Debugging Programs Accelerated with Intel® Xeon® Phi™ Coprocessors

A White Paper by Rogue Wave Software.
Debugging Programs Accelerated with Intel® Xeon® Phi™ Coprocessors

by Rogue Wave Software

© 2013 by Rogue Wave Software. All Rights Reserved

Original Publication – May 2013

Printed in the United States of America

Trademark Information

The Rogue Wave Software name, logo, and TotalView are registered trademarks of Rogue Wave Software, Inc. or its subsidiaries in the US and other countries. MemoryScape and ReplayEngine are trademarks of Rogue Wave Software, Inc. or its subsidiaries. All other company, product, or brand names are the property of their respective owners.

IMPORTANT NOTICE: The information contained in this document is subject to change without notice. Rogue Wave Software, Inc. makes no warranty of any kind with regards to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Rogue Wave Software, Inc. shall not be liable for errors contained herein or for incidental, consequential, or other indirect damages in connection with the furnishing, performance, or use of this material.
TABLE OF CONTENTS
Abstract ........................................................................................................................................... 4
Introduction ......................................................................................................................................... 4
Debugging on an HPC cluster with TotalView ................................................................................. 5
  TotalView for HPC Cluster Environments ................................................................................. 5
  TotalView Support for the Intel Xeon Phi Coprocessor ................................................................. 7
Conclusion ......................................................................................................................................... 10
About Rogue Wave Software ........................................................................................................ 10
Abstract

Intel® Xeon® Phi™ coprocessors present an exciting opportunity for HPC developers to take advantage of many-core processor technology. Since the Intel Xeon Phi coprocessor shares many architectural features and much of the development tool chain with multicore Intel Xeon processors, it is generally fairly easy to migrate a program to the Intel Xeon Phi coprocessor. However, to fully leverage the Intel Xeon Phi coprocessor, a new level of parallelism needs to be expressed which may require significantly rethinking algorithms. Scientists need tools that support debugging and optimizing hybrid MPI/OpenMP parallel applications that may have dozens or even hundreds of threads per node.

This paper will discuss how recent upgrades to TotalView® are setting the stage for HPC developers to adopt the Intel Xeon Phi coprocessor with confidence.

Keywords – Debugging, Optimization

Introduction

The Intel Xeon Phi coprocessor is a key new technology, and many HPC users may find themselves debugging and optimizing code for the Xeon Phi. The Intel Xeon Phi is an instance of what Intel calls the Many Integrated Core (MIC) architecture. As its name suggests, the Xeon Phi is a many-core (rather than multicore) processor and it sports >50 separate processor cores, each of which is capable of executing up to four thread contexts. Programs can be written either to run directly on the Intel Xeon Phi coprocessor, treating it as a >50 core node, or on the host processor with which computationally intense code is offloaded to one or more Intel Xeon Phi coprocessor cards through the use of an offloading mechanism, such as Intel’s LEO (Language Extensions for Offloading), OpenACC, and potentially through extensions to OpenMP that are being discussed right now.

With major vendors like Cray, IBM, HP, and SGI offering HPC solutions based on the Intel Xeon Phi coprocessor as an accelerator, users will be looking at a wide range of codes and considering the challenges of migrating applications to run on the Intel Xeon Phi coprocessor. This paper reviews Rogue Wave Software’s key debugging and optimization technologies that can be used to support development for these HPC clusters and help users take advantage of Intel Xeon Phi coprocessor.

The TotalView debugger supports source code debugging of C, C++, and Fortran, and features capabilities for debugging across the nodes of a distributed memory (cluster-type) supercomputer. This paper highlights recent work that we’ve done to adapt TotalView to work with the Intel Xeon Phi coprocessor.
Debugging on an HPC cluster with TotalView

*TotalView for HPC Cluster Environments*

TotalView provides a powerful and intuitive graphical source code debugging environment for a wide variety of different HPC architectures, including Cray XC, Blue Gene/Q, and infiniband based Linux clusters. In all cases, TotalView gives users full control over and visibility into program execution.

TotalView provides users control over all the parallel tasks that make up the program through a single debugger interface. Process and thread control features allow users to easily synchronize all the processes 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 that users can comfortably 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 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 using Fortran slice notation (even in C). Array data can be displayed in three ways:

1. As memory-ordered elements in list form;
2. 2D slices displayed in spreadsheet format; and
3. Represented 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 obfuscate what is happening in a program when being debugged. Rogue Wave provides TotalView with automatic translation support for STL List, Map, Vect, and String classes. Support for Set, Multi-set, and Multi-map are in the 8.12 version. These objects are transformed automatically into easy to work with 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.
MPI Debugging

TotalView is integrated with a variety of different variations on the `mpirun` or `mpiexec` command. The user manual provides greater details, but for a high-level overview, users should start with an interactive batch session (`qsub -l` on OpenPBS and PBSpro managed systems) to create an interactive allocation and then run:

```
TotalView mpiexec -a -n<num> a.out
```

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

Rogue Wave Software is currently 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 is being done across three platforms: the IBM Blue Gene/Q, Cray XE/XC, 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 TotalView Product Manager. MRnet will be fully folded into the product with documentation in a future release.

