TotalView Happenings on Cray Platforms

(An Update by Cray and Etnus)

Luiz DeRose, Robert Clark, Robert Moench
Cray Inc.
Rich Collier and John DelSignore Jr.
Etnus

ABSTRACT: *TotalView is uniquely well-suited to debug applications that are developed on Cray supercomputers, which are used for high-end modelling and simulation. These applications are very large and complex with long run times. This paper will provide an update on current available functionality for Etnus's TotalView Debugger on The Cray X1, Cray XT3, and Cray XD1.*

KEYWORDS: Programming Environment Tools, Debuggers, Cray X1 Series, Cray XT3

1. Introduction

The purpose of this paper is to highlight the presence of the Etnus TotalView debugger on Cray supercomputer systems. This includes the Cray X1 Series and the Cray XT3, primarily. While discussing TotalView on the Cray XD1 is pertinent as well, those details do not deviate substantially from any standard Linux clustered system. There are simply fewer unique issues to comment upon.

2. Etnus TotalView Overview

The Etnus TotalView debugger is a powerful, sophisticated, and programmable tool that lets you debug, analyse, and tune the performance of complex serial, multiprocessor, and multithreaded programs.

TotalView supports threads, MPI, OpenMP, C/C++ and Fortran, plus mixed-language codes. Unique features like dive, a wide variety of breakpoints, powerful data analysis, and both Graphical User Interface and Command Line Interface options make TotalView the leader in its field.

3. Cray X1 Series

Cray X1 Series Overview

Cray X1 Series systems utilize powerful vector processors, shared memory, and a modernized vector instruction set in a highly scalable configuration to provide the computational power required for advanced scientific and engineering applications. Cray X1 Series systems have high memory bandwidth and scalable system software, which are crucial to achieving peak and sustained performance.

A Cray X1 Series system combines the single-processor performance and single-shared address space of Cray parallel vector processor (PVP) systems with the high bandwidth, low latency, scalable interconnect, and scalable microprocessor-based architecture used in Cray T3E systems.

UNICOS/mp, the Cray X1 Series' operating system, provides a single system image. One result of this is that the user (and debugger) has a more seamless access to, and presentation of, the resources of the system.

Overview of Node configuration

Cray X1 Series systems are powered by multistreaming processors (MSPs). Each MSP has four internal single-streaming processors (SSPs). Each SSP contains both a superscalar processing unit and a two-pipe vector processing unit.

The logical grouping of four MSPs and cache-coherent shared local memory is called a node. Cache coherency is maintained for the four MSPs in a node.

Physically, all nodes are the same; software controls how a node is used. Processors are designed so that an application can run in either MSP mode or SSP mode. In MSP mode, an MSP provides the user-programmable processor for typical parallel applications; each MSP tightly couples the interactions of its four constituent SSPs and automatically distributes the parallel parts of a multistreaming application to its SSPs. In SSP mode, each SSP runs independently of the others, executing its own stream of instructions. Applications can be built to run with one or more MSPs or with one or more SSPs, where the optimal choice depends on the algorithms used within the application.

Execution modes

Programs can be compiled in any of three modes: Command, SSP, or MSP. The default is to compile in MSP mode, allowing the compiler to locate parallelism in the code and automatically apply the 4 SSPs of an MSP to the problem. SSP mode allows the programmer to use other, more hands-on programming models to parallelize the code. Command mode is for less parallel intensive programs (**`ls`**, **`grep`**, etc). TotalView is compiled in command mode.

Systems are configured to have some nodes set up as support nodes and others set up as application nodes. By default command mode executables execute on support nodes, while SSP and MSP mode executables execute on application nodes.

All applications (SSP and MSP mode) are placed and launched via the **`aprun`** utility. This will happen by default, but can be requested overtly when the default **`aprun`** parameters are unsatisfactory.

