TotalView 2019.2 Release Notes

Updated: July 2019

These release notes contain a summary of new features and enhancements, late-breaking product issues, migration from earlier releases, and bug fixes.

PLEASE NOTE: The version of this document in the product distribution is a snapshot at the time the product distribution was created. Additional information may be added after that time because of issues found during distribution testing or after the product is released. To be sure you have the most up-to-date information, see the version of this document on the Rogue Wave web site:

http://www.roguewave.com/support/product-documentation/totalview.aspx

Additions and Updates

TotalView Reverse Connect

The organization of modern HPC systems often makes it difficult to deploy tools such as TotalView. For example, the compute nodes in a cluster may not have access to any X libraries or X forwarding, so launching a GUI on a compute node is not possible.

Using the new Reverse Connect feature, you can easily establish a connection where the TotalView UI is running on a front-end node and debugging a job executing on compute nodes.

The basic process is to embed the **tvconnect** command in a batch script; when the batch job runs, the **tvconnect** process connects with the TotalView client to start the debugger server process on the batch node. The TotalView client would typically run on a front-end node, where the application is built and batch jobs are submitted. For more information, see Chapter 18, "Reverse Connections," in the *TotalView User Guide*.

Debugging Through Starter Shell Scripts

Often developers will wrap the startup sequence of their application with a shell script in order to set up the run time environment for the application and then run it. The scripts would often need a special "debug" switch in order to start the application under TotalView. In the 2019.2 release, TotalView has been enhanced to understand when given a shell script as the debug target and will chain through the startup sequence of the shell script, including recursing through multiple startup scripts if needed, to find the resulting target that should be debugged.

Python pybind/pybind11 Debugging Support

TotalView's mixed language Python and C++ debugging support now properly identifies pybind/pybind11 callstack intermediate frames and will remove them so that a clean call sequence between Python and C++ is presented.

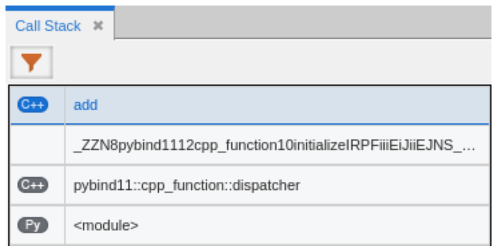


Figure 1 - pybind intermediate frames

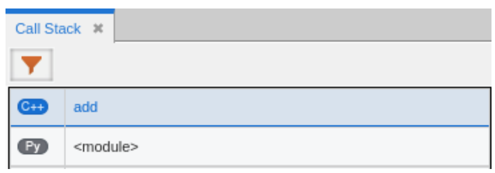


Figure 2 - Cleaned up pybind frames

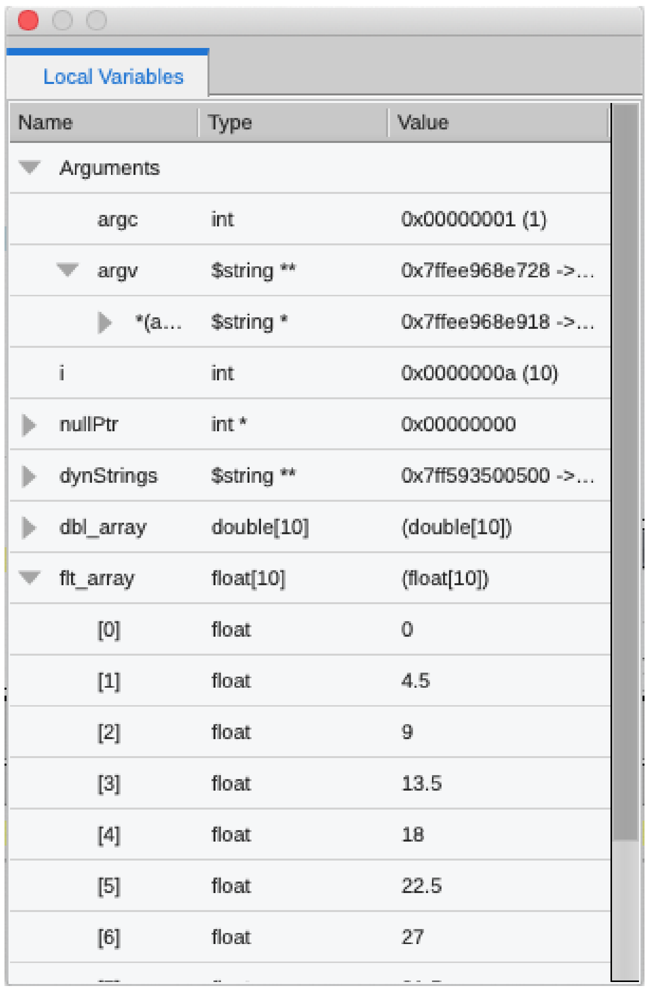
Automatic Process/Thread Breakpoint Focus Preference Change

