Debugging scalable hybrid and accelerated applications on the Cray® XC30™ and CS300™ with TotalView®

Chris Gottbrath
Rogue Wave Software
Boulder, CO
Chris.Gottbrath@roguewave.com

Abstract — **TotalView provides users with a powerful way to analyze and understand their codes, and is a key tool in developing, tuning, scaling, and troubleshooting HPC applications on the Cray XC30 Supercomputer and the CS300 Cluster Supercomputer Series. As a source code debugger, TotalView provides users with complete control over program execution and a view into their program at the source code and variable level. TotalView works with most any HPC application written either in C/C++ or Fortran, regardless of how it achieves parallelism (MPI, MPI+OpenMP, MPI+CUDA™, and both native and symmetric mode debugging with the current generation of Intel® Xeon Phi™ coprocessor) and gives users a complete view of the program to be debugged through a single graphical debugger interface. TotalView uses a scalable tree-based architecture and can scale up to hundreds of thousands of processes. This talk will introduce new users to TotalView's capabilities and give experienced users an update on recent developments including the new MRnet communication tree. The talk will also highlight memory debugging with MemoryScape (which is now available for the Xeon Phi), deterministic reverse debugging with ReplayEngine, and scripting with TVScript.**

Keywords—Debugging, Xeon Phi, CUDA, Hybrid

I. INTRODUCTION

This paper accompanies the talk given at CUG 2014, and provides a status update on the TotalView debugger for the Cray XC30 supercomputer and CS300 cluster supercomputer architectures. It expands on material presented at CUG 2013.

The particular focus of this paper is on two topics. First, recent advances in TotalView which are relevant to Cray users: updated support for CUDA, expanded support for the Xeon Phi, memory debugging on the Xeon Phi, and improved ReplayEngine support for the Cray. Second, the progress being made to achieve high performance and responsiveness for interactive debugging at scale on the Cray.

II. DEBUGGING ON THE CRAY XC AND CS300 WITH TOTALVIEW

TotalView provides a powerful and intuitive graphical source code debugging environment for a variety of different supercomputing architectures including the Cray XT™, XE™, and XC™. TotalView gives users control over and visibility into program execution. [1]

TotalView provides users control over the program through a single debugger interface. Process and thread control features allow users to easily synchronize all the threads and to exert nuanced control over large parallel jobs. The debugger also provides exceptional capabilities for controlling thread execution. Breakpoints can be set with thread width so users can more easily work with thread parallelism constructs, such as OpenMP parallel for loops.

TotalView features the ability to attach to an arbitrary subset of a parallel job and change that subset on the fly. TotalView gracefully handles multiple program, multiple data (MPMD) parallel jobs – with automatically generated groups that span the entire job, and other groups that operate only on the subsets that share executable images.

Variables and complex data structures can be examined and navigated with an intuitive variable display, data visualization, and exploration capability. This display capability makes type casting, working with pointers, and nested aggregate data types extremely easy and straightforward.

Since many scientific codes feature very important array-type data, TotalView provides a powerful array display. Arrays can be sliced and displayed using arbitrary striding which can be specified with Fortran slice notation (even in C). Array data can be displayed in three ways:

1. Memory-ordered elements in list form
2. 2D slices displayed in spreadsheet format
3. Graphically with line plots and surfaces

TotalView excels in working with arrays of aggregate data types. The user interface features a “dive-in-all” capability that makes extracting numerical fields from array-of-aggregate type structures very easy.

Data abstraction with tools like C++ template libraries can be a great thing, but it can also serve to unintentionally

[1] As previously discussed in: *Gottbrath, Chris.* Proceedings of the Cray Users Group 2013. “Debugging and Optimizing Programs Accelerated with Intel® Xeon® Phi™ Coprocessors”

obfuscate what is happening in a program when being debugged. Rogue Wave provides TotalView with automatic translation support for STL list, map, vect, set, multi-set, and multi-map and string classes. These objects transform automatically into harmonious array-, structure-, or array-of-structure type objects. Furthermore, TotalView provides the user with the ability to transform their custom data types in the same manner.

