Running the SNL MPI Benchmark Suite on the Cray T3E

Mike Davis, System Support Analyst, Silicon Graphics, Inc., Albuquerque, NM

1 Overview

Over the past two years, Sandia National Laboratories (SNL) has been porting their major application codes to run on distributed-memory and heterogeneous-cluster platforms using the Message Passing Interface (MPI) paradigm. SNL has developed a benchmark suite from this collection of codes, and SGI's Cray Research Division is currently in the process of evaluating the performance of these benchmarks on the Cray T3E computer system. This paper describes the current status of this ongoing activity and some of the results obtained so far.

The paper is divided into five sections, including this overview. Section 2 describes each of the benchmarks briefly, and identifies the benchmark analyst working on it. Section 3 outlines the strategy that we used to port the benchmarks. In Section 4 I relate some of the problems we faced in the porting process and the solutions we developed to overcome them. Finally, in section 5, I present our results to date and outline our plans for future work.

2 The SNL MPI Benchmarks

The SNL MPI benchmark suite is composed of seven key SNL application codes. The serial versions of these codes have been used to perform crucial scientific simulations at Sandia and at other research institutions for many years. Most of them are written in Fortran 77 and have been designed to execute on a variety of platforms, with special attention paid to the shared-memory moderately-parallel vector machine.

2.1 ALEGRA [1]

ALEGRA is a three-dimensional solid dynamics code capable of performing integrated shock physics and structural analysis on a single problem. Designed from the outset to run on massively parallel computers and implemented almost entirely in C++, ALEGRA is the exception to the standard SNL MPI benchmark profile outlined in 2.0 above; but it also represents the future direction of computing and software development at SNL.

The benchmark analyst working on ALEGRA is Phil Campbell (Phil.Campbell@cray.com), a System Engineer based in Albuquerque, NM.

2.2 COYOTE2 [2]

COYOTE2 is a Finite Element Analysis code designed to simulate heat conduction and chemical reaction, for applications such as lubrication flow and saturated flow through porous media. It is written in Fortran 77 and makes heavy use of Level 3 Basic Linear Algebra Subroutines.

The benchmark analyst working on COYOTE2 is Mike Davis (Mike.Davis@cray.com), a System Support Analyst based in Albuquerque, NM.

2.3 JAS3D [3]

JAS3D is a three-dimensional finite element code, written in Fortran 77 and used for analyzing large deformations of materials subjected to high strain rates.

Hal Meyer (Hal.Meyer@cray.com), an Albuquerque-based System Engineer, is the benchmark analyst working on JAS3D.

2.4 MPCTH [4]

MPCTH is a hydrodynamics code designed to treat a wide range of shock wave propagation and material motion phenomena in one, two, or three spatial dimensions. Its predecessor, CTH, was highly optimized for the Cray Y-MP architecture. MPCTH is written in Fortran 77.

Phil Campbell is working on MPCTH.

2.5 PRONTO3D [3]

PRONTO3D is a solid dynamics code designed to analyze large deformations of materials subjected to high strain rates. It is implemented in Fortran 77.

Hal Meyer is working on PRONTO3D.

2.6 Quicksilver [5]

Quicksilver is a three-dimensional finite difference electromagnetic code implemented in Fortran 77 and designed to simulate the motion of charged particles using Particle-In-Cell (PIC) techniques. Quicksilver is also an exception to the standard SNL MPI benchmark profile, because it is not designed to run in parallel in a distributed-memory message-passing environment. Like CTH, Quicksilver was extensively tuned for the Cray Y-MP to vectorize, multitask, and perform SSD I/O. It is included in this benchmark suite specifically to test the ability of a distributed-memory computer system to support a shared-memory programming environment.

Rick Roloff (Richard.R.Roloff@cray.com) is the benchmark analyst working on Quicksilver. Rick is a System Engineer based in Denver, CO.

2.7 MPSALSA [6]

MPSALSA is designed to solve chemically reacting flow problems in three dimensions. Analysis of both flow and reaction kinetics is performed. Primary applications involve the simulation of chemical deposition processes of interest to the semiconductor industry. Like ALEGRA, MPSALSA was designed from the outset to run on massively parallel computer systems. MPSALSA is written in C.

The benchmark analyst for MPSALSA is Mike Long (Mike.Long@cray.com), a System Engineer based in Salt Lake City, UT.

2.8 Commonalities among the codes

Although the benchmark suite is composed of seven distinct application codes, there are many commonalities among them that can tend to simplify the porting and optimization process.

COYOTE2, JAS3D and PRONTO3D are all members of a single application system called ACCESS **[3]**. Codes in the ACCESS system are developed and maintained in accordance with a single set of guidelines and use a common set of libraries and tools. In the production environment, the integration and maintenance of many of the ACCESS codes are done by a single group.

