

CAPTools Project:
Evaluation and Application
of the Computer Aided Parallelisation Tools

David O'Neal
The University of Illinois
Aeronautical Systems Center, Wright-Patterson Air Force Base, OH

Richard Luczak
Rice University
Aeronautical Systems Center, Wright-Patterson Air Force Base, OH

Michael White
Ohio Aerospace Institute
Air Force Research Laboratory, Wright-Patterson Air Force Base, OH

Abstract

The number of Fortran legacy codes maintained by the DoD research community is unknown, but it is generally accepted to be a significant one. Availability of a programming tool capable of accurately analyzing dependencies and generating portable parallel source codes would prompt the development teams maintaining legacy codes to test it on their own applications. Given sufficient time, the effect on utilization of HPC resources could be dramatic, as improvements in job turnaround would permit larger simulations to be considered.

The CAPTools utility developed by the University of Greenwich (UG) has proved itself capable of demonstrating the aforementioned qualities for a given set of examples (see <http://captools.gre.ac.uk/>). An evaluation of its ability to produce the same effect on a small set of research codes was initiated in October, 1998. UG and the Aeronautical Systems Center (ASC) Major Shared Resource Center (MSRC) would collaborate on the effort.

Following a brief project overview, the focus of the paper turns to key problems and the techniques applied in resolving them. Conclusions and a summary of relevant, on-going work are presented in closing.

Introduction

The origin of the PET CAPTools Project can be traced to a Government-sponsored visit from Mark Cross, Director of Research at the University of Greenwich, London, UK (January, 1998). Talks describing CAPTools and Jostle (a domain decomposition utility) were presented at the ASC MSRC. A collaborative evaluation of CAPTools was defined and proposed shortly thereafter. PET funding for technical support and travel was approved in October.

A related collaboration between UG and NASA Ames was incorporated into the ASC project scope in December of 1998. Computer scientists at Ames working on a legacy code workbench had produced a version of CAPTools that included an OpenMP (1997) source code generator. The UG and Ames packages were initially provided to ASC as separate installations, but in May, 1999, arrangements for routing DoD CAPTools releases through Ames were made and an integrated beta release emerged. Unless noted to the contrary, subsequent references to CAPTools correspond to this package.

Applications

A set of application codes was identified for testing various aspects of the CAPTools product. In most cases, special treatment was required to accommodate the input format of the CAPTools DoD-2.1Beta release.

- CAPTools currently supports standard Fortran 77 syntax (an F90 parser is in development). Statements that deviate from standard F77 will generally need to be rewritten. An unusual I/O format and a few instances of array syntax were encountered and (justifiably) rejected by CAPTools during testing. Required modifications were completely straightforward.
- Applications represented by multiple files must be gathered into a single source or itemized in a text file (.LIST suffix). The latter input option is currently undocumented. Relatively little effort is required to support either case. Include files are now supported by CAPTools, eliminating the explicit insertion exercise associated with earlier versions.
- Source files containing conditional compilation directives should be preprocessed before they are loaded into CAPTools. The parser treats compiler directives as comments and therefore all program statements are loaded. Unless an error occurs as a side-effect of the unexpected inclusions, no warning is provided to the user.
- All external references must be resolved. This typically means that stubs corresponding to library calls or platform-specific routines must be provided by the user. These same routines must then be commented back out of CAPTools generated source prior to compilation. Unresolved externals are listed during unsuccessful load attempts.

After successfully loading a code, the Analyser window opens automatically. Two distinct algorithms for determining dependencies are supported (Banerjee, Omega). Any combination of three dependency tests (Scalar, Exact, Disproofs) may be performed within any combination of three contexts (Interprocedural, Knowledge, Logic). A group of three preset selections have also been provided (Basic, Intermediate, Full). Ultimately, a full analysis is generally desired.