A. MPI debugging

TotalView integrates with the Cray **aprun** command. The user manual provides greater details, but for a high-level overview, users can simply use **qsub** to create an interactive partition on the Cray system and then run:

```
TotalView aprun –a –n<num> a.out
```

The debugger queries **aprun** for information about all the MPI tasks that make up the mpi job and then attaches to all of them.

B. NVIDIA GP-GPU accelerator debugging

TotalView supports debugging codes that take advantage of NVIDIA® GP-GPU accelerators on the Cray XK™, XC and CS™ series supercomputers.[2] TotalView 8.13 supports CUDA 4.2, 5.0, and 5.5 and CCE 8 OpenACC applications. TotalView provides full visibility into both what is running on the host processor and in the GP-GPU accelerator. The control TotalView can exert over the scheduling of the device threads on the accelerator is less than what users may be accustomed to from traditional multithreaded debugging, but it is as much as TotalView is able to provide with the CUDA runtime. Once a warp is launched on a SM, TotalView can control the progress of that warp with stepping commands (individual device threads can't be stepped or run separately from the warp) and breakpoints can be used to halt the running kernel when the first thread hits the breakpoint.

With CUDA 5.5, TotalView supports CUDA dynamic parallelism. Dynamic parallelism is the ability to launch a new kernel from within a kernel already running on the accelerator.

C. Intel Xeon Phi coprocessor debugging

The Intel Xeon Phi coprocessor, which is an available option in Cray XC and Cray CS300 series systems, is also a valid debug target. There are three ways Xeon Phi coprocessors have been used: native (called autonomous by Cray) mode, symmetric mode, and offload mode. These modes are different ways of taking advantage of the Xeon Phi to accelerate computations within an application.

In a native or autonomous mode program the application is compiled specifically for the Xeon Phi and run across the Xeon Phi coprocessors as a hybrid MPI + OpenMP application (usually OpenMP is used, TBB or pthreads could be used instead to provide thread-level parallelism). Running TotalView on a native mode application is very similar to running TotalView on any other MPI application. The only difference is that TotalView runs as a cross-platform debugger, with TotalView running on the Xeon host processor and the application (and TotalView's tvdsvr debug agents) running on the Xeon Phi. In this mode the user's application does not run directly on the Xeon host processor.

In a symmetric mode application the target program being debugged actually runs as a heterogeneous hybrid (MPI + OpenMP/Pthreads/TBB) application across both the host Xeon and Xeon Phi coprocessor nodes. The added complexity here is that some MPI ranks are running on architecture A and some are running on architecture B. With the Xeon and the Xeon Phi, the architectures are very similar, but not exactly the same. Also, load balancing is frequently a challenge since the two node types can be expected to progress at different rates through numerical code. Running TotalView on this is nearly the same, simply launch the MPI application and TotalView resolves which nodes are host processors and which are coprocessors, and does the right things. All the MPI ranks appear in the debugger as part of the same control group.

An offload application treats the coprocessor as a computational accelerator, following more directly the model of languages like OpenACC or OpenMP 4.0. The developer annotates routines they would like to have offloaded to the coprocessor and the Intel compiler and Xeon Phi runtime create a context on the Xeon Phi and transfer data and code to the coprocessor. TotalView supports debugging single node offload programs. The user can freely debug both the host code and the offloaded regions.

D. Memory debugging

TotalView includes MemoryScape, a memory debugger that gives users the ability to detect memory leaks, heap memory allocation overruns, and execute heap memory analysis and optimization. MemoryScape is integrated into TotalView and supports performing memory analysis across the many tasks of an MPI job.

MemoryScape supports the Knight's Corner Xeon Phi coprocessor in parallel programs that are accelerated using autonomous and symmetric mode.

