TotalView 2022.4 Release Notes

Updated: November 2, 2022

TotalView by Perforce

Table of Contents

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 at: https://help.totalview.io/.

---

1 Additions and Updates

AMD GPU Debugging Support

TotalView 2022.4 adds support for debugging AMD GPUs in addition to its ability to debug NVIDIA GPUs. TotalView supports versions 5.1 to 5.2 of the ROCm software stack for GPU programming, including support for debugging HIP (Heterogeneous Interface for Portability) and MPI code running on AMD MI50, MI100, and MI200 series of GPUs. TotalView supports the following debugging features on AMD GPUs:

- Process launch, attach, and detach
- Viewing scalar, vector, general, and special AMD GPU registers
- Instruction disassembly
- Breakpoint creation and deletion on AMD GPU code
- Single-stepping and fast smart-stepping
- Stack unwinding, including inlined functions
- Navigation controls for changing the logical workgroup / work-item focus or physical agent, queue, dispatch, wave, and lane focus
- Variable display with the ROCm 5.1+ compilers
- Data watchpoints on global memory variables

Preparing your program for debugging is simple. Use the "-O0 -g" flags with the ROCm 5.1+ compilers to generate the necessary debug symbols for TotalView.

To debug a program on AMD GPUs, use the "-rocm" flag when you start TotalView, for example:

```
totalview -rocm a.out
```

Once started, simply set breakpoints in the HIP kernel code by selecting the line number where you would like to stop, then begin running your program. TotalView will automatically plant breakpoints in the HIP code once it is loaded onto the GPU. Use TotalView's stepping commands to control execution, and dive on variables to view their values in your GPU code. Reverse debugging will not be supported for code running on the AMD GPUs but is planned to work with reverse debug code running on the CPU in a combined CPU/GPU debugging session. There currently is an issue while reverse debugging when the -rocm switch is used and debugging on the GPU. We are working to resolve this issue for an upcoming release of TotalView.

To learn more about specific AMD GPU debugging characteristics, see help for the `drocm` command in TotalView's Command Line Interface (`dhelp drocm`).

Array View for Array Debugging and Visualization

A new Array View is available in TotalView 2022.4 which enables focused debugging of array data. Through the Array View, you can examine statistical information about an array and visualize array data.

To bring up the Array View, either activate it from the Window > Views > Array View menu item or select Add to Array View from the context menu in the Local Variables view or Data View when selecting an array variable.

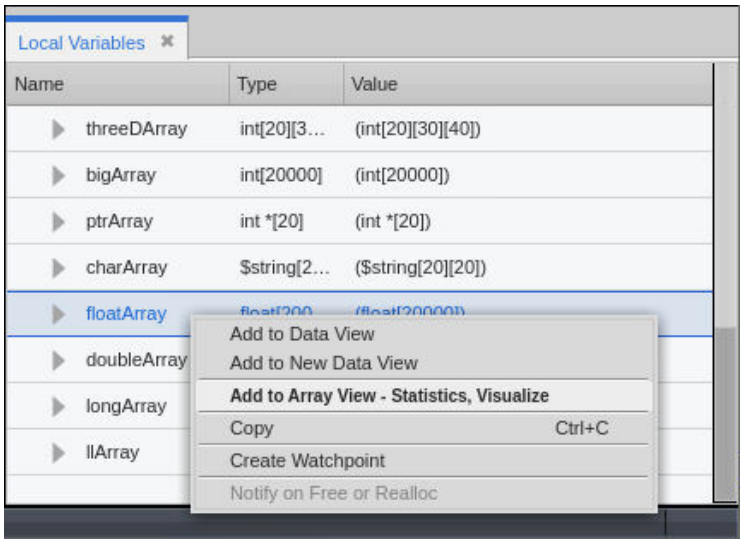
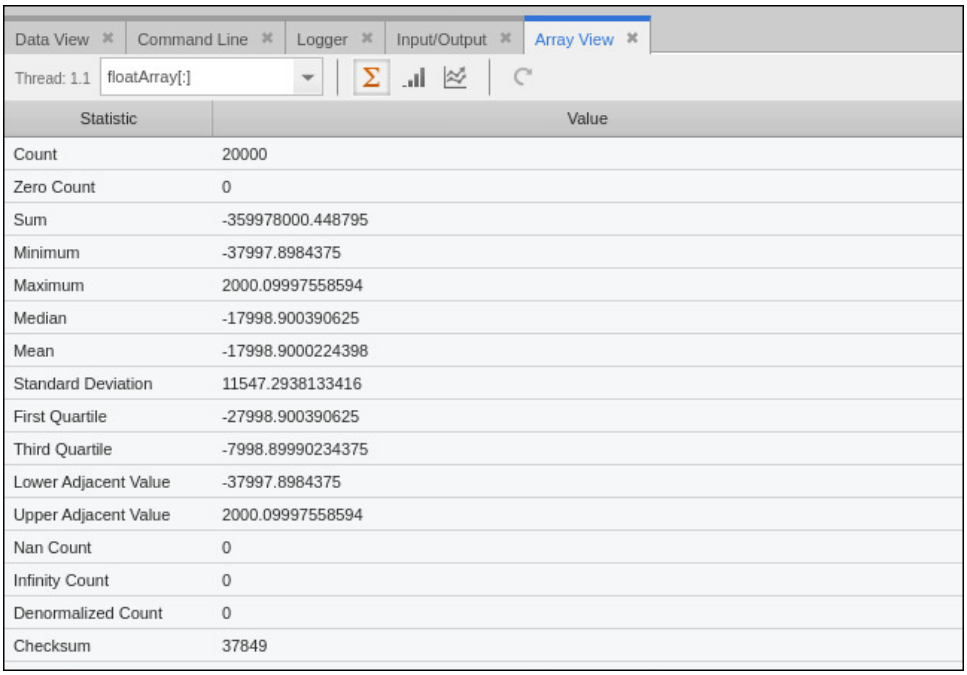