Upon completion of any type of analysis, OpenMP source can be generated immediately. Message passing codes must be partitioned and communication logic must be developed before writing a new source (CAPTools, 1998). A few caveats worthy of mention at this point include:

- CAPTools-generated message passing source code contains calls to CAPlib, a library of routines built upon various message passing primitives (MPI, PVM, shmem). This wrapper layer was implemented before MPI had emerged as the preferred standard message passing interface. A native MPI (1994) code generator has been requested.
- Automatic array allocation is supported, but the masking routines have not been extensively tested on such codes. We applied one workaround. Dynamically allocated arrays were pulled out to the main program level and declared statically, then passed back into the old declarator through a modified argument list.
- The dependency analysis kernel presumes static allocation of all variables (as if SAVE statements had been implemented everywhere). This assumption can result in detection of false dependencies across subroutine boundaries. Such dependencies should be removed before generating source code.

Validation of test codes was initiated at this point. A summary of results is provided in Table 1.

Application	Model Geometry	Subroutines and Functions	F77 Lines In	OMP Lines Out	MPI Lines Out	OMP Validation	MPI Validation
R-Jet	3D	63	7655	7471	19284		1,2,4
N-Body	3D	2	195	207	376	1-6	1-6
PFEM	2D	16	2073	1934	3371	1-8,16,32,64	1-8,16
FDL3DI	3D	76	9964	15330	11993		1-8,16
ARC	2D	28	2027	1881	2451	1-4	1,3
ADM	2D	78	12679				

Table 1. CAPTools test applications and status.

R-Jet (AFRL Air Vehicles Directorate) is a hybrid high-order compact finite-difference spectral method for simulating vortex dynamics and breakdown in turbulent jets. While the code is explicit in time, the compact finite difference scheme requires inversion of tridiagonal matrices, giving rise to the same sort of parallelization problems exhibited by implicit methods. The code uses complex variable arrays that are mapped into real variable arrays in the FFT algorithm and

block domain decomposition, both of which initially gave CAPTools some problems. Prior to this effort, this code had not been parallelized. Various parallel versions are currently being tested.

N-Body (Rice University) is a 3-dimensional n-body model with constraints. The bodies move across the surface of a unit sphere interacting with each other according to a $1/d^2$ force law (here d denotes distance between two bodies). The serial version was used as input to CAPTools. A hand-written MPI code was also made available for comparison of scalability and performance. The dependency analyses and partitioning phases completed normally without any additional information from the user. Therefore, the MPI and OpenMP source codes were generated in a fully automatic way. The corresponding executables run to completion and produce correct results when run on 1 to 6 processors of an Origin 2000.

PFEM (Parallel Finite Element Method, Carnegie Mellon University) is a research-level finite element code suitable for solving highly nonlinear boundary value problems in two dimensions. Nonconforming elements facilitate domain decompositions that effectively uncouple large numbers of element calculations. The availability of hand-written parallel versions complete with timers and error estimation made it an ideal case for validation and benchmarking purposes. After identifying and correcting a variable scoping error found in the OpenMP source, results for two nontrivial cases were validated on various partition sizes. Timings associated with these runs are presented in Tables 2 and 3. Experiments with the MPI source are still in progress.

PFEM 128x128	1 PE		2 PEs		4 PEs		8 PEs		16 PEs		32 PEs		64 PEs	
	wall	user	wall	user	wall	user	wall	user	wall	user	wall	user	wall	user
Serial	45.6	45.6												
Cray	47.1	47.1	27.6	55.2	17.6	70.8	11.4	91.4	8.4	136.0	9.8	317.9	20.9	1336.1
OMP	48.4	48.5	26.9	53.9	14.9	60.6	10.2	81.3	4.4	71.3	3.0	98.7	9.6	799.1
MPI	75.1		50.5		31.0		16.2		12.2					

Table 2. PFEM timings (sec) for a 128x128 case (50K DOF)