E. Reverse debugging

One of the most unique features of TotalView is its reverse debugging feature. Reverse debugging allows running the program backwards from the point where the failure appears to pinpoint the root cause of that failure.

[2] *Gottbrath, Chris* Proceedings of the Cray Users Group 2012 "Debugging and Optimizing Scalable Applications on the Cray"

TotalView's reverse debugging feature is called ReplayEngine, which allows users to step backwards through the program's execution history while utilizing a record and deterministic replay technique. As the program runs, the tool records program execution, with particular attention paid to non-deterministic inputs such as I/O, thread context switches, and operating system calls. If at any point the user wishes to see the previous state of the process, the tool arranges to place a synthetic unix process in that same state. This is done by creating a copy of the code and data state that was saved earlier and then re-executing the code deterministically along exactly the same execution trajectory that the program took during the record phase.

All of this is managed behind the scenes by ReplayEngine. The user interface simply shows "backwards step" and "backwards continue" commands that can be used to take the process back to earlier states. Once the process has been replayed to the desired state, all the usual process and thread inspection capabilities are usable. Any variable or data (even those that the user did not previously know would be important) can be inspected during this replay process.

The deterministic nature of this replay process makes it especially helpful to track down hard to reproduce or intermittent bugs. These defects, which might otherwise take days or weeks to diagnose without TotalView, can sometimes be resolved within a single session with ReplayEngine.

The Cray CCE compiler on Cray XC systems makes frequent use of AVX instructions to provide performance for numerical codes. The ReplayEngine capability in TotalView 8.12 had limited support for AVX instructions and had difficulty supporting users on the Cray XC. TotalView 8.13's ReplayEngine does not have this problem and can be used on Cray XC systems with ease.

F. *Scripting with TVScript*

TotalView is most often used for interactive graphical debugging, but it also is completely scriptable with a TCL based CLI. This can be used to automate repetitive tasks and drive completely non-interactive debugging sessions.

TVScript is a simple way to drive non-interactive debugging sessions. TVScript is a driver script, written in the TotalView TCL command line interface language, which takes a target executable and a set of instructions about where to set breakpoints, and then drives the target program towards completion. TVScript has an event-action model. An event is triggered each time a program hits a breakpoint. Other events occur when the program reaches certain other specifically defined states, such as program completion, segmentation violations, or memory errors. TVScript driver program can take a variety of different actions in response to these events.

The most frequent action is to report information to a debugging logfile, which can be parsed after the fact to diagnose the behavior of the program. TVScript merges some of the benefits and conveniences of "print" style debugging with the control and capabilities of a powerful interactive debugger.

TotalView, MemoryScape, ReplayEngine, and TVScript are fully supported for Cray XC and CS300 environments.

III. Scalability

Rogue Wave is collaborating with specific customers on a scalability project. The project team members implemented a server tree network using the MRnet technology. The tree allows for scalable broadcast and reduction techniques to be used on communication between the debugger and debug agent processes.

This work provides a foundation to deliver both large absolute scale (debugging across very large total numbers of processes) and responsiveness and performance at scale (fast startup and fast debugger operations when debugging at scale). In terms of absolute scale, TotalView has debugged across 786,432 cores and as many as 1,048,576 threads on the Lawrence Livermore National Laboratory (LLNL) Sequoia system using this new scalable infrastructure.

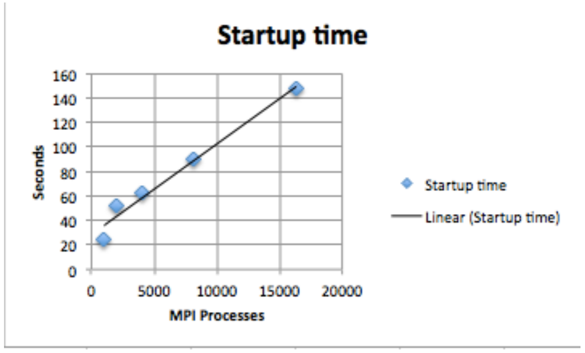
In terms of performance at scale, work is being done to tune the performance of launch and attach in order to optimize the responsiveness of various debugger operations such as stepping, displaying program status, and variable data. Performance at scale varies depending on a number of characteristics, including the linking model (static or dynamic), the total number of debug symbols, and the use of language features like C++ templates. For this reason a collaborative methodology called application driven tuning is used, which involves tuning debugger behavior on real programs at scale on our collaborator's leadership-class supercomputers.