In past releases, the "Automatically focus on threads/processes at breakpoint" preference was defaulted to false. This often creates confusion during a debugging session when a thread or process not in focus hits a breakpoint and stops. In this case the process/thread is not automatically brought into focus. Users have to manually examine the set of processes and threads and change the focus. For the 2019.2 release we have changed the preference default to true so that any process or thread that hits a breakpoint will be brought into focus in your debugging session. If you prefer the old behavior, simply change the preference in the Action Points tab in the Preferences dialog or add "dset TV::GUI::pop_at_breakpoint false" to your .tvdrc file.

New User Interface Improvements

TotalView's new user interface, activated through the Display Preferences panel or using the -newUI command line option, continues to deliver new enhancements to make debugging your applications even easier. If you have any feedback about the new user interface, requests for new or missing features or any problems please send email to tv-beta@roguewave.com.

- Local Variable View
  - In previous releases the Local Variable tab was part of the Call Stack View and locked in a set position as part of the view. For the TotalView 2019.2 release the display of local variables has been broken out into its own view. This enables users to move the view to new locations or even undock it completely. In addition, the local variable view functionality has been overhauled and allows structures to be expanded in place.



- New Input/Output View available
  In the TotalView 2019.1 release the ability to debug applications requiring standard input was added. The default use case is for users to enter the input in the terminal that TotalView as launched in. In the 2019.2 release, we've added the ability to enter this input directly in the user interface instead of through the terminal. To activate this workflow, turn on the "Input Output view support" in the RW Labs tab on the Preferences

dialog and restart the debugger. A new Input/Output view will be available that allows you to enter input to your running application.