Optimization of TotalView’s 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.

While a variety of different tests have been run at different scales and on different architectures, TotalView has already been able to debug more than 1 million threads on the Blue Gene/Q.

Memory Debugging

TotalView includes the MemoryScape 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 has been extended to support the Xeon Phi, for several but not all uses.

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 the root cause of that failure.

TotalView’s reverse debugging feature is named ReplayEngine, which allows users to step backwards through the program’s execution history utilizing a record and deterministic replay technique. As the program runs, the tool operates in a record mode in which program execution is recorded, 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. It does this 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.

These activities are 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 (including those not known to 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, are sometimes resolved within a single ReplayEngine session.

**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 such non-interactive debugging sessions. It 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 power and capabilities of a powerful interactive debugger.

**TotalView Support for the Intel Xeon Phi Coprocessor**

Intel Xeon Phi coprocessors can be used either in an accelerator-like mode or in a mode that more closely resembles separately addressable multicore nodes. The workflows are a bit different, as are the ways that the Intel Xeon Phi coprocessor appears in the user interface.

Each mode of use is briefly discussed below.

**Treating the Intel Xeon Phi Coprocessor as a Hosted Multicore Node**

There are a couple of ways that users can utilize the Intel Xeon Phi coprocessor as a multicore Linux node. The Intel Xeon Phi coprocessor runs a separate OS instance, usually a special version of Linux, from the OS running on the host node. It is common to log directly into that Linux instance and run applications there. Code can generally be cross-compiled on the host system using the Intel compilers by specifying that the desired target architecture is MIC (the Intel compiler uses the –mmic flag for this mode). An application compiled this way can be manually run in the traditional fashion on the Intel Xeon Phi coprocessor. This mode of use is called “native mode.”
Alternately, if a system is set up correctly, users can utilize `mpiexec` to run a parallel job. Again, the job will be compiled with the `-mmic` flag, and users may need to direct the MPI or resource management system to run the MPI tasks specifically on the Intel Xeon Phi coprocessors.

TotalView supports remote debugging and is compatible with a variety of MPI launcher programs. Rogue Wave adapted TotalView to display the instructions and registers used on the MIC architecture. In addition, Rogue Wave compiled a variant of the TotalView debug server so that it would run on the Intel Xeon Phi coprocessor and to provide MPI and remote-system, cross-debugging support for MPI. TotalView’s main executable runs on the host Linux-x86-64 environment when users are debugging Intel Xeon Phi coprocessors.

Users will need to ensure that their program executable is accessible from the host environment. They also need to confirm that TotalView is accessible from both the host processor and the Intel Xeon Phi coprocessor node. This software can be placed on a file-system volume that is mounted both on the host and on the Intel Xeon Phi coprocessor device. This is not an unusual configuration.

Then, users need to verify that they can directly address the Intel Xeon Phi coprocessor device(s) on which the processes that will be debugged are running. Users need to do this from the machine in which they are planning to run TotalView. A user’s debugging session, right now, will be limited to the set of Intel Xeon Phi coprocessors that they can directly address from the host node. Rogue Wave is examining what it will take to support situations in which a cluster of host nodes exists, each of which can only address its own Intel Xeon Phi coprocessors.

The final condition is licensing. Those interested can contact Rogue Wave to find out if additional tokens are needed to debug on Intel Xeon Phi coprocessors.

If those conditions are met, users can simply direct TotalView to debug a remote process, which happens to be running on the Intel Xeon Phi coprocessor, or they can launch an MPI parallel job on the host machine.

Launching TotalView to debug a native, non-MPI application uses a command line such as:

```
totalview -r <host> ./a.out
```

The `-r` flag simply tells the debugger that the target program is remotely running on the host named in the parameter.

Launching TotalView to debug an MPI job uses a command line in the form of:

```
totalview -args mpiexec <mpiexec args> ./a.out
```

This is the same concept as with many non-Intel Xeon Phi coprocessor launch scenarios. Users are starting up `mpiexec` under TotalView, and when the parallel job is launched, an interface is used between the debugger and `mpiexec` so that `mpiexec` tells the debugger the location of the tasks that comprise the job. (Please see the TotalView documentation for more example launch strings.)
In either case the debugging experience that users receive on the Intel Xeon Phi coprocessor is very much what would be expected while debugging native code on the host processor. In the likely event that the program running on the Intel Xeon Phi coprocessor uses OpenMP or some other threading discipline, users will want to take advantage of TotalView’s thread control capabilities.