Launching of distributed memory jobs

On the Cray X1 Series system, TotalView automatically launches SSP and MSP mode applications using **`aprun`**, while launching command mode executables without **`aprun`**. If the default **`aprun`** values are unacceptable, they can be controlled with the **`-app`** switch on the TotalView call line. The call:

```
totalview -app "-np 4" a.out
```

will invoke TotalView on the application **`a.out`**, use **`aprun`** to gang schedule and start four processes, run them to the first user code (e.g. routine **`main`**), and stop them, as shown in Figure 1. The processes are organized in a control group, such that flow control commands (e.g. **`step`**, **`go`**, etc.) will be applied to the entire group. The user can set debug breakpoints and continue a controlled debug session.

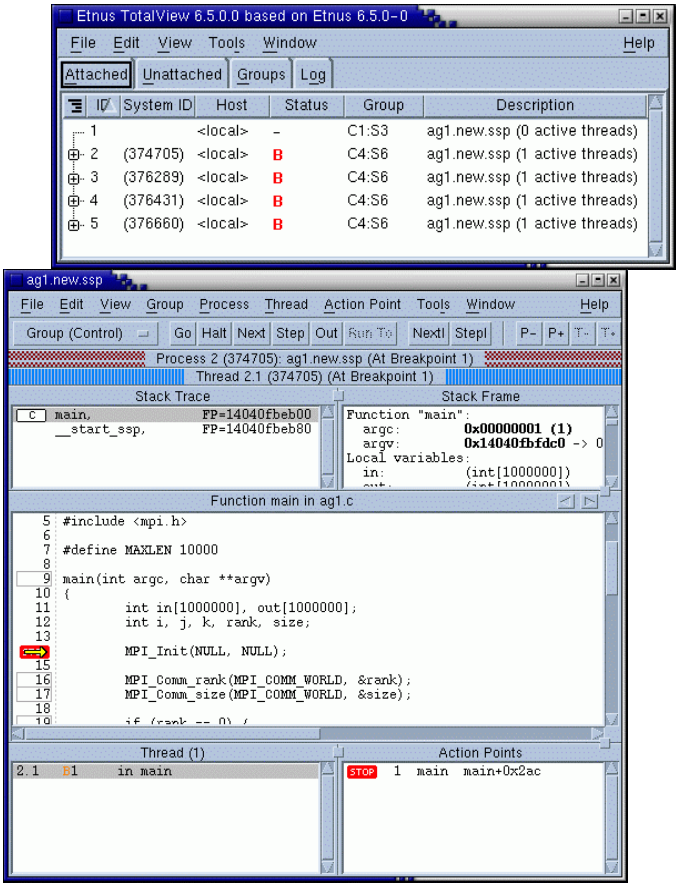

Figure 1: TotalView root window and process window from a 4 processors run.

A useful variant of the above scenario is to use TotalView to attach to a running application. In such a case, TotalView is invoked without parameters and the *Unattached* tab can be used to display all tasks belonging to the user and to which TotalView is currently not attached. By double clicking on any process in the intended application, TotalView will attach to all of the processes of that application. Having attached to them, TotalView will present a *Process Window* with the entire set of processes combine as a control group, as shown in Figure 2.

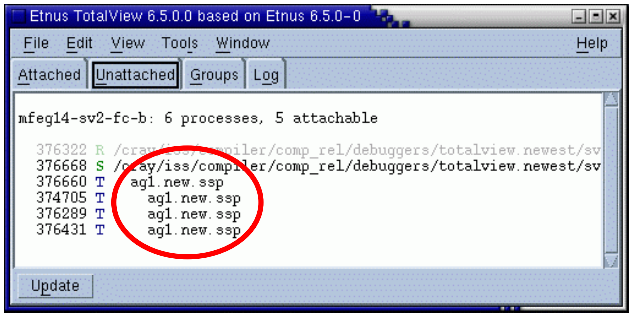

Figure 2: Attaching to a running application

It should also be noted that TotalView can be used to debug commands (rather than applications). This is done automatically whenever the executable supplied to TotalView is a command mode executable.

Access to SSP specific information of an MSP

The standard `-g` debugging compilation disables the default streaming of MSPs. That is, only SSP0 performs useful work, while SSPs 1-3 are parked in an idle loop. SSPs 1-3 do get put to work should a streamed region get executed (library routines or other routines not compiled with `-g`).