Figure 1 - Add to Array View menu item

The toolbar area of the Array View displays the process and thread for the array data being displayed. The array expression is shown along with the slice of the array to debug. In Figure 2, array expression *floatArray[:]* is shown and the *[:]* specifies to debug the whole array. A portion of the array can be specified by providing the beginning and ending indices, such as *[10:15]*.

The Array View displays array statistics information initially. In previous versions of TotalView this was separate functionality available in the Array Statistics View.



Statistic	Value
Count	20000
Zero Count	0
Sum	-359978000.448795
Minimum	-37997.8984375
Maximum	2000.09997558594
Median	-17998.900390625
Mean	-17998.9000224398
Standard Deviation	11547.2938133416
First Quartile	-27998.900390625
Third Quartile	-7998.89990234375
Lower Adjacent Value	-37997.8984375
Upper Adjacent Value	2000.09997558594
Nan Count	0
Infinity Count	0
Denormalized Count	0
Checksum	37849

Figure 2 - Array Statistics

A histogram plot of the array data is shown when the bar plot icon is clicked in the toolbar. Adjust the number of bins used in the plot using the field in the plot.

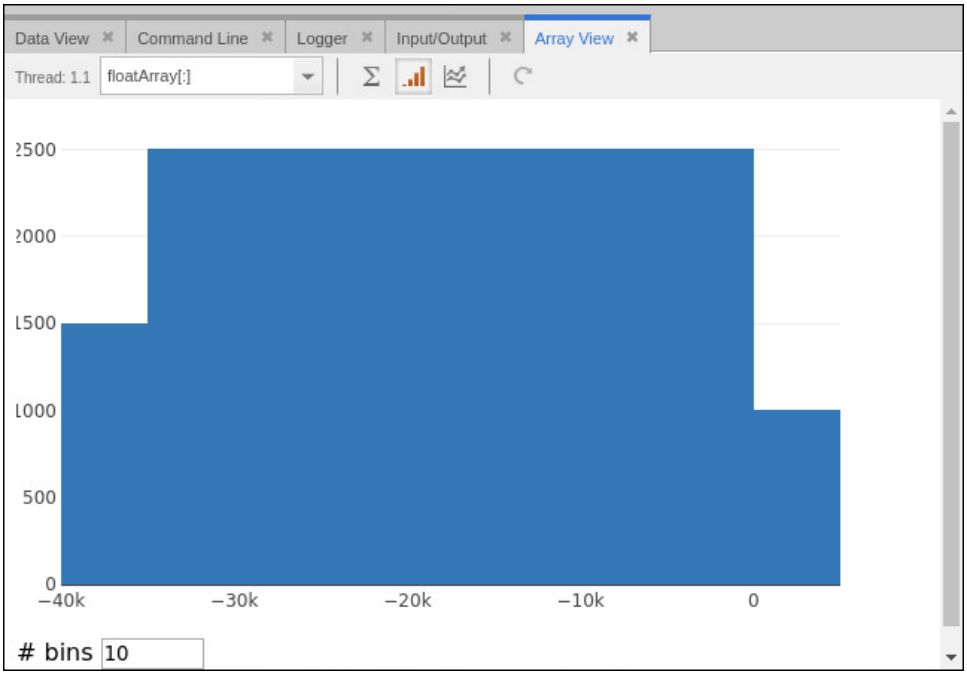


Figure 3 - Visualize Array in Histogram Plot

A line plot of the array data is also available when the line plot icon is selected in the toolbar.