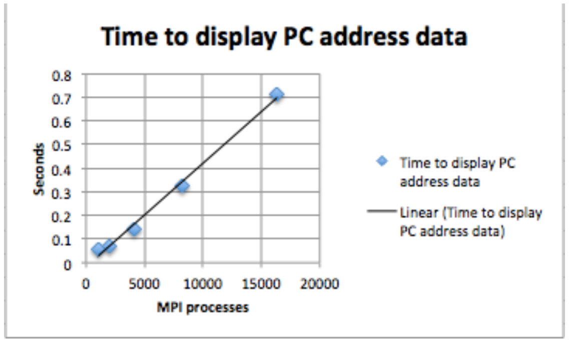
This work is being done across three platforms: the IBM® Blue Gene™, Cray XE, and x86-based Linux® + Infiniband™. The MRnet infrastructure is already in place and users can receive a technical preview of TotalView that includes the MRnet capability by contacting the author. MRnet will be fully folded into the product with documentation in a future release of TotalView.

Optimization of TotalView operational performance using the new infrastructure is ongoing. Rogue Wave is working with a range of different applications and tuning the debugger's performance with respect to those applications.

Rogue Wave conducted testing using the ALE3d multi-physics code from LLNL on the Sierra system. ALE3d is a C++ application and was dynamically linked with 16 shared libraries. Sierra is a 64-bit Linux cluster with Infiniband interconnect and 12 cores per compute node. Runs were conducted with 24 ranks per node. This represents a substantial and realistic target program, far more complex than a "hello_mpi_world" app.

Two things are critical for performance at scale. One is the time to get a parallel session started under the debugger. This start-up time only occurs once per debugging session, but what really matters is the time taken to perform individual debugger operations that will be performed many times during a debugging session. The start-up time includes the time to launch the job, attach the debugger, and stabilize the processes. The second critical item is the time needed for the debugger to perform specific operations. Rogue wave has optimized gathering and displaying the PC address and report the time it takes to request, gather, and display the PC data from all the MPI processes below. It should be emphasized that these are timings gathered during an application driven optimization process that is still ongoing.





This test captures both the start-up time and data aggregation time. Rogue Wave has tested on an XE6 (Hopper) and XK7 (Titan) as part of this larger project. Both tests focused around start-up time. At approximately 16,000 MPI tasks, start-up took about 70 seconds on Hopper with an application called IRS. At the same scale it took about 190 seconds on Titan with an internal test application called tx_mpi_longmsg.

One distinct benefit of the TotalView architecture is that the memory usage on the compute nodes is very low and does not increase significantly as the number of processes or threads on that node increase. This is critical because it reserves the compute node memory for the users application that is being debugged. Alternate techniques have been known to hang or crash the session due to excessive demand for compute node resources in other, less efficient, tools.

IV. CONCLUSION

This paper discussed the current status of TotalView for the Cray XC and CS 300. This involved reviewing the different kinds of debugging TotalView supports: source code debugging of C/C++ and Fortran, MPI debugging, NVIDIA GP-GPU debugging, Xeon Phi debugging, memory debugging, reverse debugging, and batch debugging. This paper highlighted that over the last year Rogue Wave has added support for CUDA 5.0 and 5.5 debugging, Xeon Phi native, symmetric, and offload debugging, Memory debugging on the Xeon Phi, and corrected a frustrating defect in ReplayEngine that made it hard to use on the Cray XC. This paper also gave some specifics on Rogue Wave's work to provide a scalable debugger with great performance and responsiveness both at low and high scale.