In addition, ALEGRA and MPSALSA use parts of the ACCESS support environment to perform their binary datafile I/O in a platform independent manner. Lastly, both MPSALSA and COYOTE2 use a common parallel linear system solver package called AZTEC **[7]**.

3 Benchmark Strategy

The benchmark process was divided into five stages; each stage addresses a key issue in porting these codes to the T3E.

3.1 PVP / CF77 Verification

Although the codes that make up the benchmark suite have, in their serial configurations, been successfully executed on a wide range of platforms, the MPI versions have been exposed to only a few environments. Some, but not all, of the benchmark codes had been run on a Cray Parallel Vector Processor (PVP) system, such as the Cray J90, using the CF77 compiling system and the MPI component of the Cray Message Passing Toolkit (MPT). Our first porting step, then, was to verify that the codes would execute properly on a Cray PVP system.

3.2 CF90 Port

Since the CF77 Compiling System is not available on the T3E, we had to port the Fortran 77-based codes to the CF90 programming environment.

3.3 T3D Port

During the course of this benchmarking project, the availability of T3E systems for benchmarking purposes has been limited. T3D cycles were in comparatively good supply, however, so we deemed it reasonable to make a T3D port the next step in the process. Because the MPI component of MPT is not available on the T3D, we used the T3D MPI software from Edinburgh Parallel Computing Centre (EPCC) to build the codes **[8]**.

3.4 T3E Port

Once a benchmark code was executing successfully on the T3D, it would be moved to the T3E and run there. Each benchmark code in the suite comes with at least two input problem sizes, called “short” and “long.” The short problem is designed to run on 4 PE’s and the long problem 128. Until very recently, only two T3E systems have been generally available to us, one with 4 PE’s and one with 64. So our next logical step was to ensure that the “short” benchmark problems would execute.

3.5 T3E Scale-up

We are now in the process of locating a 128-PE T3E system on which to run the “long” benchmark problems. This step will yield the most interesting information.

4 Lessons Learned

Each step in the porting process presented its own unique set of problems. For most of them, the cause and the solution were fairly obvious; there were some, however, which have required significant time to track and solve.

4.1 PVP/CF77 Verification

This step in the porting process proved relatively painless, although two problems of interest did arise. First, a few of the benchmark codes referenced obsolete versions of routines that are now part of the Posix Fortran interface. (This is actually more pertinent to CrayLibs than CF77.) The CF77 compiling system recognizes these at load time and flags them as unsatisfied external references. We solved the problem by modifying the source code to make the correct Posix Fortran (PXF) calls. Note that it is generally NOT safe to simply equivalence these obsolete calls to their Posix counterparts, because the argument lists are quite different.

Second, the codes that executed IPXFARGC to get the number of arguments on the command line would find that number to be 2 greater than expected. The extra arguments were added to the command line by the **mpirun** script, which is used to start the MPI application. Since **mpirun** always adds exactly 2 arguments to the command line, and they are always the last two arguments, a reasonable fix is to deduct 2 from the return value of IPXFARGC. This is, in fact, the fix we used. It is worth noting, however, that this problem is not present in the T3E environment, due to differences in the implementations of the Network and MPP versions of MPI.

4.2 CF90 Port

Problems encountered in the CF90 port all fell into the category of data type conflicts.. We found instances where variables of non-default size (e.g. INTEGER*4, LOGICAL*4) were being passed to Fortran intrinsics and I/O statements. The CF90 compiler flags these occurrences and issues a diagnostic message. We solved the problem by modifying the source code so that the passed variables were of default size.

4.3 T3D Port

It was in porting the benchmark codes to the T3D that we encountered the most interesting problems. And because they usually exhibited no direct symptoms they were also the most difficult to track down and solve.

First, there was the problem of differing data word size. See Table 1 below. All of the SNL MPI benchmark codes had been executed successfully on a platform whose characteristics match those described in the row labelled "Workstation." In the case of those codes that had been run on a PVP machine, the source code had been modified to include the appropriate conditional code to handle PVP characteristics. But, since the codes had no provision for the T3D, sizes were being computed incorrectly, thus causing indexing and data corruption problems. We fixed the problem by closely examining all conditional sections of code (typically denoted by "#ifdef CRAY" constructs) and changing all references to "sizeof(float)" into "sizeof(double)."

