TotalView 8.15.10 Release Notes

Updated: October 2015

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

C++11 Support

With the 8.15.10 release TotalView now supports C++11 features for the GNU compiler including support for lambdas, transformations for smart pointers, auto types, R-Value references, range-based loops, strongly-typed enums, initializer lists, user defined literals and transformations for many of the containers such as array, forward_list, tuple and others!

Linux PowerLE Support

TotalView now supports the Linux PowerLE (Little Endian) systems. All major debugging support is provided, including memory debugging, Remote Display Client, CUDA and MPI debugging.

NVIDIA CUDA 7.5 Support

TotalView 8.15.10 continues to improve its CUDA support with support for CUDA 7.5.

Platform Updates

TotalView 8.15.10 introduces support for the following platforms

- **Platforms:**

  ### Linux PowerLE

  The TotalView 8.15.10 release provides support for the new Linux PowerLE (Little Endian) systems.

- **Compilers:**

  GCC 5.2

  Intel 16.0

Bug Fixes for 8.15.10

TVT-15035 -std=c++11 Unable to dive on shared_ptr

TVT-18666 -std=c++11 & gcc 4.9/5.2 STL maps and sets not transforming

TVT-18722 -std=c++11 Invalid compilation scope error raised when diving on a variable in a function of type "auto"

TVT-19234 Starting with Totalview 8.15.0 using Intel MPI (any version) with -tv gets an immediate Internal Error.

TVT-19301 Debug information referencing a negative register number (generated by the Clang 3.2 compiler) can cause TotalView to raise a fatal error: ran off end of chunk.

TVT-19353 TotalView fails to execute in a "Digital Guardian" environment.

TVT-19586 TotalView fails to find libstdc++.so.5 when trying to instantiate an MRNet tree

TVT-19800 MemoryScape gets a floating point exception when using Guard Blocks and the program has a calloc of zero size.

Also, miscellaneous customer reported documentation errors including **TVT-19507, TVT-19588** & **TVT-19589.**

Known Issues

Licensing

Trouble executing licensing scripts on Linux hosts due to LSB 3 compliance requirement

When running the new licensing software (flexlm-11.11 or flexlm-11.12) the scripts may not be able to run the lmhostid or lmgrd commands. This version of FlexNet Publisher software requires a run-time linker with a file name that denotes Linux Standard Base 3 compliance. This requirement can be met by installing the "lsb-3" (or higher) package for the host where the licensing software resides or simply creating the appropriately named symbolic link (e.g. /lib/ld-lsb.so.3, /lib64/ld-lsb-x86-64.so.3, or /lib64/ld-lsb-ppc64.so.3 which should link to the corresponding ld-linux dll for that platform).

TotalView sometimes cannot acquire license due to FlexNet bug

If you are using Linux Power, Linux Itanium, or AIX then you are still affected by the bug in the FlexNet Publisher software that results in TotalView not being able to acquire a license when your license file contains multiple licenses with different maintenance expiry dates (i.e. the 4th field on the INCREMENT line). The licensing software seems to skip over some of the licenses in this case. After ascertaining 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
```

Mac OS X

With multiple displays attached to a Mac OS X 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 one uses 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 only be issued to the console (the OS X desktop). After the password challenge is met once, TotalView can be run 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

TotalView Remote Display Client hangs

If you are using the Remote Display Client on Mac OS X Mountain Lion or Mavericks the application may freeze if more than one Terminal window is open. To get around this issue set the display number in the Terminal's Advanced Options dialog to a value that will not conflict with another user.

Yosemite crash

There have been reports of an intermittent crash of the Mac OS while debugging an MPI program with TotalView on the first release of Yosemite. Crash reports have been sent to Apple as this is symptomatic of a kernel bug.

Linux

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 RHEL 5 (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." will be 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 RHEL 6.

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.

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 slower.

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

Linux - Ubuntu

ptrace_scope setting causes various failures with TotalView and MemoryScape

Beginning with Ubuntu 10.10, the kernel places a security restriction that prevents attaching to a process that was not directly started by the current process. This security restriction prevents TotalView from debugging applications that are not directly started by the debugger, which would include most MPI jobs or currently running processes. You can turn off this feature by entering:

- su root // a simple sudo of the following does not seem to work
- echo 0 > /proc/sys/kernel/yama/ptrace_scope

This will enable debugging and attaching to processes. This setting will last until the system is rebooted. For a permanent change, change the kernel.yama.ptrace_scope setting on the last line in /etc/sysctl.d/10-ptrace.conf.

Memory debugging by linking against the TotalView libraries may not work

There are a number of cases where 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, one must 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

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. They work based on the assumption that the debug type information can be spread out over multiple compilation units. For instance, Clang will not emit type definitions for types that are not needed by a module and could be replaced with a forward declaration. Further, Clang will only emit 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 will never emit type information for types that are not referenced at all by the program.

-fstandalone-debug is the default on Mac OS X.

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

CUDA

Dynamic parallelism not fully supported

With CUDA 5.5-7.0 we have limited support for dynamic parallelism. You should be able to use TotalView in the CUDA 5.5/CUDA 6.5 runtime with applications that display dynamic parallelism; however 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.

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.

Debugging forked processes on Solaris 11 OS

If the process calls a fork, and the process is being debugged under TotalView, sometimes the fork system call may silently fail. As a result TotalView will continue debugging the parent process only.