With the above in mind, TotalView defaults to showing SSP0 when debugging an MSP executable. However, it is still desirable to be able to access SSPs 1-3 in certain situations. The most common is when one of those SSPs takes a synchronous exception. In such an event, TotalView displays an indication of the offending SSP on the "Thread" status line of the *Process Window*, as shown in Figure 3.

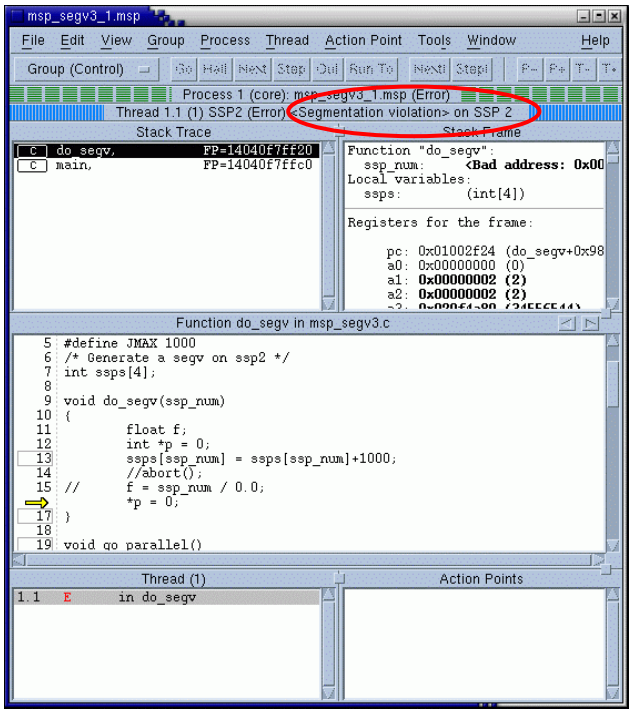

Figure 3: Indication of the offending SSP on the "Thread" status line of the Process Window.

To further investigate an exception, it is possible to direct TotalView to switch from displaying information only about SSP0 and instead display information only about an alternative SSP. On changing the SSP focus, all TotalView windows of all processes will update with data from the requested SSP. This feature can only be used for display or modification of data. Any attempt at flow control (stepping, etc) will cause the focus to return to SSP0. This feature is controlled with the SSP focus widget, shown in Figure 4.

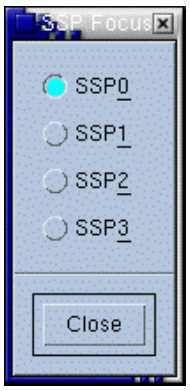

Figure 4: SSP focus widget.

Examining core files

On the Cray X1 Series system, a distributed memory job can generate a core file for each process in the job. TotalView provides an interface that makes accessing a set of core files quite straightforward.

The naming convention for core files is:

```
core.apid.pid
```

where **apid** is the id for the application team and **pid** is the process id of the individual process.

An example of bringing up TotalView on a core file set is:

```
totalview -e "coreset a.out core.2972"
```

This will invoke TotalView, which will come up with a single *Process Window* "attached" to all of the core files in the set, core.2972.* as processes of a common control group.

Vector registers

TotalView has been extended to display the vector and BMM registers of the Cray X1 Series processors. They are displayed with the other more standard registers, but as *divable* 64-element arrays. As with other registers, variables, and data, they can be cast to all of the data types of machine.

Programming models

TotalView supports a wide range of parallel programming models. In the distributed memory arena, it supports MPI, SHMEM, and to lesser extent, CAF and UPC. For shared memory models it supports Pthreads and OpenMP.

4. TotalView on the Cray XT3

Cray XT3 Overview

Cray XT3 supercomputer systems are powerful, massively parallel processing (MPP) systems. Cray has combined commodity and open source components with custom designed components to create a system that can operate efficiently at immense scale.

Cray XT3 systems are based on the Red Storm technology that was developed jointly by Cray Inc. and the U.S. Department of Energy's Sandia National Laboratories.

Cray XT3 systems are designed to run applications that require large-scale processing, high network bandwidth, and complex communications. Typical applications are those that create detailed simulations in both time and space, with complex geometries that involve many different material components. These long running, resource-intensive applications require a system that is programmable, scalable, reliable, and manageable.

