TotalView

# TotalView Change Log

Versions 8.0 to 2024.1

PERFORCE

www.perforce.com

## ACKNOWLEDGMENTS



TotalView by Perforce
http://totalview.io

# Contents

## TotalView Change Log, 8.0 - 8.15.10 and 2016.01-2024.1

# TotalView Change Log, 8.0 - 8.15.10 and 2016.01-2024.1

This document lists updates made to the TotalView product for versions 8.0.0 through Version 8.14, and 2016.01 to 2024.1, including MemoryScape and ReplayEngine. For specifics about the current release, please see the document "TotalView_Change_Log.pdf" in the PDF directory of your installation, or the "TotalView Change Log" on the TotalView documentation site.

Note that releases after 8.15.10 (October, 2015) use a different version numbering pattern:

*<year>.<release-number>*

For example, the first release in 2024 is 2024.1.

# TotalView

## TotalView 2024.1

### *TotalView Remote Client Support for Windows*

Now, run the full TotalView user interface to perform remote debugging from Windows-based systems to back-end systems supported by TotalView. This builds on the existing macOS and Linux x86-64 based-support for remote debugging.

### *Reprise License Technology*

This release changes to the Reprise License Manager (RLM) technology instead of Flexera-based technology. All new and renewed TotalView licenses will be issued as RLM licenses.

Flexnet Publisher (FNP) and FlexNet Embedded (FNE) license technology is still part of TotalView, and existing FNP and FNE licenses will continue to work, although FNP / FNE licenses will be phased out and removed in future releases. See the TotalView Installation and Licensing Guide for full details about RLM and how to transition from FNP / FNE.

### *FlexNet Embedded License Server Upgrade*

The FlexNet Embedded (FNE) license has been upgraded to 2022.12.0-0 from FNE 2020.12. This avoids the well-known log4j security vulnerability present in the FlexNet License Server Manager (FLSM) component in FNE 2020.12. While FLSM was present in past TotalView license server distributions, it was not documented and there are no reports of its use; however, this upgrade avoids any potential security vulnerabilities. For more information, see the "Known Issues, Licensing" section in the TotalView Release Notes.

### *Bug Fixes and Performance Improvements*

Numerous bug fixes and performance improvements have been addressed.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Assembler View**

  Take advantage of improved support for inline functions, GPU level assembly display, and display of function labels.

### *Platform Updates*

- **Operating Systems**

  macOS Sonoma

  **ROCm**

  ROCm 6.0

## TotalView 2023.4

### *Bug Fixes and Performance Improvements*

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

### *New License Technology and Discovery*

The Reprise License Manager (RLM) replaces both FlexNet Publisher (FNP) and FlexNet Embedded (FNE) licenses for new or renewing licenses. Existing customers using FNP or FNE will migrate to RLM beginning in 2024.

TotalView now discovers locally installed license files in a more streamlined way, as the installation creates a single license directory rather than directories specific to the type of license.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Registers View**

  Examine CPU and GPU registers for target programs. Add them to the Data View to edit their value or cast to a different type.

- **Assembly View Early Access**

  Debug low-level machine instructions using the new Assembly view. Instruction-level stepping commands and breakpoints provide precise, fine-grained control.

- **C++ Type Transformations**

  Examine the data pointed to by an iterator to ease the debugging of STL collection data. See the Release Notes for a list of specific list of supported STL collection classes.

### *Platform Updates*

- **Operating Systems**

  Fedora 37 and 38

- **Compilers**

  GCC 13

- **ROCm**

  ROCm 5.7

- **CUDA**

  CUDA 12

## TotalView 2023.3

### *Bug Fixes and Performance Improvements*

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

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Array Slicing and Striding**

  Use the Array View's new Array Configuration dialog to slice an array and add an optional stride to isolate a portion of the array for analysis. You can also change the array's datatype in this dialog.

- **Memory Buffer Overwrite Detection Report**

  Generate an on-demand Corrupt Guard Block Report to see overwritten blocks of memory.

- **Apple ARM-based Architectures**

  Debug on Apple's ARM-based M1 and M2 architectures.

- **Python 3.11 Support**

  Debug mixed C/C++, Python 3.11 programs.