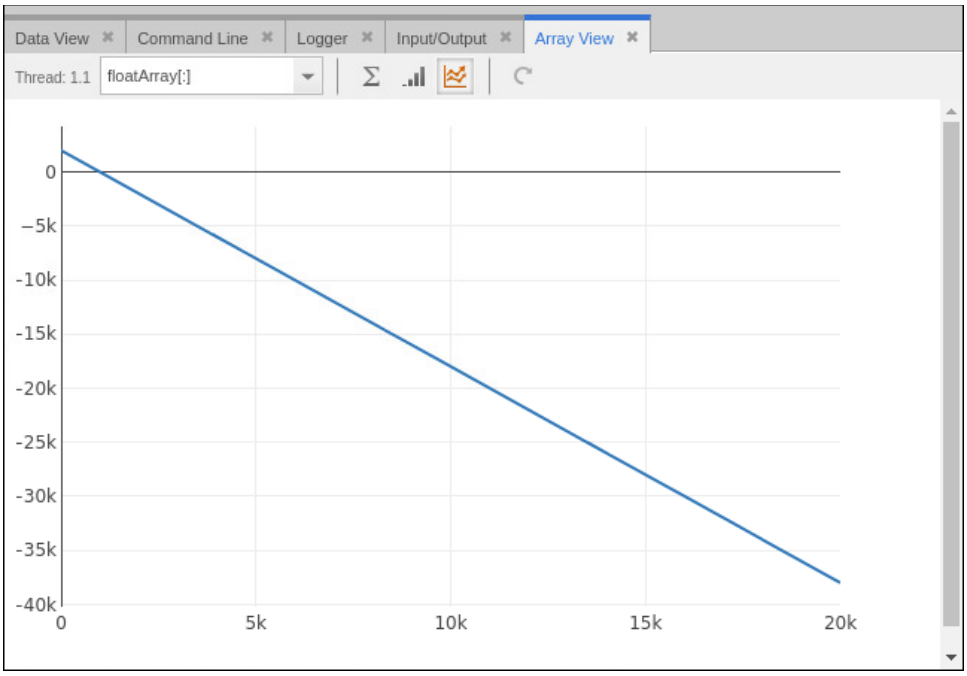


Figure 4 - Visualize Array in Line Plot

Watch for more array debugging capabilities to be added to the Array View in coming versions of TotalView!

TotalView Startup Performance Enhancements

For TotalView 2022.4, further startup performance improvements optimize how the debugger looks up symbols and entry point values in the runtime linker. This can have a big impact on debugger startup performance for applications built with many shared libraries.

> **DEBUGGING TIP:** A further startup performance optimization customers should consider using is to instruct TotalView to defer processing of shared libraries until later in the debug session. By adding the TotalView command line option -*no_dlopen_always_recalculate* the debugger will delay processing shared library symbols until they are needed later in the debug session. Learn more about tuning TotalView's parallel debugging performance in the "Scalability in HPC Computing

Environments" section of the *TotalView User Guide,* Part III Parallel Debugging chapter.

TotalView Base Compilation System and Dependency Changes

TotalView 2022.4 changes the base compilation for the following systems:

Linux x86-64

- Base build system was changed from CentOS 7.0 to CentOS 7.9, resulting in a base OS of CentOS/RHEL 7.9 and above. Ubuntu 18.04 and 20.04 are also supported.

  Note that TotalView's user interface requires libxkbcommon and libXcomposite libraries, typically installed as part of a desktop graphical environment.

Linux PowerLE

- Base build system was changed from Ubuntu 14.04 to CentOS 7.5. The new base supported system changes to CentOS/RHEL 7.5 and above. Ubuntu 18.04 and 20.04 are also supported.

  Note that TotalView's user interface requires libxkbcommon and libXcomposite libraries, typically installed as part of a desktop graphical environment.

macOS

- Base build system was changed from macOS High Sierra to macOS Big Sur. TotalView's base compilation support stays the same with support for macOS Big Sur and Monterey.