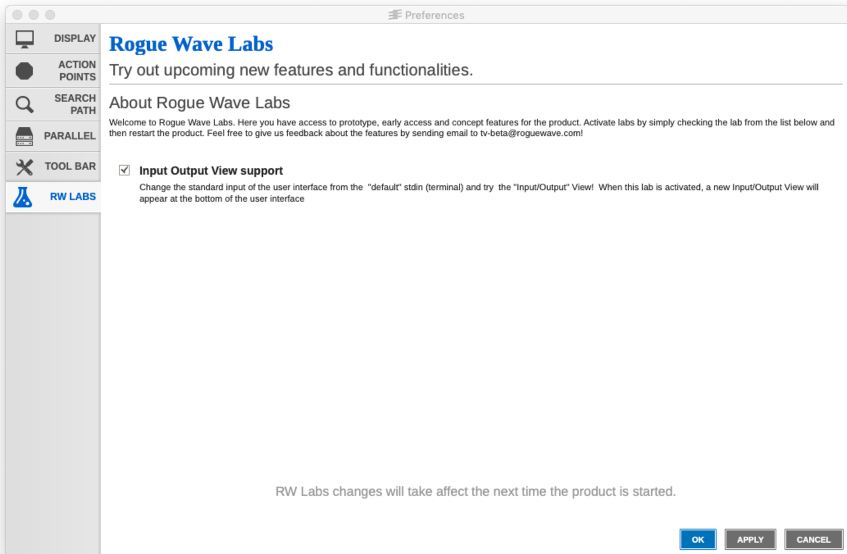


Figure 3 - Activate the new Input/Output View under RW Labs

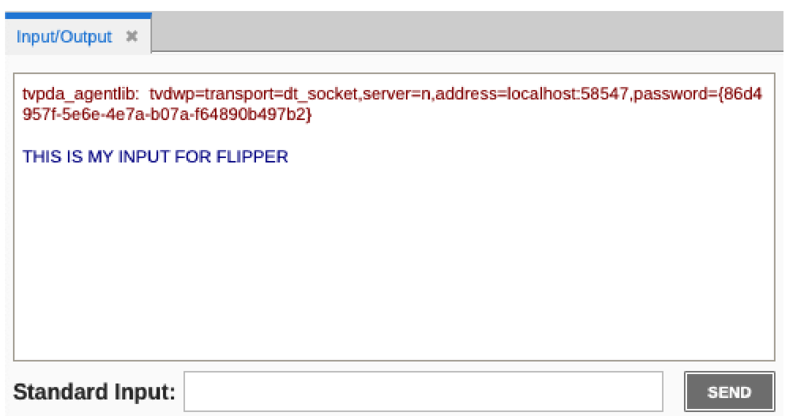


Figure 4 - Enter input to your application in the new Input/Output view

- Change new UI font size

  The ability to change the overall font size for the user interface has been added to the TotalView 2019.2 release. To change the font size, bring up the Preferences dialog and select the Display tab. Use the Font Size slider to change the font size to small, medium, large or extra-large. The default font size is medium. The next time you start TotalView the

UI will use the specified font size.

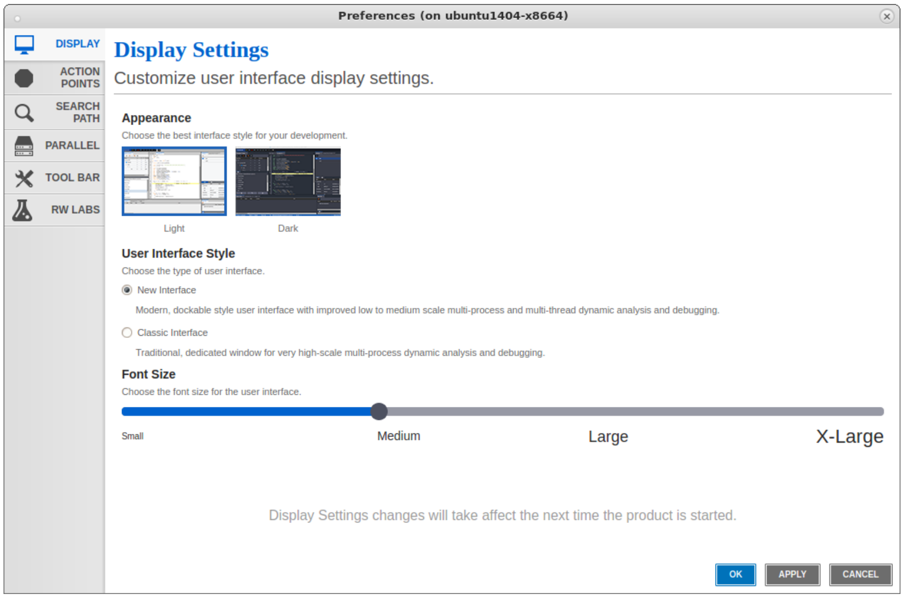


Figure 5 - Font Size slider on Preferences Display tab

- Bug fixes and performance improvements
  Numerous bug fixes and performance improvements have been made to the new UI including properly focusing on main for MPI, execve and Fortran applications.

Platform Updates

TotalView 2019.2 introduces support for the following platforms:

OS:

- Fedora 30

Compiler:

- GCC 9

Bug Fixes for 2019.2

TODO

Deprecation Notices

Linux Power

Starting with release 2019.3, TotalView will change the base build platform to RHEL 6.5.

IBM Blue Gene/Q

Starting with release 2020, TotalView will no longer support the IBM Blue Gene/Q platform.

Intel Xeon Phi Offload Coprocessors

Starting with release 2020, TotalView will no longer support the Intel Xeon Phi offload coprocessors.

Known Issues

New User Interface

Linux ARM64 and Linux PowerLE Help

When using the new user interface on Linux ARM64 and Linux PowerLE, in application help is not available. Help is always available online at https://docs.roguewave.com/en/totalview/current/.

Python Debugging

Anaconda

TotalView supports debugging of the python interpreter in release 4 of Anaconda but is not working with the recent release of Anaconda 5. Something in the way they build the python interpreter has broken the ability to debug python.

Ubuntu 16.04

When debugging python on Ubuntu 16.04 TotalView is not detecting the python interpreter automatically and does not turn on python filtering. Filtering can be turned on by clicking the "filter" icon in the toolbar of the Call Stack view.