*Platform Updates*

- **Operating Systems**

  macOS Monterey and Ventura (ARM64 M1/M2)

- **Compilers**

  Intel oneAPI 2023, GCC 12

- **ROCm**

  ROCm 5.6

# TotalView 2023.2

*Bug fixes and performance improvements*

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

*The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Visualize Array Data with 3D Surface Plots**

  You can now visualize array data with a new 3D surface plot, useful to help discover anomalies in data value range, numerical spikes, and NaNs. This feature is available after adding a data element to the Array View and selecting the Surface Plot button. See the Release Notes for more detail on how to use this feature.

- **TotalView UI Share Group Target Support**

  The UI has added the ability to target a share group for various debugger operations. As TotalView acquires processes during a debugging session, it organizes them into a control group and share groups, placing processes that share the same executable image into a common share group. This allows you to advance just the process in a specific share group, particularly useful when debugging MPI Multi-Program/Multi-Data (MPMD) style applications.

- **Memory Buffer Overwrite Detection**

  Memory buffer overwrite protection is implemented via guard blocks that can help you detect when an allocated memory block has been overwritten beyond the ends of its bounds. If these blocks are not overwritten when freed, TotalView will raise a memory event.

- **Memory Block Hoarding**

  When hoarding memory, TotalView doesn't immediately release memory that has been deallocated; instead, it records that the block was deallocated, but does not ask the operating system to free it. Your program can continue accessing that memory without it being reallocated. This can help uncover use after deallocation errors.

  Platform Updates

  - **Operating Systems**

    AIX 7.3

    Rocky Linux

    macOS Ventura (Intel)

  - **Compilers**

    IBM Open XL C/C++ 17.1

  - **ROCm**

    ROCm 5.5

# TotalView 2023.1

### *AMD GPU ROCm Debugging*

This release adds support for ROCm 5.4, as well as automatic discovery for target programs using the ROCm toolkit. As a result, the **-rocm** command line switch is no longer needed to debug ROCm programs.

### *CUDA 11.8*

TotalView now supports CUDA 11.8, which introduces a new, unified CUDA debugger backend; however, some serious problems were discovered during testing which won't be addressed until the CUDA 12.0 r525 driver is released.

For this release of TotalView, to use CUDA 11.8, you must force CUDA to fall back to the legacy CUDA debug implementation by setting the `CUDBG_USE_LEGACY_DEBUGGER` environment variable to `1`:

```
export CUDBG_USE_LEGACY_DEBUGGER=1
```

Because every debugger *and* application process in a debug session must use this setting, the easiest solution is to set this environment variable in your shell startup file (e.g., **.cshrc** or **.bashrc**).

See the Release Notes for more detail.

### *Python 3.10 Mixed Language Debugging Support*

Support for Python 3.10 has been added, when performing mixed language C/C++ and Python debugging.

### *AIX 7.2 Support*

Support has been added for AIX 7.2; however, targets launched through a fork/exec may hang in low-level ptrace() system calls, due to a kernel bug in the underlying AIX OS. IBM is planning to release a software fix for this issue.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Performance Enhancements**

  Optimization to user interface updates and debug event processing now results in faster debugging session launches, especially for large scale parallel jobs. These changes also improve UI updates during debugging operations, such as when quickly stepping through multiple lines of source code.

- **Memory Painting**

  The new memory painting feature paints memory to find memory access problems for uninitialized or freed memory. For both newly allocated memory blocks and freed memory blocks, a hexadecimal paint pattern is written into the memory block, providing a consistent memory value for cases in which your program tries to read from those memory blocks. See the Release Notes for more detail.

- **TotalView Working Directory**

  Specify a working directory where TotalView will execute your target program. If not provided, the default is the directory from which you invoked TotalView.

  Enter this value into the UI or on the command line when starting TotalView. Modify it at during a debug session using the **Process > Modify Arguments** menu or in the Sessions Manager.

## TotalView 2022.4

### *AMD GPU Debugging*

This release adds support for debugging AMD GPUs, expanding TotalView's GPU debugging capabilities beyond NVIDIA GPUs. This includes support for versions 5.1 and 5.2 of the ROCm software stack for GPU programming, and support for debugging HIP (Heterogeneous Interface for Portability) and MPI code running on AMD MI50, MI100, and MI200 series of GPUs. See the Release Notes for more detail.