Cray XT3 Node configuration

The basic scalable component is the node. There are two types of nodes. Compute nodes run user applications. Service nodes provide support functions, such as managing the user's environment, handling I/O, and booting the system. Each compute node and service node is a logical grouping of a processor, memory, and a data routing resource.

Users log into service nodes and invoke commands and user applications from them. User applications are then propagated to compute nodes where they run using the Message Passing Interface (MPI) and SHMEM parallel programming, distributed memory models.

The applications launcher, yod

The invocation of a user application is accomplished with the aid of a command called **yod**. **yod** executes on the service node on behalf of the application, arranges for the allocation of compute nodes and the launching of the application on those nodes.

TotalView and yod

To debug an application on the Cray XT3, TotalView and **yod** are used in conjunction (much as TotalView and **mpirun** might be used on other systems). In the following call:

```
totalview yod -a -sz=4 a.out
```

TotalView is invoked on the service node with **yod** and the user's executable (**a.out**) as parameters. When the user tells TotalView to start debugging (via **go**), TotalView will launch **yod**. **yod** will facilitate the launch of the user's executable and coordinate TotalView's attach to all of the launched processes on the compute nodes. TotalView gives the user an opportunity to stop the processes, opening a *Process Window* on the group of

processes in the application. The processes are organized in a control group, such that flow control commands (e.g. **`step`**, **`go`**, etc.) will be applied to the entire group. The user can set debug breakpoints and continue a controlled debug session.

A useful variant of the above scenario is to use TotalView to attach to a running application. In such a case TotalView is invoked without parameters and the *Unattached* tab can be used to display all tasks belonging to the user and to which TotalView is currently not attached.

By double clicking on the **`yod`** process, TotalView will attach to **`yod`** and all of the processes of the application associated with that **`yod`** process. Having attached to **`yod`**, TotalView will present a *Process Window* for the **`yod`** process.

Of greater interest are the application processes. By viewing the *Attached* tab, the application processes can be seen attached as a group. Double clicking on the group will present a *Process Window* with the entire set of processes combined as a control group.

It should also be noted that TotalView can be used to debug commands (rather than applications) running on a service node. This is done without the use of **`yod`** or compute nodes.

Scalability

TotalView controls the user's application processes with server processes running on service nodes. Each server controls up to 64 processes. Much as login nodes are load balanced by distributing users evenly over the system's service nodes, so too are the TotalView server processes. In this manner, the debugging of large numbers of processes is distributed across the resources to the entire system.

Security

The creation of TotalView server processes is accomplished through the use of secure shell clients (ssh). Each server is dynamically launched, as needed, with ssh. This provides secure encrypted communications between TotalView and its servers running on one or more of the service nodes. This does mean, of course, that users need to have proper machine authorization and keys set up to use TotalView.

Bibliography

Cray XT3TM System Overview
Cray X1TM Series System Overview
http://www.etnus.com/

About the Authors

Dr. Luiz DeRose is a Sr. Technical Engineer and the Programming Environment Tools Manager at Cray Inc. He has twenty years of experience designing and developing tools for HPC. He can be reached at 1340 Mendota Heights Rd, Mendota Heights, MN 55120 USA, E-mail: ldr@cray,.com

Bob Moench is a Software Engineer at Cray Inc. He has worked on debuggers for the last several years. He can be reached at 1340 Mendota Heights Rd, Mendota Heights, MN 55120 USA, E-mail: rwm@cray.com

Bob Clark is a Software Engineer at Cray Inc. He has worked on debuggers for the last several years. He can be reached at 1340 Mendota Heights Rd, Mendota Heights, MN 55120 USA, E-mail: clark@cray.com

Rich Collier is the VP of Engineering at Etnus. He can be reached at 24 Prime Parkway, Natick, MA 01760 USA, E-mail: rich.collier@etnus.com

John DelSignore is the Chief Architect at Etnus. He has worked on debuggers for twenty years. He can be reached at 24 Prime Parkway, Natick, MA 01760 USA, E-mail: john.delsignore@etnus.com