Bug fixes and performance improvements

Numerous bug fixes and performance improvements have been addressed, including performance improvements for data transformations.

2 Platform Updates

There are no platform updates in the TotalView 2022.4 release.

3 Bug Fixes for TotalView 2022.4

TVT-36965 TotalView is slow starting up on 10000 processes, 79 nodes
TVT-37067 TotalView not correctly displaying the value of a bit field

4 Deprecation Notices

Linux x86 32-bit

Starting with the first release of 2023, TotalView will no longer provide a 32-bit only Linux x86 version of the debugger. TotalView will continue to support debugging 32-bit executables from a Linux x86 64-bit system.

Base TotalView Linux x86-64 Support Change

Starting with TotalView 2022.4, TotalView will be built on a CentOS 7.9 build system instead of CentOS 7.0.

Cross Debugging from Linux x86-64 to Linux Power and Linux Cell

Starting with TotalView 2022.4, TotalView will no longer support cross debugging from Linux x86-64 to Linux Power and Linux Cell architectures.

Linux Power

We are planning to drop support for Linux Power with the first TotalView release in 2023. Please notify us at techsupport@roguewave.com if you have concerns about this.

AIX

We are planning to drop support for AIX starting with the first TotalView release in 2024. The final 2023 release with AIX support will continue to be made available. Please notify us at techsupport@roguewave.com if you have concerns about this.

Solaris

We are planning to drop support for Solaris starting with the first TotalView release in 2024. The final 2023 release with Solaris support will continue to be made available. Please notify us at techsupport@roguewave.com if you have concerns about this.

5 Known Issues

Microsoft Windows Subsystem for Linux (WSL/WSL2)

TotalView does not yet support Microsoft WSL/WSL2. Some preliminary testing has been done and there are networking issues that need to be resolved. If you are interested in TotalView supporting WSL please send a request to techsupport@roguewave.com.

Python Debugging

Anaconda

TotalView supports debugging of the python interpreter in release 4 of Anaconda but this functionality does not work with the recent release of Anaconda 5. A change in the way it builds the python interpreter has broken the ability to debug python in TotalView for the python Anaconda 5 distribution.

Licensing

TotalView releases built with FlexNet Publisher 11.16.6 must have licenses served by a license server at 11.16.6 or higher

FlexNet Publisher client library version 11.16.6 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 v11.16.6 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 v11.13.1 from previous TotalView releases. In this case the vendor daemon's debug log file shows "(toolworks) Request denied: Client (11.16) newer than Vendor Daemon (11.13). (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@perforce.com to set up a new license file for the FlexNet Embedded license server.

macOS

Trace Error when Debugging MPI Applications on Monterey

When debugging an MPI application on Monterey TotalView may fail with a fatal "trace_attach" error as it tries to attach to the MPI processes.

Memory Debugging on macOS Big Sur and Monterey

MacOS Big Sur and Monterey changed how system-provided libraries are provided and this is impacting TotalView's ability to track memory allocations and deallocations with its memory debugging technology. The development team is looking into the issue and will provide a fix as soon as possible.

Big Sur Dynamic Linker Cache

MacOS Big Sur ships with a built-in dynamic linker cache for all system-provided libraries. This is a change from previous versions of macOS. As part of this change, copies of dynamic libraries are no longer present on the filesystem. As the result, TotalView debugger will fail to construct full backtraces that extend into the dynamic system libraries.

Physical console access needed when running TotalView on macOS High Sierra

Due to new security changes in macOS High Sierra and higher, TotalView will only run from the console and cannot be run through a remote desktop technology such as VNC.

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 Later using 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.

Linux

Reverse Debugging Crash on Ubuntu 22.04 / CentOS 9

Newer Linux distributions such as Ubuntu 22.04 and CentOS 9 changed restartable sequences structures and could cause a crash when reverse debugging. The rseq (restartable sequence) system call is not supported. This includes attaching to running programs that previously issued this system call. glibc, from version 2.35 onwards, attempts to register a restartable sequence with the kernel, preventing the reverse debugging engine from attaching to running processes using those versions of glibc. This glibc behavior can be disabled by adding glibc.pthread.rseq=0 to the GLIBC_TUNABLES environment variable of the target application. To avoid the crash simple set the environment variable to 0:

```
export GLIBC_TUNABLES=glibc.pthread.rseq=0
```

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, **Error! Unknown document property name.** 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

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 techsupport@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.