### *Startup Performance Enhancements*

Improved performance for runtime linking, particularly relevant when loading many shared libraries.

### *Minimum System Requirements*

Minimum requirements for running TotalView have been updated for some platforms:

- **Linux x86-64**:
  - **CentOS**: CentOS 7.9/RHEL 7.9, upgraded from CentOS 7.0

    **Ubuntu**: 18.04, from 16.04
  - **macOS**: Big Sur, from High Sierra
- **Linux PowerLE**
  - **CentOS**: CentOS 7.5/RHEL 7.5, upgraded from CentOS 7.0

    **Ubuntu**: 18.04, from 16.04

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **New Array View**

  The new Array View enables focused debugging of array data. This new view has been combined with the existing Array Statistics View to provide broader support for visualizing and debugging arrays. See the Release Notes for more detail.

## TotalView 2022.3

### *AMD GPU Debugging*

This release expands TotalView's GPU debugging capabilities, with preliminary support for debugging AMD GPUs.This includes support for versions 5.1 and 5.2 of the ROCm software stack for GPU programming, and support for debugging HIP (Heterogeneous Interface for Portability) and MPI code running on AMD MI50, MI100, and MI200 series of GPUs. See the Release Notes for more detail.

### *NVIDIA OpenACC*

This release adds official support for debugging NVIDIA OpenACC compiled applications.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Dive Stack Expansion State**

  The Data View's dive stack functionality has been enhanced to maintain the expansion and scroll state so you can traverse up and down the stack and observe the state at any point.

## TotalView 2022.2

### *dprint Command Timeout Option*

The CLI **dprint** command has a new timeout option (**`dprint -timeout`** *`nSeconds`*) to define the maximum number of seconds the command should run. See the CLI Commands section in the *TotalView Reference Guide* for more information.

### *Verify Shared Libraries Using Build IDs for Better Performance*

When shared libraries are loaded during a parallel job across a node cluster, the library's build ID, when available, is now used to verify each library, rather than checksumming, providing much improved processing performance. This behavior is controlled by new variable **TV::check_unique_id** in concert with an existing variable **TV::checksum_libraries**.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Data Debugging Improvements Using Dive Stacks**

  The Data View now includes support for navigating a dive stack of expressions when diving through data structures in the Data View. Navigating up and down the stack allows easy traversal through the data structure, streamlining workflows to focus on your data.

### *Platform Updates*

- **Operating systems**:

  - Fedora 35
  - Ubuntu 22.04

  Both Fedora 35 and Ubuntu 22.04 incorporate numerous system level changes that impact debuggers. This release provides preliminary support for these two operating systems, including incorporates various enhancements and fixes to support debugging on these platforms. Additional testing should result in full support in the next 2022.3 release.

- **Compilers**:
  - Intel oneAPI 2022
- **CUDA**:
  - CUDA 11.6 and 11.7
- **zLib**:

  The compression utility zLib has been upgraded to the latest version to eliminate CVE security vulnerabilities in the previous version.

# TotalView 2022.1

***macOS Monterey Support***

This release adds support for Apple's latest version of macOS 12, Monterey.

***The New UI TotalView User Interface***

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Memory Debugging Block Notify**

  This release adds the ability to watch a specific allocated memory block and raise an event when it gets freed or deleted. This is useful when a specific block of memory is freed, but your program accesses the memory later through a dangling pointer reference. At this point, the memory may or may not still contain valid data and typically results in sporadic crashes or data problems in your program.

  This feature is not yet covered in the product documentation, but detailed notes regarding its use are in the 2022.1 Release Notes.

### *Platform Updates*

- **Compilers**:
  - GCC 11
- **Operating systems**:
  - macOS Monterey (12)
  - Fedora 34
- **CUDA**:
  - CUDA 11.4 and 11.5

## TotalView 2021.4

### *Cray EX (Shasta) Support*

This release adds support for the new Cray EX (Shasta) platform. Cray EX systems are extremely powerful and highly configurable. TotalView must be configured properly to successfully debug a parallel job on Cray EX systems. For details, see the Cray-specific section under "Known Issues" in the TotalView release notes, available at https://help.totalview.io.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **GPU Status View**

  Analyze your code running across one or more GPUs with the new GPU Status view which aggregates and filters attributes to display how CUDA code is running on one or more GPUs within a node and across a cluster. The view automatically opens when TotalView recognizes a kernel loaded on the device and a CUDA context thread is created.

