Ames Grant
IN-62-CR
309058
P-13

Final Report for NASA Grant NCC 2-603
U.C.S.D. Number 90-1041
Ames Control Number 89-197

PARALLEL ADA BENCHMARKS FOR THE SVMS

Final Report
Prepared By

PHILIPPE E. COLLARD
University of California, San Diego
California Space Institute
Mail Stop A-021
La Jolla, CA 92093-0221
(619) 534-6369

Presented To

MR. ANDREW GOFORTH
NASA/Ames Research Center
Information Sciences Division
Mail Stop 244-4
Moffett Field, CA 94035

Submitted This Date
March 30, 1990

Period of Research Grant
April 1989 - April 1990

NASA Grant NCC 2-603
U.C.S.D. Number 90-1041
Ames Control Number 89-197

Grant Entitled: "PARALLEL ADA BENCHMARKS FOR THE SVMS".

NASA Technical Officer:

Mr. Andre Goforth
National Aeronautics and Space Administration
Ames Research Center
Information Sciences Office
Mail Stop 244-4
Moffett Field, California 94035

NASA Contracting Officer:

Ms. Barbara Hastings
National Aeronautics and Space Administration
Ames Research Center
University Affairs Office
Mail Stop 241-25
Moffett Field, California 94035

Report Prepared By:

Mr. Philippe Collard, Specialist, Principal Investigator
University of California, San Diego
California Space Institute
339 Nierenberg Hall/POSS
Mail Stop A-021
La Jolla, California 92093-0221
(619) 534-6369

UCSD/SIO - Office of Contracts & Grants

Mr. Norman Sattler, Contracts & Grants Officer
c/o Marian Crosser/Sandra Varond, Grants Analysts
University of California, San Diego
Scripps Institution of Oceanography
118 Scripps Building
Mail Stop A-010
La Jolla, California 92093-0210
(619) 534-1293

TABLE OF CONTENTS

I - Introduction

The use of parallel processing paradigm to design and develop faster and more reliable computers appears to clearly mark the future of information processing. NASA has started the development of such an architecture: the Spaceborne VHSIC Multi-processor System (SVMS). Ada will be one of the languages used to program the SVMS. One of the unique characteristics of Ada is that it supports parallel processing at the language level through the tasking constructs. It is important for the SVMS project team to assess how efficiently the SVMS architecture will be implemented, as well as how efficiently the Ada environment will be ported to the SVMS.

AUTOCLASS II, a Bayesian classifier written in Common Lisp, has been selected as one of the benchmarks for SVMS configurations. The purpose of our R&D effort was to provide the SVMS project team at NASA, Ames Research Center, with a version of AUTOCLASS II, written in Ada, that would make use of Ada tasking constructs as much as possible so as to constitute a suitable benchmark. Additionally, we had to develop a set of programs that would measure Ada tasking efficiency on parallel architectures as well as determine the critical parameters influencing tasking efficiency. All this was designed to provide the SVMS project team and NASA, Ames Research Center, with a set of suitable tools in the development of the SVMS architecture.

II - Results

The following was to be delivered to the SVMS project team:

- A design document detailing how the Autoclass program would be implemented in Ada as a concurrent process;
- A version of the Autoclass program implemented in Ada according to the design document specifications and fulfilling the project requirements;
- A users manual for the program;
- A set of programs, written in Ada, allowing the SVMS team to evaluate the performances of the parallel Ada run time on the SVMS.

As of this date, all of the above identified items have been delivered to NASA, with the exception of the users manual. The users manual is almost complete, and will be delivered as soon as possible to NASA personnel.