Cray

Debugging your program within supercomputer environments can often be challenging. Reference the sections below to learn pointers on how to successfully run TotalView on Cray EX (Shasta) systems, enable memory debugging and perform reverse debugging on your program within a Cray environment.

Running TotalView on a Cray EX (Shasta) system

TotalView has been tested on several Cray EX systems with different configurations. The following are recommendations for running TotalView on several specific system configurations. If your system does not match these configurations, please contact the TotalView support team at techsupport@roguewave.com to have them help guide you to the proper configuration to use. If you are using NVIDIA GPUs on your Cray EX system, please see the "Cray EX Configured with NVIDIA GPUs" below.

Cray EX Configured with: Cray Shasta / PALS

The Cray Shasta environment with the PALS workload manager contains several issues that have been worked around in TotalView, primarily by disabling the use of the Cray Common Tools Interface (CTI) library. When CTI is disabled, TotalView will use its traditional MRNet/SSH based tree instantiation instead of an MRNet/CTI tree.

More seriously is a critical problem with PALS where it has been observed that "mpiexec.pals" fails when trying to debug multiple processes per node. In this case, the "mpiexec.pals" program currently fails and prints the error message: "Error: RPC timeout after 15s, received 0 of 1 responses". This bug has been recorded by HPE as PE-37769 and preliminary investigation shows that there is a bug in the PMI layer. Only the first process on a node calls the barrier. PMI needs a change to make all processes on a node wait at the barrier.

The following are the recommended requirements for successfully launching TotalView on a Cray EX system configured with Cray Shasta / PALS. Example launch commands follow the requirements.

1. The Cray Common Tools Interface (CTI) library and Parallel Application Launching Service (PALS) packages that are installed on the system have several bugs that prevent the TotalView MRNet/CTI support from working. Perforce has modified TotalView to work around these bugs by avoiding the use of MRNet/CTI, and instead forcing the use of a traditional MRNet/SSH launch approach. Until the CTI bugs are resolved, set the TVD_DISABLE_CRAY=1 environment variable before starting TotalView.
2. TotalView's MRNet/SSH tree instantiation requires the following:
    a. SSH from the front-end (login) node to the compute nodes, and from the compute nodes to other compute nodes. If SSH is not allow or not configured correctly, MRNet/SSH tree instantiation will fail.
    b. Each user must configure SSH to allow password-less SSH. That is, it must be possible for the user to SSH from one node to another without a password. There are numerous articles on the Internet that describe how to do this (for example, https://linuxize.com/post/how-to-setup-passwordless-ssh-login/).
    c. SSH must be configured to disable strict host key checking. TotalView is configured to do this automatically when run in the MRNet/SSH mode by setting the TVD_DISABLE_CRAY=1 environment variable. The "totalview"

startup script does this by passing the following configuration options to the TotalView executables:

```
-xplat_rsh ssh
-xplat_rsh_args '-x -o StrictHostKeyChecking=no'
```

3. TotalView requires the MPI starter program to implement the MPIR Process Acquisition Interface, which allows it to attach to a parallel program. On Cray Shasta/PALS system, the "mpiexec.pals" command must be used to launch jobs, because the "cray mpiexec" command does not support MPIR. The options for the "mpiexec.pals" program are similar to "cray mpiexec", so generally it is possible to use "mpiexec.pals" in place of "cray mpiexec". Use "mpiexec.pals --help" for more information on "mpiexec.pals".
4. The "--no-transfer" option should be provided to "mpiexec.pals", which disables transferring application binaries to the compute nodes. Using this option avoids two issues when debugging:
    a. Caching the application binaries. The debugger will use the original copies of the application binaries rather than copying them from a compute node into the TotalView library cache, which saves time and space.
    b. Complains from the debugger regarding the system deleting the transferred copies of the application binaries: "We can no longer read the executable '/run/palsd/…'".
5. The "mpiexec.pals" program has a bug where it will print the following message:

    ```
    Usage: cray pals apps signal create [OPTIONS] APID
    Try 'cray pals apps signal create --help' for help.

    Error: Error received from server: 503 Service Unavailable
    ```

    There appears to be no consequences to the error message. The command was trying to signal the ranks with a SIGCONT but, it is unnecessary since when running under the debugger it just stops the process again.
6. The "mpiexec.pals" program currently fails when trying to debug multiple processes per node. Use the "-ppn 1" option to restrict the job to use one process per node. When attempting to debug multiple processes per node, mpiexec,pals will print the error message: "**Error: RPC timeout after 15s, received 0 of 1 responses**".