- **CUDA-MEMCHECK**

  The new UI adds support for CUDA-MEMCHECK which allows developers to easily identify the source and cause of memory access errors with code running on the GPU. Turn on CUDA-MEMCHECK by toggling the option on the **Program Session** dialog or from the **Debug > Enable CUDA Memcheck** menu option.

- **Process Hold support added**

  Hold a process to prevent it from running, useful when you need to run only a subset of processes or to run only specific processes held at a barrierpoint.

  To hold a process, select **Process > Hold**. Any commands to resume execution will be ignored until the process is unheld. The held state for the process in focus is indicated in the window title and status bar. The ability to hold threads and groups of processes will be added in subsequent TotalView releases.

- **Array Debugging and Visualization (Early Access)**

  Available as an early access feature, the Array Tool provides a central location for performing array debugging operations, such as viewing array statistics and visualizing array data. This version does not yet include the overall, planned functionality. To activate the Array Tool, enable the option in the Labs panel at **File > Preferences > Labs**. Please send an email to tv-beta@perforce.com with feedback regarding array debugging capabilities that you might find useful. For detail on using the Array Tool, see "Early Access: Array Debugging and Visualization with the Array Tool" in the TotalView release notes, available at https://help.totalview.io.

# TotalView 2021.3

### *Reverse debugging*

Enhancements to reverse debugging include better support of handling signals while running backwards and properly focusing on the thread that hits a breakpoint while executing in reverse.

### *TotalView Student Edition Installations*

The installer for the Student Edition of TotalView has been updated to provide a native installation experience, improving the installation and application behavior of TotalView on macOS.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Memory debugging event reports**

  Receive event notifications for various memory problems in your code, including memory freed twice, memory API failures, non-heap memory being freed, and other errors.

- **Upgraded Python support**

  Mixed language debugging C/C++ with Python now supports the latest Python version, encompassing 3.5 to 3.9, as well as Python 2.7.

- **GPU Status view (Early Access)**

  Analyze your code running across one or more GPUs with the new GPU Status view which aggregates and filters attributes to display how CUDA code is running on one or more GPUs within a node and across a cluster. Enable it by choosing Preferences > Labs, then restarting TotalView's new UI.

# TotalView 2021.2

### *NVIDIA A100 Ampere GPU and MIG*

Support for NVIDIA's latest A100 Ampere GPUs including Multi-Instance GPU (MIG).

### *NVIDIA CUDA 11.3*

Support for NVIDIA's latest version of CUDA, 11.3.

### *Record and replay the execution of your software with LiveRecorder*

Use LiveRecorder (Undo's Software Failure Replay platform) to load LiveRecorder-generated recording files. LiveRecorder greatly accelerates software defect reproduction and resolution by recording the execution of your application (outside of the debugger) as it fails or behaves unexpectedly. With LiveRecorder, software failures are captured "in the act" during the testing and DevOps stages and recorded to a file. Use TotalView to load the LiveRecorder file and replay the recording to get instant visibility into what your program did and why. Contact us to learn how to combine TotalView and LiveRecorder to experience 100% failure reproducibility, resolve defects faster, and accelerate software delivery. LiveRecorder combined with TotalView will enable you to maintain quality at velocity, troubleshoot customer issues faster, and reduce internal stress through a predictable, reproducible workflow.

### *Installation guide and license server enhancements*

The ***TotalView Installation Guide*** has been reorganized and revised, now presenting the installation and licensing instructions in a streamlined, product-oriented approach. In addition, the TotalView FlexNet Publisher license server is now distributed in its own installation bundle rather than with the TotalView application bundle to make it easier for administrators to install the license server and developers to install the TotalView application.

### *The New UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Memory debugging heap reports**

  Examine the overall usage of the heap to understand where the memory is being allocated and held in your program. With TotalView's new Heap Status View, clearly see all the allocations your application has made so that you can tune your memory usage.

***Platform Updates***

- **NVIDIA**:
  - The latest A100 Ampere GPUs, including Multi-Instance GPU (MIG)
  - CUDA 11.3
- **Compilers**:
  - Intel 2021.1
- **MPI/OpenMP**:
  - OpenMPI 4.1.0

# TotalView 2021.1

***macOS Big Sur Support***

TotalView now supports Apple's latest version of macOS 11, Big Sur.