Licensing

TotalView releases built with FlexNet Publisher 11.13.1 must have licenses served by a license server at 11.13.1 or higher

FlexNet Publisher client library version 11.13.1 is built into our recent releases. This means, according to FlexNet Publisher's component version compatibility rules (near the end of FNP's License Administration Guide PDF), the license server *must be* at v11.13.1 or higher. Although these rules have long been in place with no problems, we've recently been receiving reports of license checkout failures when using the license server v11.12.1 from previous TotalView releases. In this case the vendor daemon's debug log file shows "(toolworks) Request denied: Client (11.13) newer than Vendor Daemon (11.12). (Version of vendor daemon is too old. (-83,21049))". As noted in FNP's License Administration Guide PDF, this issue can be avoided by making sure our latest license server components are in place.

TotalView sometimes cannot acquire license due to FlexNet bug

If you are using Linux Power or AIX then you are still affected by the bug in the FlexNet Publisher software that results in TotalView's inability to acquire a license when your license file contains multiple licenses with different maintenance expiration dates (i.e. the

4th field on the INCREMENT line). The licensing software skips some of the licenses in this case. If you know that your license file is being read and is correct, and you think you might be running into this bug, we recommend that you add the "sort" keyword and value (such as sort=1, sort=2, sort=3) to each INCREMENT line in the license file in any order. This bug has been reported to Flexera and is identified as SIOC-000145042.

Here is an example of adding the "sort" keyword:

```
SERVER linux-power 0050569b402c

VENDOR toolworks

INCREMENT TotalView_Enterprise toolworks 2014.1231 permanent 1 \
        sort=1 VENDOR_STRING="processors=16 platform=linux-power" \
        SIGN=3350A26C395A

INCREMENT TotalView_Enterprise toolworks 2015.1231 permanent 1 \
        sort=2 VENDOR_STRING="processors=16 platform=linux-ia64" \
        SIGN=C7D11FB667C8

INCREMENT TotalView_Enterprise toolworks 2016.1231 permanent 1 \
        sort=3 VENDOR_STRING="processors=16 platform=linux-x86_64" \
        SIGN=BEC7534A248A
```

FlexNet Embedded Licensing Technology

The Linux ARM64 and Linux PowerLE platforms require FlexNet Embedded for their licensing technology while the remaining TotalView platforms use FlexNet Publisher. In order to share tokens across all of the TotalView platforms we have also been porting them to support FlexNet Embedded as well. If you need to share tokens across Linux ARM64, Linux PowerLE and the other TotalView platforms, customers should contact license@roguewave.com to set up a new license file for the FlexNet Embedded license server.

FlexNet Embedded License Server MTU Requirements

The license server used for the FlexNet Embedded technology appears to need a larger Maximum Transmission Unit (MTU) setting than the default on some systems. We have found with some customers that a value of 1,500 was adequate for successful network transmissions between the license server and TotalView.

macOS

Physical console access needed when running TotalView on macOS High Sierra

Due to new security changes in macOS High Sierra, TotalView will only run from the console and cannot be run through a remote desktop technology such as VNC. We are still assessing what changes need to be made to TotalView so that it will run remotely on macOS High Sierra systems.

With multiple displays attached to a macOS machine, some TotalView windows may not be noticeable

When a user attaches an external monitor to a running Macbook Pro, or adds multiple displays to a Mac, window rearrangement may move some windows off-screen. This may result in a TotalView modal window not being found until you use the Mac command to display all the windows (Mission Control). This appears to be an interaction between XQuartz and Darwin. It has been seen in Mavericks, but it's possible it will show up in other releases. There may be a workaround in System Preferences->Mission Control by disabling "Displays have separate Spaces."

Physical console access needed when starting TotalView

Starting in Mountain Lion, OS X security policies require that users meet a password challenge in order to use TotalView, and the challenge can be issued only to the console (the OS X desktop). After the password challenge is met once, you can run TotalView repeatedly from the same login session without further challenges.

It is possible to work around this need for physical access with the following steps. The first few of these are likely already set in order to allow TotalView to run.

- Install XQuartz and TotalView
- Ensure every user needing debugging is in the _developer group
- Allow X11 forwarding in the sshd_config file (disabled by default)

- In a terminal window enter the following two commands:
  - DevToolsSecurity -enable (this step is optional if this was already enabled)
  - sudo security authorizationdb write system.privilege.taskport allow

Visualizer fails under macOS Sierra and Xquartz

Attempts to use the visualizer tool fails with a message 'Error: attempt to add non-widget child "dsm" to parent "vismain"' which supports only widgets.