Another problem arose because of differences in the behavior of the Fortran SIGN intrinsic on the T3D compared to the PVP. The SIGN intrinsic accepts two arguments, say X and Y, and returns -ABS(X) if Y is less than zero or ABS(X) if Y is greater than or equal to zero. The difference occurs in the case where Y has a value of -0. On the PVP, SIGN returns ABS(X), whereas on the T3D it returns -ABS(X). This problem exhibited itself in JAS3D and PRONTO3D, where the following construct was heavily used:

```
    T = SIGN (0.5,Y) + SIGN (0.5,-Y)
```

which is another way of saying:

```
      IF (Y.EQ.0) THEN
         T = 1
      ELSE
         T = 0
      ENDIF
```

On the T3D, this constuct was always yielding a value of zero for T. We solved the problem by replacing all occurences of the construct with the equivalent IF block.

A problem that appeared in the ACCESS dynamic memory management library turned out to be due to the difference in the return value of the **sbrk** system call on the T3D versus the PVP. On the T3D, **sbrk** returns a byte address, whereas on the PVP it returns a word address. The ACCESS memory manager was designed to return a word address to its caller, and contained conditional code to decide whether or not to scale the **sbrk** address after allocation. This conditional code was treating the PVP and the T3D alike, which caused improper addressing and data corruption problems on the T3D. We fixed the problem by modifying the source code of the ACCESS memory manager to interpret **sbrk** addresses as byte addresses on the T3D.

The internal representation of the Fortran character descriptor is drastically different on the T3D compared to the PVP. Specifically, it is a two-word structure with the address in the first word and the length in the second. This caused problems in sevaral Fortran-callable C-language utility routines, where it was assumed that Fortran character arrays passed in as arguments could be treated as C character arrays or as non-character arrays. This assumption was safe on the PVP system so long as the Fortran character array being passed was aligned on a word boundary. On the T3D, however, translation is always necessary. The recommended method of translating Fortran character descriptors in a C function is via the **_fcd** family of functions defined in the header file **fortran.h**. We fixed these problems by identifying them individually as they presented themselves, and modifying the appropriate source code modules to do the **_fcd** translation.

In a related matter, the EPCC MPI library for the T3D does not support the use of any Fortran-based message-passing calls with a value of MPI_CHARACTER for the data type argument. The EPCC documentation **[8]** describes this restriction, justifying it as necessary to preserve the performance of the message-passing routines; it also includes a technique for circumventing the limitation, involving the use of non-character arrays and the EQUIVALENCE statement. We chose to implement a less intrusive workaround, using a small C-language interface library. Note that this restriction also exists within the MPI component of the Cray Message Passing Toolkit.

The only "easy" problem that occured during the T3D port involved missing sort and search routines. The CrayLibs environment on the PVP platform includes some popular routines to sort numeric data (ORDERS) and perform searches on sorted

Table 1. Sizes of Numeric Data Types on Various Platforms

Platform	**Real**	**Double Prec.**	**float**	**double**
Workstation	4	8	4	8
Cray PVP	8	16	8	8
Cray T3D	8	N/A	4	8

data (WHENEQ et al., ISRCHEQ et al.). These routines are not available in CrayLibs on the T3D. The JAS3D and PRONTO3D codes require these routines to perform their material-contact analysis. We fixed this problem by locating source code for the sort and search routines in the Cray Unicos source archive and integrating it into the benchmark codes.

4.4 T3E Port

Once ported to the T3D, the benchmark codes had no problems moving to the T3E. All we had to do was recompile the codes with targeting set for the T3E and delete all references to the EPCC MPI software (the MPI component of the Cray Message Passing Toolkit was available for us on the T3E). From our experience so far, it appears that the T3E is fully compatible with the T3D in the areas of functionality and software environment.

5 Future Work

As of the date of this writing, we are in the process of running the benchmark codes on the T3E under the Apprentice performance analysis tool to identify areas to optimize, and we are trying to locate a 128-PE system on which to run the "long" benchmark problems. We expect to be able to report on the results of these activities at the next CUG. Further, we will be testing the Quicksilver benchmark on HPF/CRAFT when it becomes available. In the meantime, we hope that our experiences in porting the SNL MPI benchmarks to the Cray T3E system will prove to be of use to other T3E users.

6 References

[1] http://www.sandia.gov/1431/ALEGRAw.html

[2] http://www.cfd.sandia.gov/docs/coyote/coyote-welcome.html

[3] http://www.cs.sandia.gov/SEACAS/SEACAS_Overview.html

[4] http://www.sandia.gov/1431/CTHwdoc

[5] http://www.ppt.sandia.gov/projects/quicksilver

[6] http://www.cs.sandia.gov/CRF/mpsalsa.html

[7] http://www.cs.sandia.gov/HPCCIT/aztec.html

[8] http://www.epcc.ed.ac.uk/t3dmpi/Product