Note that Big Sur ships with a built-in dynamic linker cache of all system-provided libraries. As part of this change, copies of dynamic libraries are no longer present on the file system, so TotalView will be unable to construct full backtraces that extend into the dynamic system libraries.

***Remote UI Enhancements***

Remote UI session configurations now support specifying an optional ssh private key file when establishing the remote connection. Remote UI configurations now also allow shell commands to run before the remote TotalView debug server is launched, making it easy to set up paths, modules, and other environment settings before the debugger is started.

### *TotalView FlexNet Embedded License Server Update*

This release provides an update to the FlexNet Embedded license server and client code used within TotalView. If you have a FlexNet Embedded style license, typical for Linux ARM64 and Linux PowerLE platforms, you should schedule to update your license. Testing has shown the previous version will work fine with the latest versions of TotalView but for security and bug fix reasons it is always best to upgrade to the latest version of the license server.

### *The new UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- Memory debugging leak detection

  TotalView has long provided memory debugging support as part of its MemoryScape product and is now incorporating memory debugging into the new user interface. Easily enable memory debugging on your target process and, at any point during the debugging session, check for memory leaks. Mixing memory debugging and source code debugging brings many advantages, including the incorporation of TotalView's other debugging capabilities, such as reverse debugging.

### *Platform Updates*

- **Operating systems**: macOS 11 Big Sur
- **Compilers**: OpenMP Debug (OMPD) 5.x Support, Clang 12

## TotalView 2020.3

### *TotalView Remote Display Client Microsoft Windows Version Update*

The Microsoft Windows version of the Remote Display Client has been updated with various bug fixes. It also allows users to choose between the Remote Display Client Viewer or, if installed, the RealVNC Viewer.

### *Student License Updates*

TotalView Student has been updated to enable debugging on up to 32-core systems and to debug up to 32 processes at a time without a limit on threads. It also allows debugging of CUDA applications, use of reverse debugging and memory debugging.

### *The new UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Remote UI Connections**

  Run the TotalView interface from your Linux or macOS laptop and connect to a remote debugging session. Simply configure the remote host information, connect to the remote server, and TotalView will launch a remote debugger on the target host where all of TotalView's debugging tools are available to you.

- **OpenMP Debugging Enhancements**

  A new OpenMP view is available where developers leveraging the latest OpenMP v5 enhancements can easily view task and parallel regions, internal control variables, thread relationships, and runtime call-stack boundaries. OpenMP v5 is currently implemented only in the latest Clang compilers so OpenMP developers can use the latest features though with their own build of Clang.

### *Platform Updates*

- **Operating systems**: Fedora 32, Ubuntu 20.04
- **Compilers**: GCC 10

## TotalView 2020.2

### *CUDA 11 Support*

This release adds support for CUDA 11 enabling debugging support on NVIDIA's latest CUDA 11 based applications.

### *Solaris ported to 64 bits*

TotalView on Solaris has been ported to a 64-bit application, allowing the debugging of very large targets with gigabytes of debug information.

### *TotalView Remote Display Client updated*

TotalView's Remote Display Client (RDC) has been updated for the Linux and macOS platforms. The RDC is a convenient tool for easily setting up a remote debugging session and is distributed as part of TotalView. Simply install the RDC on your laptop or workstation using the GUI installer and set up a session to the remote system you will debug on.

### *The new UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Dive-in-All to view a structure member across an array**

  When debugging an array of structures, it is often desirable to focus on one of the structure members across the entire array. This release adds the ability to focus on a specific structure's data member by simply right clicking on the data member and selecting "Dive in All". The resulting array can be analyzed with various array tools such as Array Statistics.

- **Dedicated Data View for focusing on specific data**

  Easily focus on a data expression by adding it to a new Data View. This allows users to view and analyze the data in a dedicated View without having to scroll past other rows of data.

### *Platform Updates*

- **Operating systems**: Support added for Fedora 31

## TotalView 2020.1

### *TotalView FlexNet Publisher License Update*

TotalView now supports FlexNet Publisher license server version 11.16.6 to address multiple FlexNet security vulnerabilities. This new version requires an update to your FlexNet license server. See "FlexNet Update" in the "Updating an Existing License" section of the *TotalView Installation Guide* for additional details.