Mojave Support

The majority of users who install TotalView on Mojave will be able to debug successfully. We have seen one case where the user runs into an error starting up and the error says that the system call task_for_pid() is not working properly. This can be worked around by running TotalView under the sudo command. The issue is believed to be due to an incorrect setup of the keychain access and is being investigated.

Linux

Intel 17 and 18 compilers not generating debug information for a declared integer

The Intel 17 and 18 version compilers are not generating proper debug information for declared integers in Fortran applications. As a result, TotalView is not able to proper evaluate the variable expression and display the variable values. This is a compiler bug but a workaround is available by simply adding the `-debug extended` compilation flag option which adds symbols for local scalar variables and parameters.

Split-DWARF and .gdb_index support, and related options

State variable **TV::dwarf_global_index** is a boolean flag that controls whether or not TotalView considers using the DWARF global index sections (.debug_pubname, .debug_pubtypes, .debug_typenames, etc.) in executable and shared library image files. It defaults to **true**. It may be useful to set this flag to **false** if you have an image file that has incomplete global index sections, and you want to force TotalView to skim the DWARF instead, which may cause TotalView to slow down when indexing symbol tables. Command option **-dwarf_global_index** sets the flag to **true**, and **-no_dwarf_global_index** sets the flag to **false**.

State variable **TV::gdb_index** is a boolean flag that controls whether or not TotalView considers using the .gdb_index section in executable and shared library image files. It defaults to **true**. It may be useful to set this to **false** if you have an image file that has an incomplete .gdb_index section and you want to force TotalView to skim the DWARF instead. Command option **-gdb_index** sets the flag to **true**, and **-no_gdb_index** sets the flag to **false**.

ReplayEngine On-Demand Records Can Show Invalid Stack Trace

In some circumstances in which a ReplayEngine debugging session is driven to the beginning of recorded history, the debugger will display an invalid stack trace and stack frame. We have observed this when debugging a ReplayEngine recording file that was created during a live debugging session in which ReplayEngine was enabled on-demand.

To recover a valid stack, simply step or continue the session - that is, move forward in history. If the beginning of history is specifically of interest, it can be reached directly by opening a CLI window and issuing the command "dhistory -go_time 1".

OpenMPI 1.8.4 with ReplayEngine enabled on older Linux releases.

We have observed a problem with the combination of Replay Engine, Open MPI 1.8.4, and older Linux releases such as RHEL5 (Red Hat Enterprise Linux). When the MPI runtime system closes shared libraries during its startup, a munmap(2) system call may attempt to unmap memory which is in use by Replay Engine. The error message "Unsupported memory access with syscall (11). Conflict with replay private internal memory." is displayed. This error is unrecoverable, but it may be possible to use Replay Engine on the same application by enabling it after the application has completed its MPI_Init call. We have not seen this problem with newer Linux releases such as RHEL6.

TotalView Message Queue and Intel MPI 5.0

By default, the TotalView Message Queue will not work with Intel MPI 5.0 without setting correctly LD_LIBRARY_PATH to the Intel MPI debug libraries. This can be done by sourcing one of the “mpivars.sh/csh” scripts provided by Intel with an added “debug” argument. For example, issue the command “source PATH/impi/5.0.3.048/bin64/mpivars.sh debug”, making sure to replace PATH with the path to your Intel MPI compiler installation. TotalView will then properly pick up the MPI message queue information and display it in its Message Queue window.

Memory Debugging and Intel MPI 5.0+

If a user wants to do memory debugging and they statically link their MPI program with the Intel MPI 5.0+ libraries, MemoryScape will detect a Double Allocation error. This is because, starting with Intel MPI 5.0, the MPI libraries redefine free() and MemoryScape depends on the system free() to see the deallocations. To work around this problem, the user will need to link dynamically or fall back to the Intel MPI 4.0+ libraries.

Debugging IBM Platform MPI Jobs in TotalView

Users have seen some issues when trying to use TotalView on an IBM Platform MPI job. If you try to launch the job from the Session Manager, or the Parallel Tab of the Startup Parameters window, TotalView may show an error that the target program has crashed while trying to load shared libraries. When run under mpirun, this error does not show since mpirun sets up the environment correctly. One can avoid the problem by setting the environment variable LD_LIBRARY_PATH to add the path to the library directory containing the missing libraries. The libraries should be in the ‘lib’ directory that is the same level as the ‘bin’ directory containing mpirun.