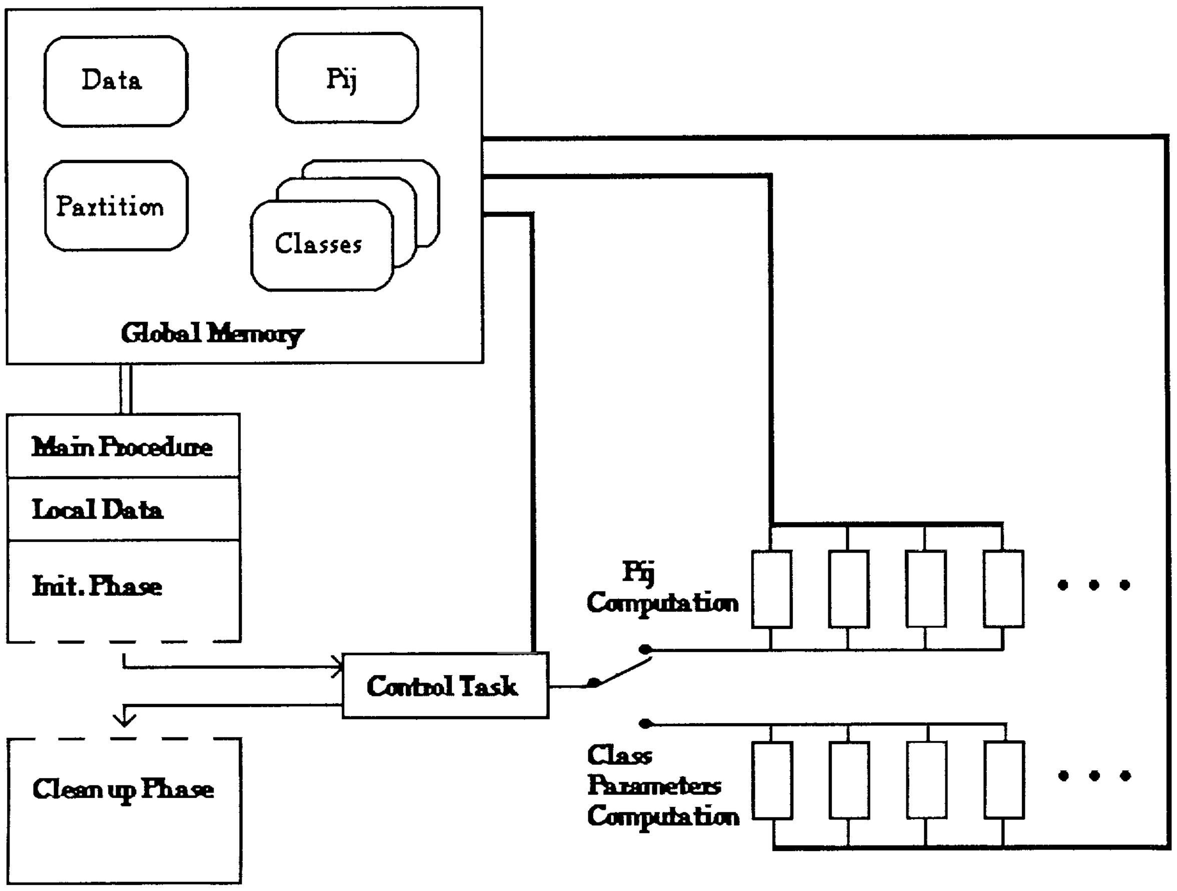
One of the most critical requirements for building a parallel Ada version of the Autoclass program was that it had to run 2.5 times faster with 4 processors than with 1, when processing a subset of the IRAS database. According to a model developed by Jordan [1] , this requirement meant that 80% of the program had to execute in parallel. The Autoclass algorithm is known to allow that much parallelism. The main design problem was to determine if an implementation of the program, in Ada, would permit us to take advantage of a sufficient amount of Autoclass parallelism to ensure that the requirement be met.

A design analysis was performed during the first two months of the project and a design document was submitted to NASA for review. After review by NASA personnel, the proposed design was accepted. Figure 1 is a diagram of the program architecture. The implementation phase called for the development of a prototype of the program where all the Ada tasking structures would be in place, but where all computational subroutines would be replaced with busy loops. This prototype was used to verify that the proposed design met the 2.5 times faster (with 4 processors than with one) requirement. Tests were conducted on a Sequent / Balance with 4 processors and a Sequent / Balance with 16 processors. The rational for using these systems as a test bed was that if the requirement was met on these slower, technologically older machines, it should be met on the SVMS systems which was to be a state-of-art VLSI system.

