The Cray XT Programming Environment

Luiz DeRose, Mark Pagel, Heidi Poxon, and Adrian Tate
Cray, Inc.

ABSTRACT: *In this paper we presented the programming environment for the Cray XT system, which consists of state of the art compiler, tools, and libraries, supporting a wide range of programming models. The Cray XT programming environment has been designed to address the complexity of large scale HPC systems, to help users achieve the highest possible performance from the hardware. Its design focus is on providing an ease of use and optimised performance on the Cray XT system.*

KEYWORDS: Programming environment, programming models, compilers, math software, tools.

1. Introduction

It is the role of the Programming Environment to close the gap between observed performance and peak performance, in order to help users achieve highest possible performance from the hardware. The Programming Environment should address issues of scale and complexity of high end HPC systems and hide the complexity of the system from the user, to help them to be more productive.

In this paper we describe the Programming Environment for the Cray XT systems, which was designed with the focus on performance and ease of use. The paper is organized as follows: Section 2 addresses programming languages and compilers. Section 3 discusses programming models. Math software is presented in Section 4, and tools in Section 5. Finally, we present our conclusions in Section 6.

2. Compilers & Parallel Programming Models

The Cray XT programming environments relies on state of the art compilers provided by independent software vendors such as PGI and PathScale. The GNU compilers are also available on the systems. The main languages used in scientific and technical computing: Fortran, C, C++, and Java are the supported languages.

The programming environment of the Cray XT system supports an extensive set of programming models, including distributed memory approaches like MPI and SHMEM, shared memory mechanisms like OpenMP, and the Unified Parallel C (UPC) PGAS language. In this section we describe the main characteristics of each of these models.

2.1 MPI

The Cray implementation of the Message-Passing Interface (MPI), which supports parallel programming across a network of computer systems, is based on MPICH-2 and implements the MPI-2 standard, except for spawn support.
It also implements the MPI 1.2 standard, as documented by the MPI Forum in the spring 1997 release of MPI: A Message Passing Interface Standard.

The Cray MPI implementation on XT calls the Portals layer of software that has been specifically adapted for the XT architecture. Many enhancements have been made over the past year to improve performance, especially for dual-core and quad-core Opterons. Some of those enhancements have been made in portals and MPI and are enabled by default. Others are MPI specific and require enabling environment variables to take advantage of those specific optimizations. The "intro_mpi" man page contains descriptions of these environment variables.

2.2 Cray SHMEM

The Cray SHMEM data-passing library routines are similar to the message passing interface (MPI) library routines: they pass data between cooperating parallel processes. The Cray SHMEM data-passing routines can be used in programs that perform computations in separate address spaces and that explicitly pass data to and from different processing elements (PEs) in the program.

The Cray SHMEM parallel programming model assumes an MPI-1 like group of processes that runs in parallel from job launch to job termination. No processes can be added or removed from this group and all processes execute the same application. Thus, Cray SHMEM applications are of the SPMD (Single Program Multiple Data) type. However, a Cray SHMEM application can be part of a larger MPMD (Multiple Program Multiple Data) type MPI job. Cray SHMEM is a one-sided message passing model in which memory is private to each process.

The Cray SHMEM routines minimize the overhead associated with data passing requests, maximize bandwidth, and minimize data latency. Data latency is the length of time between a PE initiating a transfer of data and a PE being able to use the data.

Cray SHMEM routines support remote data transfer through "put" operations that transfer data to a different PE and "get" operations that transfer data from a different PE. Other supported operations are work-shared broadcast and reduction, barrier synchronization, and atomic memory operations. In addition, non-blocking "put" operations are also supported. The "intro_shmem" man page gives descriptions of the supported interface calls as well as additional information on how to use the Cray SHMEM library.

2.3 OpenMP

OpenMP provides multithreaded, shared-memory parallelism using directives to specify data locality, work distribution, and control flow. The API is available for both C/C++ and FORTRAN. OpenMP is supported on the Cray XT system running the Cray Linux Environment on the compute nodes and is provided by the compiler vendors PGI and PathScale.