While testing the above issue it was noted that the classic launch method of

```
totalview  mpirun –a –np 4 ./foo
```

did not appear to work correctly. TotalView would attach to all the processes, but only rank 0 was held at the point where the job went parallel. The other processes would run to a point where they were waiting on rank 0. To work around this, launch through the GUI as described above, or use the –tv switch for mpirun

```
mpirun –tv –np 4 ./foo
```

Newer Linux kernels that prohibit non-root access to /proc/self/pagemap and ReplayEngine

If non-root access to /proc/self/pagemap is prohibited, the ReplayEngine will emit an ignored assertion warning when the program being debugged enters record mode for the first time. Furthermore, any unknown syscalls will subsequently be handled a little more slowly.

The change affects Ubuntu 15.04 and likely other new distribution releases.

While using ReplayEngine, attaching to 32-bit application from 64-bit hosts sometimes fails

On some 64-bit hosts, attaching to a 32-bit target fails and results in a crash. The underlying technology behind ReplayEngine assumes that in a 64-bit environment, the target is also a 64-bit application and was not explicitly designed to support a mixed environment.

Benign Warning Messages Displayed when ReplayEngine is run on SuSE Linux Enterprise Server 11 with Service Pack 1. (SLES 11 SP1)

As of the 2016.06 release, ReplayEngine no longer fails when attempting to do replay mode operations (move the target backward into history) on Linux x86-64 platforms running SuSE Linux Enterprise Server 11 with Service Pack 1 but some warning messages such as the following are displayed:

```
127083  client/set_current_child.c:456:set_current_child_fn
[78347:78347]: Failed to restore process name for pid 78357: -5

127084  client/set_current_child.c:179:set_current_child_fn
[78347:78347]: Failed to set process name for pid 78357: -5
```

These messages are benign, and your reverse debugging session will work normally.

We have only observed this problem on SLES 11 SP1. It is possible however, that other platforms running the same Linux kernel version (2.6.32.12-0.7) will also run into this problem. The only available workarounds are to update the OS to a more recent release (for example, installing Service Pack 2).

std::string shows as opaque value compiled with Clang 3.5

When trying to debug a program compiled with Clang 3.5 that uses a std::string variable, the variable is listed as type std::string:64 with a value of

Opaque std::basic_string<char,std::char_traits<char>,std::allocator<char>>

When the same program is compiled with the Clang 3.3 compiler, the std string is seen as a simple STL container, and the string value is seen as 'abcd'.

Clang supports a number of optimizations to reduce the size of debug information in the binary. These optimizations work based on the assumption that the debug type information can be spread over multiple compilation units. For instance, Clang does not emit type definitions for types that are not needed by a module and could be replaced with a forward declaration. Further, Clang only emits type info for a dynamic C++ class in the module that contains the vtable for the class.

It has been found that using the **-fstandalone-debug** option which turns off these optimizations works around the problem with the opaque value above.

The **–fstandalone-debug** option is useful when working with 3rd-party libraries that don't come with debug information.

Note that Clang never emits type information for types that are not referenced at all by the program.

-fstandalone-debug is the default on macOS.

-fno-standalone-debug is the default on Linux-x86-64. To work around the opaque value problem above, use the **–fstandalone-debug** option.

Linux - Ubuntu

Memory debugging by linking against the TotalView libraries may not work

There are a number of cases in which it is recommended to link the Heap Interposition Agent (HIA) into the target program to allow memory debugging without having to enable it in the GUI each time. Starting in Ubuntu 12, the linker does not link in libraries that are not directly used by the program. This means that the link line for the agent, with TVLIB pointing to the TotalView library,

gcc –g –o memprog memprog.c –L$TVLIB –ltvheap –Wl,-rpath,$TVLIB

may not pick up the HIA. Instead, add the option "-Wl,--no-as-needed" before the inclusion of the tvheap library. The new compile/link line will look like

gcc –g –o memprog memprog.c –Wl,--no-as-needed –L$TVLIB –ltvheap –Wl,-rpath,$TVLIB

CUDA

CUDA 8.0 Debugger API Internal Error

Certain NVIDIA driver versions associated with CUDA 8.0 may provoke "CUDA Debugger API internal error" messages from TotalView or CUDA-GDB. Known driver versions that have this problem are r361, r367, and r375, however the problem may exist in other driver versions. The internal error typically involves debugging a multi-thread application, when multiple host threads in the process are launching CUDA kernels. NVIDIA has confirmed the driver error, and plans to release a fixed driver version. A release date is not yet available. If the problem occurs, TotalView may print a message and the program may appear to hang.