The Autoclass program is an iterative process composed of two mutually exclusive phases. It was decided that a task would be assigned to the processing of each class to be inferred in the first phase. For the second phase, a variable (user defined) number of tasks (called Pij tasks) would be assigned to the computation of the matrix of probabilities. This rather conservative design was to ensure that the number of Ada tasks could be handled by the Ada run time system while at the same time giving some flexibility so as to take advantage of the parallel architecture.

[1] Harry Jordan, "Interpreting Parallel Processor Performance Measurements", SIAM J. Sci. Stat. Comput., Vol 8, No 2, March 1987.

The test of the prototype allowed us to determine what was the optimum number of Pij tasks for a particular processor configuration. Figure 2 shows 2 curves. Each curve represents the time it took the prototype to execute the simulation for a given number of processors (one to 15). One curve was obtained with a number of Pij tasks equal to 4 and the other with a number of Pij tasks equal to 10. In both cases, it was demonstrated that the design would allow the 2.5 times faster criteria to be met. Additionally, one can notice that for low number of processors (<4), the two curves are very close but only the curve for 10 Pij tasks shows an improvement when the number of processors goes above 4 (since with 4 Pij tasks, no additional parallelism can be exploited). This show us that the design of parallel programs in Ada should account for potential scale up of the hardware configuration.



Main design features:

- **All data resident in global memory**
- **Maximum visibility among tasks**
- **One task per class (class parameters)**
- **Adjustable number of Pij computation tasks**
- **Computation cycle controlled by separate task**

Figure 1 - Architecture of the parallel Autoclass / Ada program

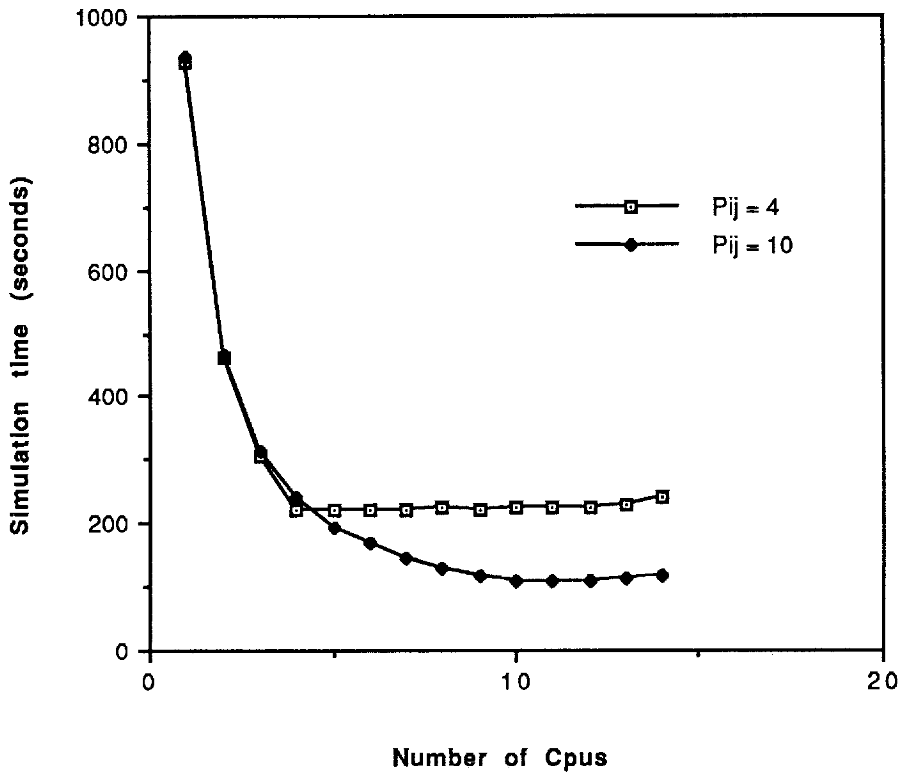


Figure 2 - Simulation time versus number of Cpus for prototype

Following the demonstration that the program was properly designed, the computational subroutines and I/O subroutines were incorporated into the program. A test case was provided comprised of 531 out of the 5300 objects of the IRAS databases. Each object is described by 95 variables. A number of classes equal to 20 was to be inferred through five iterations of the program computational loop. Tests were performed on the Sequent / Balance with 16 processors. Figure 3 shows the performance curve obtained when the number of processors used to run the program varies from 1 to 16. The curves is very regular. When plotted with Log/Log axes, it demonstrates that the program exhibits a linear speed up factor as a function of the number of processor used. Thus the program runs 4 times faster with 4 processors than with 1, 8 times faster with 8 processors, etc. This greatly exceeds the project requirement, and is actually the maximum performance that one can get out of a computer with a parallel architecture.