2.4 UPC

Unified Parallel C (UPC) is C extension to allow programmers to specify both data distribution and work distribution in a single program multiple data (SPMD) programming model. The Cray XT-UPC is a combination of the the Berkeley UPC compiler, a source to source translator and the Intrepid GCCUPC, a UPC to assembly compiler based on GNU gcc. The four main components of the Cray XT UPC Package consist of:

1. GASNet, the Global-Address Space Networking Version 1.8.0, which was ported by UC Berkeley/LBNL, to run on top of the Cray XT Portals implementation.
2. UPCR, the UPC Runtime library Version 2.5.10 from UC Berkeley/LBNL
3. The Berkeley UPC Translator (BUPC) Version 2.4.0, also ported by UC Berkeley/LBNL
4. GCCUPC Version 4.0.3.4, a UPC to assembly compiler based on gnu gcc, developed by Intrepid.

Both BUPC and GCCUPC generate code which is linked with UPCR and GASNet. Both compilers are compliant with UPC specs Version 1.2. They also include the Berkeley reference implementation of UPC-IO and an optional portion of the UPC V1.2 specs. The C code generated by BUPC can be compiled with the PGI compiler (pgcc) or the GNU compiler (gcc).

3. Math Software

Cray XT Series Math Software consists of a combination of Cray custom tuned algorithms and third Party products, often with Cray added value or specifically tuned for the Opteron processor. Base linear algebra operations are performed using AMD's Core Math Library (ACML) which contains BLAS and LAPACK routines tuned specifically for the AMD64 instruction set. ACML also contains a library of fast vector intrinsic for the efficient calculation of sets of transcendental functions.

Cray XT-LibSci contains base linear algebra functionality provided with the Goto Library, parallel linear algebra functionality through Cray-tuned versions of ScaLAPACK and PBLAS, and sparse direct solvers through the SuperLU (for non-symmmetric matrices). For users of traditional Cray FFT programs, XT-LibSci contains also a suite of FFTs programs that preserve the popular Cray FFT interface. However, these programs are provided for compatibility reasons only – Cray's recommended FFT package on XT systems is FFTW.

FFTW is provided by MIT and is the industry standard FFT library for scalar systems. FFTW automatically generates code that is tuned for the host machine, but Cray also provide information to the FFTW developer to help the library work well for Cray interconnect and MPI implementations. Currently, the newest and most efficient version of FFTW is FFTW3.1 does not contain parallel algorithms and as such Cray provide a second version FFTW2.1.5. These 2 separate versions are accessed via different modules.

The Cray Iterative Refinement Toolkit (IRT) is packaged as part of the XT-LibSci product and is a library of faster linear solvers. This library uses mixed precision iterative refinement, with a potential speed-up of 2x (in practice more likely 1.2 to 1.7) depending on the condition of the user's systems. IRT contains both parallel and serial versions of the common linear solvers, and includes a

very sophisticated convergence scheme that allows either forward or backward error to be minimized. See the intro_irt manpage for more details.

Libsci.10.2.0 also includes a highly tuned custom version of the popular iterative solver library PETSc. This release includes optimizations for certain matrix classes..

4. Tools

4.1 Debuggers

In addition to gdb, two third party debuggers are available on the Cray XT system. The TotalView debugger from TotalView Technologies is a powerful, sophisticated, and programmable tool that lets users debug, analyze, and tune the performance of complex serial, multiprocessor, and multithreaded programs. TotalView is available with its command line interface (CLI) and its graphical interface, supporting threads, MPI, SHMEM, OpenMP, C/C++ and FORTRAN, as well as mixed-language codes.

The Distributed Debugging Tool (DDT) From Allinea is an intuitive, scalable, graphical debugger. DDT can be used as a single-process or a multi-process program (MPI) debugger. Both modes of DDT are capable of debugging multiple threads, including OpenMP codes. DDT provides all the standard debugging features (stack trace, breakpoints, watches, view variables, threads etc.) for every thread and every process running as part of the application. It also supports the standard HPC languages.