The MemoryScape memory debugging capability, discussed above, supports debugging Native and Symmetric mode applications. Note that the limited amount of memory available per Xeon Phi nodes, and the relatively large number of threads, makes memory optimization an important challenge.

Working with a Heterogeneous Set of Cluster Nodes

The forthcoming TotalView 8.13 also supports a variation on this model. Users can run an MPI application heterogeneously across both the Xeon host processor and the Xeon Phi coprocessor nodes. This is called “symmetric mode” and it gives the potential benefit of being able to take advantage of the computational power of both the host Xeon nodes and the Xeon Phi nodes that are hosted. The potential downside is that load balancing an application may be more complex due to the inhomogeneous set of compute resources.

Treating the Intel Xeon Phi Coprocessor as an Accelerator

The alternative way users are encouraged to utilize Intel Xeon Phi coprocessors, especially when adapting large programs that already run on the host processor, is to take the existing code and simply add directives that provide the compiler with hints on how to act if an Intel Xeon Phi coprocessor is present. Then certain units of work could be offloaded to the coprocessor. These extensions are supported by the Intel compilers. Other similar models, based on language extensions such as OpenACC, OpenCL, and OpenMP, are being developed by a variety of vendors.

TotalView was extended to handle this situation by having the capability to recognize when the host program is dispatching a new routine to the coprocessor. When it recognizes that this is happening, the debugger will automatically pause the application and start up a debug agent on the Intel Xeon Phi coprocessor. This happens behind the scenes, and TotalView ends up attached to and controlling both a local process (the host process that dispatched the work to the Intel coprocessor) and a process running on the coprocessor (the one that resulted from the dispatch of work). The host and offload processes will show up in the debugger as separate processes running on distinct nodes.

The Intel compiler will generally create two copies of code designated to be offloaded to the Intel Xeon Phi coprocessor. One copy is compiled for the host Xeon processor and is used in the event that the code is run on a machine without an Intel Xeon Phi coprocessor. The other copy is compiled to run on the Intel Xeon Phi coprocessor. These two distinct function implementations (for two different processor architectures) are conceptually the same to the user.

TotalView recognizes this fact by ensuring that breakpoints are shared across both function instances. Therefore, users can set breakpoints on offloaded functions in the host process even before work is dispatched. When the program later runs and the function is loaded on the Intel Xeon Phi coprocessor, TotalView recognizes that these functions are conceptually the same and will automatically apply a corresponding breakpoint to the program running in the coprocessor.
In terms of features, there is not much difference between offload and remote native mode debugging with TotalView on the Intel Xeon Phi coprocessor, except the fact that users may end up actively debugging both on the host and offload processes.

TotalView support for Intel Xeon Phi coprocessors officially began with TotalView 8.12. This document describes this current level of functionality. One significant improvement that will be delivered in early 2014 is support for debugging symmetric mode jobs that run as a single heterogeneous MPI job across both the host and Xeon Phi nodes.

There are a few limitations for debugging with TotalView on the Intel Xeon Phi coprocessor. Two major features are not available on the Xeon Phi with 8.12: ReplayEngine and MemoryScape. MemoryScape memory debugging for Native and Symmetric mode programs will be available in 8.13, which will be released in early 2014. ReplayEngine support is being explored but has not yet been scheduled.

Conditional watchpoints are still to be supported for the Intel Xeon Phi coprocessor. Users can utilize watchpoints and conditional evaluation points, but not conditional watchpoints. Finally, there are a few Intel Xeon Phi coprocessor specific instructions and registers that are not disassembled and displayed correctly at this time. See the release notes and product documentation for details.

Conclusion

TotalView provides a seamless and polished Intel Xeon Phi coprocessor debugging experience, which allows HPC developers to more easily and efficiently port and maintain scientific codes for any machine accelerated with an Intel Xeon Phi coprocessor.

About Rogue Wave Software

Rogue Wave Software, Inc. is the largest independent provider of cross-platform software development tools and embedded components. Rogue Wave application development products reduce the complexity of prototyping, developing, debugging, and optimizing multi-processor and data-intensive software applications. Rogue Wave customers include industry leaders in the Global 2000 as well as leading government institutions and universities. Rogue Wave is a portfolio company of Audax Group. For more information, visit www.roguewave.com.