Following these results, tests were conducted on another parallel computer: a Sequent / Symmetry. This system is equipped with more recent microprocessor technology (Intel 80386). Again the results show greater performance improvements as the number of processors used increases, although some irregularity in the curve (Figure 4) can be observed. A series of experiments allowed us to determine that these irregularities can be attributed to the different scheduling algorithm used in the Sequent / Symmetry operating system.

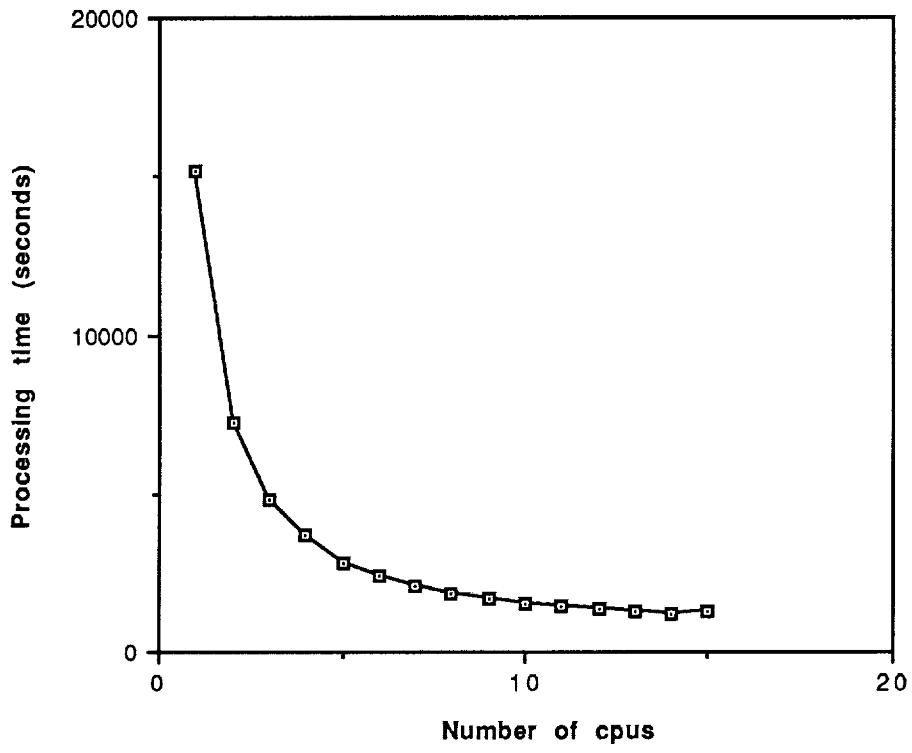


Figure 3 - Performance curve for parallel Autoclass / Ada on Balance

The version 3.1 of the program was delivered to NASA personnel. This version allows the processing of the entire IRAS database (5300 cases). The following files compose the set of Ada sources for the parallel Autoclass / Ada program:

• AUTOCLASSIIP31.ADA	the main program
• AUTOCLASS_TYPES.ADA	a package of type definitions
• AUTOCLASS_UTIL.ADA	a package with the Autoclass specific math library
• MATHFUNCTION.ADA	a package a special math functions
• ENVIRONMENT.ADA	a package defining the processing environment
• GENDEF.ADA	a package of ancillary type definitions
• MATRIXDEF.ADA	a package of ancillary type definitions
• VECTORS.ADA	a package for vector manipulations
• FILEIO.ADA	a package of data file I/O procedures
• USERIO.ADA	a package of terminal I/O procedures

The program has been compiled and executed on the following platforms: Vaxstation 2000 (VMS) + DEC/ADA compiler, Vax 8800 (VMS) + Telesoft compiler, MIPS (Unix) + Verdix compiler, Sequent / Balance (Unix) + Verdix compiler, Sequent / Symmetry (Unix) + Verdix compiler, Encore/Multimax (Unix) + Verdix/Encore compiler. This demonstrates the high level of portability of the program.