4.2 Performance Tools

The Cray Performance tools provide an integrated infrastructure for measurement and analysis of computation, communication, I/O, and memory utilization, which is unique in the industry and represents the state of the art in performance measurement. This framework for performance analysis consists of the CrayPat Performance Collector and the Cray Apprentice^2 Performance Analyzer. CrayPat provides an infrastructure for automatic program instrumentation at the binary level with function granularity, creating an intuitive and easy to use interface for performance tuning of scientific applications on all Cray platforms. Users can select the functions to be instrumented by groups, such as MPI, I/O, memory, and user functions, or by name. CrayPat also provides an API for fine grain instrumentation. When instrumenting at a function level, users do not need to modify the source code, the makefile, or even recompile the program. CrayPat uses binary rewrite techniques at the object level to create an instrumented application, which is generated with a single static re-link, managed by the CrayPat infrastructure. A second main component of the CrayPat Performance Collector is its runtime performance data collection library, which can be activated by sampling or by interval timers. Performance data can be generated in the form of a profile or a trace file, and its selection is based on an environment variable. A third CrayPat main component is its report generator, which is a utility that reads the performance file that was created by the runtime library and generates text reports, presented in the form of tables. Special reports are produced, depending on the groups that were selected. For example, the collection of I/O allows reporting on bytes transferred, I/O wait time, files used, and so on. The collection of MPI allows the generation of detailed MPI profiles.

CrayPat supports programs written in Fortran, C, and C++, with the MPI, SHMEM, and OpenMP, programming models (or combinations of these programming models). It uses the PAPI library for collection of hardware performance counters and allows the user to select PAPI hardware counter presets, as well as native Opteron events. In order to facilitate the user selection of hardware counters events, CrayPat also provides set of predefined hardware counters groups. Derived hardware metrics, such as cache hit rate and MFlop rate are computed depending on the selected events or groups.

The Cray Apprentice^2 Performance Analyzer is a multi-platform, multifunction performance data visualization tool that takes as input the performance file generated by CrayPat and provides the familiar notebook-style tabbed user interface, displaying a variety of different data panels, depending on the type of performance experiment that was conducted with CrayPat and the data that was collected. Cray Apprentice^2 provides call-graph based profile information and timeline based trace visualization, supporting the traditional parallel processing and communication mechanisms, such as mpi, OpenMP, and SHMEM, as well as performance visualization for I/O. It can help developers to identify and correct load imbalance, excessive serialization and excessive communication problems.

Cray Apprentice^2 can be run either on the Cray XT service nodes, or on a remote Linux server or workstation. This allows a remote user who does not connect to the Cray XT through a high-performance network to still benefit from the power of this GUI, without having to experience long delays due to X Window traffic on the network.

5. Conclusions

In this paper we presented the programming environment for the Cray XT MPP system, which consists of state of the art compiler, tools, and libraries, supporting a wide range of programming models. The Cray XT programming environment has been designed to help users achieve the highest possible performance from the hardware. Its design focus is on providing an ease of use and optimised performance on the Cray XT system.

About the Authors

Dr. Luiz DeRose is a Sr. Principal Engineer and the Programming Environments Director at Cray Inc. He has more than twenty years of experience in HPC software design and development. He has published more than 40 peer-review articles in scientific publications, primarily on programming environment topics. He can be reached at ldr@cray,.com.

Mark Pagel is the manager of the MPT group at Cray Inc. He has more than 15 years of experience on software support for high performance computing. He can be reached at pags@cray.com

Heidi Poxon is the technical lead of the Performance Tools and Simulator groups at Cray Inc. She has more than 15 years of experience in software support for high performance computing. She can be reached at heidi@cray.com.

Adrian Tate is the technical lead of the Math Software group at Cray Inc. He has more than 10 years of experience in development of applications and mathematical software for high performance computing. He can be reached at adrian@cray.com