### *FlexNet Embedded Failover Server Support*

FlexNet Embedded license users can now take advantage of failover licenses. Request a failover license at license@perforce.com, then see "Installing a Failover Server" in the TotalView Installation Guide.

### *The new UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **Stepping and Local Variable View Performance Enhancements**

The UI has been optimized with significant performance improvements to stepping, "nexting," and displaying local variables.

- **Array Statistics View Auto Update**

  The Array Statistics View now supports automatic updates when process execution halts, via a new Auto Update toggle button.

### *Platform Updates*

- **Operating systems**: macOS Catalina
- **Compilers**: GNU GCC and GFortran 9

## TotalView 2020

### *Display Thread Names*

If developers assign a specific thread name to threads, TotalView now displays it in the user interface (both classic and new UI). The **dstatus** CLI command has also been enhanced to display the thread name. Traditionally, TotalView has displayed only the thread system ID; now it also displays both the User Thread ID (if it exists) and the Kernel Thread ID.

### *Licensing Server Enhancements*

All platforms can now take advantage of a FlexNet Embedded (FNE) license. FNE licenses can be served either by an FNE license server or installed locally on the system on which TotalView will run, called buffer-style.

### *Enhanced Documentation Format*

TotalView documentation has been redesigned with a new look-and-feel, with more powerful search capabilities.

### *The new UI TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the new, modern user interface in this release include:

- **New Array Statistics View**

  View statistics about arrays through a new view that displays array statistics and supports slicing arrays to generate statistics on a sub-portion.

- **Updated Documents view**

  The Documents view now displays all active files and documents, not just source files.

- **Displaying Thread Names**, as discussed above.

### *Platform Updates*

- **Operating systems**: RHEL/CentOS 8
- **Compilers**: Intel Parallel Studio XE 2020

## TotalView 2019.3

### *The NextGen TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

New features added to the NextGen user interface in this release include:

- **New Documents view**
  Easily switch between open documents with the new Documents View so you can quickly jump to the source file you need to debug.
- **Detaching from processes in a debugging session**
  You can now detach from processes while debugging, allowing better management of a session and the ability to focus only on the processes you need to debug.

### *Cray Debugging Improvements*

Building on the TotalView Reverse Connect functionality released in TotalView 2019.2, numerous improvements and bug fixes make it much easier and more reliable to establish interactive debugging sessions on Cray supercomputers.

### *TotalView Tcl Engine Update*

The internal TotalView Tcl engine has been updated to version 8.6.9, bringing numerous new Tcl features. TotalView's CLI is built around the Tcl interpreter, but there are no changes to the CLI commands or functionality due to this upgrade.

### *Filtering dlopen Events Improvements to Increase Startup Performance*

When deferring the processing of shared library load events to improve startup performance, TotalView now supports negated glob matching patterns using the Tcl's **string match** command to more finely control which libraries are immediately recalculated for breakpoints and which are deferred.

### *ARM and NVIDIA GPU Server Debugging Support*

TotalView adds debugging support to the latest ARM/NVIDIA GPU Servers and latest version of the CUDA SDKs. This addition extends TotalView's already strong GPU debugging support on Linux x86-64 and IBM PowerLE platforms and brings the same powerful debugging capabilities to the ARM platform.

### *Platform Updates*

Added support for Open MPI 4

### *Deprecations*

- **Solaris 10** TotalView 2019.3 is the last release that supports Solaris 10. For 2020, TotalView will be built on Solaris 11.
- **Linux Power**
  As of 2019.3, TotalView has changed the base build platform to RHEL 6.5.
- **IBM Blue Gene/Q** For 2020, TotalView will no longer support the IBM Blue Gene/Q platform.
- **Intel Xeon Phi Offload Coprocessors** For 2020, TotalView will no longer support the Intel Xeon Phi offload coprocessors.