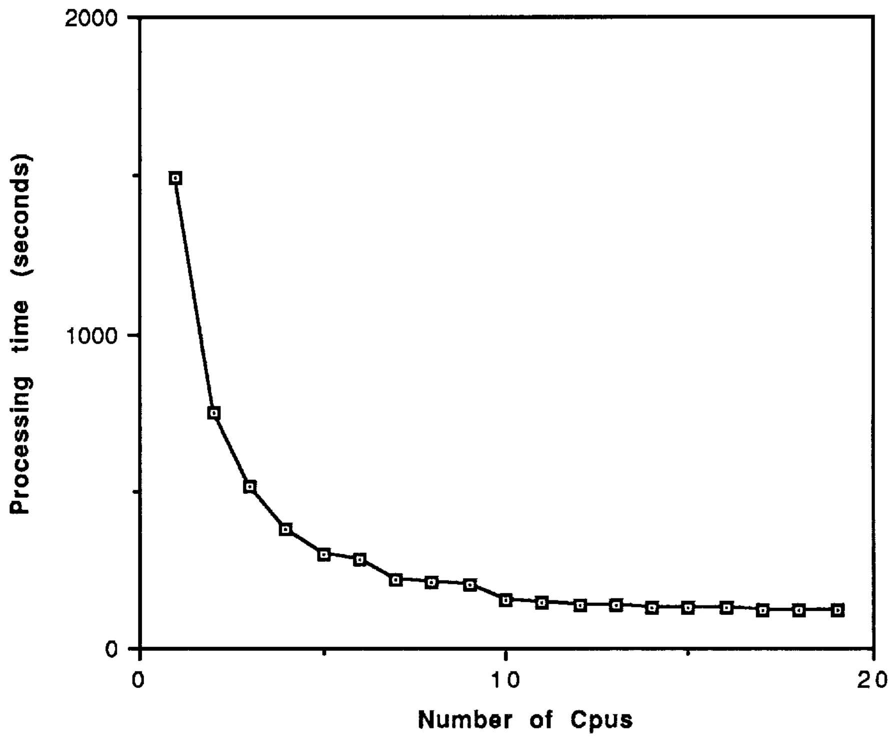


Figure 4 - Performance curve for parallel Autoclass/Ada on Symmetry

The parallel Autoclass / Ada program, in its current form, has allowed us to check all the Ada run time features for which we would have been required to develop specific programs (also part of the deliverable.) These facts facilitates the evaluation of parallel Ada environment. Each time the program has been ported to a new platform, weaknesses and strengths of the Ada environment were tested by the program. The program is thus used instead of the library of programs that would have been required for this purpose. There is enough configurable parameters with the current version of Autoclass / Ada to allow the testing of all the parameters that need to be evaluated in an Ada environment.

Following the successful processing of 1/10 of the IRAS database, the program was extended to its current version to allow the processing of the entire IRAS database. Runs of the program were executed both on the Sequent/Balance and the Sequent/Symmetry. The program has also been used to process a database of profiles characterizing cloud formations over a thermal front in the Atlantic ocean. These various experiments have demonstrated that the implementation of a concurrent version of Autoclass in Ada was indeed possible. The results have impressed the Artificial Intelligence group of NASA / Ames and have opened the door for the utilization of Ada on parallel architectures for supercomputing applications.

III - Conclusion

The design and implementation of a concurrent version of the Autoclass program in Ada has been completed successfully. The performances of the resulting program exceeds the initial project requirements. The program has been extended to allow the processing of large databases. Additionally, the program can be used as a benchmark and testbed for parallel Ada environments.

We wish to thank NASA / Ames for giving us the opportunity to make a contribution to the development of advanced computing capabilities for NASA's future missions.