PFEM 256x256	1 PE		2 PEs		4 PEs		8 PEs		16 PEs		32 PEs		64 PEs	
	wall	user	wall	user	wall	user	wall	user	wall	user	wall	user	wall	user
Serial	982.0	982.0												
Cray	965.3	965.3	635.3	1264.7	431.0	1720.0	284.7	2233.4	173.0	2668.8	125.3	3794.1	145.3	8862.6
OMP	1000.4	1000.4	635.0	1269.8	341.0	1364.3	192.2	1534.0	118.9	1899.6	86.5	2766.2	93.6	5517.6
MPI														

Table 3. PFEM timings (sec) for a 256x256 case (200K DOF)

Note that all timings reported by Tables 2 and 3 were taken on a 195 MHz SGI Origin 2000. Executables associated with the first three rows of data were produced by compiling a single source code three different ways. The original source contained hand-written Cray Microtasking

directives. It was modified by CAPTools to include OpenMP directives as well. Executables were created by disabling all multiprocessing directives (no options specified), enabling only the Cray directives (-cray_mp), and enabling only the OpenMP directives (-mp).

FDL3DI (parallel version furnished by Wright State University) is a high-frequency Navier-Stokes model used to study aeroelastic effects. A parallel implementation based on the use of a chimera overset grid decomposition scheme was modified to run in a serial fashion. From this code, an MPI source based on a one-dimensional data distribution was recently completed and testing has been initiated. Collective use of the numerous variations of the FDL3DI code maintained by AFRL is heavy. Interest in developing a recipe for converting all of these sources to parallel form is what led to the inclusion of the FDL3DI code in this study.

ARC (Aluminum Reduction Cell, University of Pittsburgh) models electrothermal effects in commercial ovens used to reduce pure aluminum out of bauxite ore. Both steady state and time-dependent versions in two dimensions were available for testing. Corresponding three-dimensional codes are currently under development. Prior to this effort, none of the ARC codes had been parallelized. Analysis of the steady state code has nearly been completed. Work with the time-dependent code has just commenced.

ADM (Axisymmetric Damage Model, AFRL Materials Directorate) is a fracture mechanics code specifically designed to simulate crack propagation in composite materials. ADM is based on a meshless method that produces a solution for a system of ordinary differential equations. A hand-written parallel port based on sectional decomposition was implemented in 1998. A loop-level solution layered under the existing decomposition is also desired. Numerous GOTO statements and REAL*16 variables characterize the code. As of this writing, we have not yet been able to get the ADM source through the dependency analyser.

Summary

We are on the verge of completing twelve new parallel ports, half of which are directly linked to ongoing DoD research projects. We can also confirm that CAPTools is able to generate OpenMP and MPI source codes that compile, run, and produce correct results on an SGI Origin 2000 (three cases completed at this point). Material regarding performance and scalability will be presented at a later date.

Our collaboration over the past few months has been exceptional. The CAPTools interface has improved and the system has become much more robust, but work still remains (examples are cited in the Applications section). Although we regard CAPTools as a research-level application, we can recommend deployment in a limited sense (restricted access for pioneer users). Scheduling for a comprehensive CAPTools training event should be considered. Direct support from UG will be necessary to assure positive outcomes.

Acknowledgments

We wish to express our sincere thanks to Professor Mark Cross, Drs. Constantinos Ierotheou and Steven Johnson, and Peter Leggett of the Parallel Processing Research Group, University of Greenwich, London, England, UK, and Dr. Henry Jin of the NASA Ames Research Center, Moffet Field, CA, without whose help this project would not have been possible. We would also like to thank Drs. Miguel Visbal and Raymond Gordnier, AFRL Basic CFD Research, and Dr. Karen Tomko, Wright State University, for contributing the FDL3DI test code, and Drs. Nicholas Pagano, Gregory Schoeppner, and G. P. Tandon, AFRL Materials Directorate, for contributing the ADM test code.

References

1. Computer Aided Parallelization Tools User's Guide, Parallel Processing Research Group, University of Greenwich, London, UK, Version 2.0 Beta, October, 1998.
2. MPI: A Message Passing Interface Standard, University of Tennessee, Knoxville, TN, May 5, 1994.
3. OpenMP Fortran Application Program Interface, Version 1.0, October, 1997.