NVIDIA Pascal Unified Memory Debugger Internal Error

CUDA applications running on Pascal under the debugger may cause a debugger internal error (for example, a SEGV) when the application process exits. Known driver versions that have this problem are r361 and r375, however the problem may exist in other driver versions. The debugger internal error typically involves debugging a CUDA application that exits after using unified memory. NVIDIA has confirmed the driver error, and plans to release a fixed driver version. A release date is not yet available. If the problem occurs, TotalView will exit with an internal error.

Dynamic parallelism not fully supported

With CUDA, we have limited support for dynamic parallelism. We plan improvements to our functionality for displaying the relationships between dynamically launched kernels and navigating the various running kernels.

Layered textures not supported

TotalView does not yet support CUDA layered textures. If you try to examine a layered texture in the TotalView Data Pane, a “Bad address” message will be displayed and you will see “ERROR: Reading Texture memory not currently supported” displayed on the console. If you require layered textures support, please contact TotalView support at support@roguewave.com and let us know how you are using textures so we can develop the best solution to support you.

Solaris

Oracle Studio 12u4 – TotalView unable to evaluate virtual function calls

When debugging applications compiled with the latest version of the Oracle Studio 12u4 compiler, TotalView is unable to call virtual functions through its expression system. This appears to be a shortcoming in debug information from the compiler and should be addressed in cooperation with the Oracle compiler team.

SGI

Memory debugging MPI programs on SGI systems needs special linking

The TotalView and MemoryScape memory debugging Heap Interposition Agent (HIA) technology conflicts with the SGI memory manager when used in MPI programs. The easiest way to get around this problem is to disable the SGI memory manager by unsetting the MPI_MEM_ALIGN environment variable. Without this variable set, the SGI memory manager will not be loaded and the HIA will work correctly, enabling memory debugging to take place.

Cray

Debugging your program within supercomputer environments can often be challenging. Reference the sections below to learn pointers on how to successfully enable memory debugging and perform reverse debugging on your program within a Cray environment.

Memory debugging on Cray systems

Use the pointers below to help achieve a successful memory debugging session within the Cray environment:

- **Install TotalView on a shared file system visible to the Cray compute nodes.**
  In order for the required memory debugging shared libraries to be located when your program is running on a compute node it is best to install TotalView on a shared file system.

- **Cray TotalView Support Module is not required for TotalView 8.15.0.**
  As of TotalView 8.15.0 and its use of the MRNet tree framework TotalView no longer requires the Cray TotalView Support Library to be installed in order to run. If the

MRNet tree framework is turned off then the Cray TotalView Support Module will be required.

- **Statically linking your program against the tvheap_cnl library.**
  One of the most foolproof ways of enabling memory debugging is to statically link against the tvheap_cnl library that is shipped with TotalView. The static library fully supports multithreaded applications. Statically linking your application with the tvheap_cnl library will automatically enable memory debugging in your program when debugged under TotalView and MemoryScape. See the "Linking Your Application with the Agent" discussion in the User Guide for more information on how to statically link your applications with the library.

- **Dynamically linking your program against the tvheap_cnl library.**
  It is possible to dynamically link your application against the dynamic version of the tvheap_cnl library. In this scenario the tvheap_cnl library must be visible to the Cray Compute Node systems, either through a shared file system or by the Cray environment automatically staging the applications shared library dependencies on the compute node.

- **Do not enable memory debugging on the aprun starter process.**
  Turning on memory debugging for the aprun starter process will cause it to fail and prevent the job from starting. The proper way to enable memory debugging is to use the static or dynamic linking options described above.

Reverse debugging on Cray systems

Use the pointers below to help achieve a successful reverse debugging session within the Cray environment:

- **Do not enable reverse debugging on the aprun starter process.**
  Turning on reverse debugging for the aprun starter process will cause it to fail and prevent the job from starting. The proper way to turn on reverse debugging is to launch your parallel job and reach a breakpoint in the code and then dynamically turn on the Replay Engine reverse debugging option. It will begin recording the execution of the program from that point forward.

- **Reverse debugging not working on inter-node jobs on Cray systems**
  When debugging processes within an inter-node job, the reverse debugging engine will crash, causing the debugging session to become unusable and TotalView exits.