Here is an example of some commands to allocate 4 nodes, run an MPI application with one process per node on 4 nodes, and run that application under TotalView:

```
# Set TVD_DISABLE_CRAY=1 to disable use of Cray CTI launch
export TVD_DISABLE_CRAY=1

# Test that a normal launch on these nodes works
mpiexec.pals -n 4 -ppn 1 ./mpi_hello_world

# Launch the job under TotalView, disable transferring binaries
# to compute nodes, and run one process per node
totalview -args mpiexec.pals --no-transfer --verbose \
     -n 4 -ppn 1 ./mpi_hello_world
```

Cray EX Configured with: Cray Shasta / SLURM

The Cray Shasta environment with the SLURM workload manager contains an issue that can be worked around in TotalView by disabling the use of the Cray Common Tools Interface (CTI) library.

The following are the recommended requirements for successfully launching TotalView on a Cray EX system configured with Cray Shasta / SLURM. Example launch commands follow the requirements.

1. The Cray Common Tools Interface (CTI) library package that is installed on the system has bugs that prevent the TotalView MRNet/CTI support from working. Perforce has modified TotalView to work around these bugs by avoiding the use of MRNet/CTI, and instead forcing the use of a traditional MRNet/SSH launch approach. Until the CTI bugs are resolved, set the TVD_DISABLE_CRAY=1 environment variable before starting TotalView.
2. TotalView's MRNet/SSH tree instantiation requires the following:
    a. SSH from the front-end (login) node to the compute nodes, and from the compute nodes to other compute nodes. If SSH is not allow or not configured correctly, MRNet/SSH tree instantiation will fail.
    b. Each user must configure SSH to allow password-less SSH. That is, it must be possible for the user to SSH from one node to another without a password. There are numerous articles on the Internet that describe how to do this (for example, https://linuxize.com/post/how-to-setup-passwordless-ssh-login/).
    c. SSH must be configured to disable strict host key checking. TotalView is configured to do this automatically when run in the MRNet/SSH mode by

setting the TVD_DISABLE_CRAY=1 environment variable. It does this by setting the following internal configuration options through the TotalView CLI commands:

```
-xplat_rsh ssh
-xplat_rsh_args '-x -o StrictHostKeyChecking=no'
```

Here is an example of some commands to allocate 4 nodes, run an MPI application with one process per node on 4 nodes, and run that application under TotalView:

```
# Set TVD_DISABLE_CRAY=1 to disable use of Cray CTI launch
export TVD_DISABLE_CRAY=1

# Allocate a node for the job
salloc -C gpu -N 1 -G 4 -t 30 -q regular -A nvendor_g

# Test that a normal launch on these nodes works
srun -n 4 ./mpi_hello_world

# Launch the job under TotalView
totalview -disable_cray -mrnet_super_bushy \
   -args srun -n 4 ./mpi_hello_world

# If running a job that uses CUDA, launch the job under |
# TotalView without CUDA.
totalview -disable_cray -no_cuda \
   -args srun -n 4 ./mpi_hello_world
```

Cray EX Configured with NVIDIA GPUs

We have not fully tested debugging NVIDIA GPUs on Cray EX systems, therefore it is not yet supported and may not work. We have seen application hangs with one of our CUDA test programs, so we have reason to believe that there is a bug in TotalView, the CUDA debug API, or the NVIDIA kernel driver on these systems. Further investigation is needed. We'd appreciate feedback on your experiences debugging GPU code on Cray EX systems. Note the following:

a. Due to limitations in the CUDA debug API, is not possible for a single debugger process to debug more than one CUDA process per node. It is possible to work around this limitation in MPI applications by using the TotalView "-mrnet_super_bushy" option. This option will force TotalView to create an MRNet tree that consists of one debugger server process per MPI process.

b. To disable CUDA debugging in TotalView, use the "-no_cuda" option. This option might avoid CUDA-related application or debugger hangs; however you will not be able to debug code running on the GPUs.

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.
  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.
  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.
  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

With the 2021.3 release of TotalView, when trying to enable reverse debugging on a process the error message "Couldn't attach to at least one group member" is raised. This issue will be addressed in the next version of TotalView.

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.

AIX

TotalView Help Does Not Display with Firefox on AIX

Firefox on AIX is unable to display the TotalView help due to an error in the processing of the help set. A workaround is to view the TotalView help set online at https://help.totalview.io with another browser.