing to the user. Code optimization, error detection, and debugging facilities offered by these compilers are not as sophisticated as one would expect on such a powerful machine as the STAR-100.

In view of the substantial program redesign that was necessary to produce an efficient vector code, there are serious doubts about the effectiveness of a simple, mechanical translation of serial code. In fact, we agree with Stone(1) that vectorizing compilers which would produce such translations would be "...stop gaps at best".

CONCLUSION

Since it appears that a vector formulation is a more "natural" expression of the problem, it would seem that as vector machines become more prevalent, increasing numbers of algorithms will be written directly in vector form. Therefore, it is our opinion that vectorizers now have and will continue to have only limited utility. Thus, the greater part of future effort in software development should be directed towards the vector languages themselves.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Dr. James Hansen of GISS for funding the project and for his constant supervision and guidance. Ms. Gail Whitten, Mssrs. Robert Cralle, Frank McMahon, and Stanley Solbeck of Lawrence Livermore Laboratory provided valuable assistance in the implementation. Mssrs. Arthur Lazanoff, Chet Berkey, and Harold Reppe of Control Data Corporation deserve special acknowledgement for their continued valuable assistance. Mr. John Laubenheimer of GTE Information Systems wrote sections of the code. The authors are also indebted to Mr. V.C. Devan, project manager at GTE/IS for his moral support.

DEDICATION

The authors would like to dedicate this paper to the memory of Mr. Ronald Karn of GTE/IS who, until his untimely death, was an important contributor to this project.

TABLE I. MACHINE COMPARISONS

	360/95(GISS) secs.	STAR(CDC) secs.	95:STAR
Dynamics*	0.65	0.056	11.61:1
Physics*	0.41	0.070	5.85:1
14-day Run	1472.31	143.92	10.23:1

TABLE II. LANGUAGE COMPARISONS

	LRLTRAN(LLL) secs.	STAR FORTRAN(CDC) secs.	LLL:CDC
Dynamics*	0.085	0.056	1.52:1

*Timings are for a single call.

REFERENCES

(1) Stone, H.S., "Problems of Parallel Computations" - Complexity of Sequential and Parallel Numerical Algorithms - (ed. Traub, J.F.) - Academic Press - 1973.

(2) Somerville, R.C.J., et al. "The GISS Model of the Global Atmosphere" - Journal of the Atmospheric Sciences, 31, No. 1, pp. 84-117 (1974).

(3) Martin, J.T., Zwakenberg, R.G., Solbeck, S.V. - Livermore Time-Sharing System, Pt. III, ch. 207 - LTSS-207 (Ed. 4) - December 1974.

(4) Control Data Corporation - FORTRAN Language Reference Manual - STAR-100 Computer System - No. 60386200 - 1976.

(5) Karn, R. - "Parallel Computing: Timing for ILLIAC IV" - Software, Practice and Experience, vol. 6, pp. 579-584 (1976).

(6) Control Data Corporation - STAR-100 Computer - Hardware Reference Manual - No. 60256000 - 1975.