## TotalView 2019.2

### *The NextGen TotalView User Interface*

To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can also launch the new UI with the **-newUI** switch:

totalview -newUI

New features added to the NextGen user interface in this release include:

- A new **Local Variable View**: Local variables now appear in a separate view which can be moved or undocked and in which structures can be expanded in place.
- New **Action Points Preferences Tab**: There is a new Action Points tab on the Preferences dialog, including a preference to “Automatically focus on threads/processes at breakpoint” which now defaults to true. This means that any process or thread that hits a breakpoint will be brought into focus in your debugging session.
- New **Input/Output View** available: Enter input directly into the user interface instead of only through the terminal. To activate this workflow, turn on the “Input Output view support” in the **Preferences** dialog > **RW Labs** tab, then restart the debugger. A new Input/Output view will be available that allows you to enter input to your running application.

- **New Preference to change the UI Font Size**: You can now change the UI font size. 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.

### *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 in which 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 runtime 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. TotalView now recognizes 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' 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.

### *Automatic Process/Thread Breakpoint Focus Preference Change*

In past releases, the "Automatically focus on threads/processes at breakpoint" preference defaulted to false. This could create 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, requiring that users manually examine the set of processes and threads and change the focus.

For the 2019.2 release, the preference defaults 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 fals**e" to your **.tvdrc** file.

### *Stability Improvements and Platform Updates*

Support for Fedora 30 and the GCC 9 compiler. Numerous bug fixes and performance improvements include properly focusing on main for MPI, **execve** and Fortran applications.

## TotalView 2019.1

### *NVIDIA® Jetson AGX Xavier™ Support*

You can now debug your CUDA-based autonomous machine applications running on the Xavier's ARM 64-bit CPU and 512-Core Volta GPU architectures, including using TotalView's memory debugging technologies to find memory leaks and other memory problems in your code.

### *NextGen New Dark Theme and Other Features*

TotalView's NextGen UI, activated through the Display Preferences panel or using the **-newUI** command line option, offers several new enhancements, including:

- A new **Dark Theme**. The new UI is designed to support multiple themes. Try the first new theme by selecting the Dark icon from the Display tab on the Preferences dialog, then restart TotalView.
- The ability to use standard input **stdin** to debug programs that require input entered at the console while the application is running.
- Improved performance of UI updates when stepping applications and displaying large source files with many breakpoints.

### *Python 3 Support*

When debugging mixed language Python and C/C++ applications, you can now use Python 3.5 and above. A range of OS Python distributions are supported including Enthought. A variety of OS Python distributions have been tested including the Enthought Python distribution. The Anaconda Python distribution is not supported due to the unavailability of debug information with that distribution.

### *FlexNet Embedded Support for TotalView Clients on macOS*

The macOS platform now supports FlexNet Embedded style tokens.This enables customers sharing tokens between Linux Arm64, Linux PowerLE, and Linux x86-64 to also share them with macOS versions of TotalView.

### *Numerous Platform Updates and Bug Fixes*

This release includes numerous bug fixes, and operating system and compiler updates.

## TotalView 2019

The 2019 release mainly includes bug fixes, stability improvements, and operating and compiler updates.

### *Stability and Performance Enhancements*

Numerous CUDA enhancements have been added in this release to improve performance and stability when debugging multi-GPUs in a cluster environment and also when debugging host and GPU-managed memory variables.

The reverse debugging ReplayEngine has also been updated, resolving a warning on Intel Skylake architectures and a memory issue on systems using the InfiniBand interconnect.

Numerous user-reported bugs have also been fixed in this release.

### *Platform Updates*

Operating system additions include support for Mac OS Mojave (10.14), Fedora 28 and 29, and Ubuntu 18.04. TotalView has also been validated against GCC 8.

### *FlexNet Embedded License Server Specification File*

Linux ARM64 and Linux PowerLE users can now use a special license server specification file to instruct TotalView which FlexNet Embedded (FNE) license server it should use, especially useful when working in cluster environments. To set up this special FNE license server specification file, see Chapter 3, "License Setup: Linux PowerLE & ARM64," in the *TotalView Installation Guide*.

## TotalView 2018.3

### *The NextGen TotalView User Interface*

TotalView's new user interface is now officially supported. To change between the new UI and the TotalView Classic UI, use the Preferences dialog on the Display menu. You can launch the new UI with the **-newUI** switch:

```
totalview -newUI
```

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

- **Barrier Point Support**
  The ability to create Barrier Points to synchronize threads and processes has been added.
- **Set Breakpoint**
  You can now use the Set Breakpoint menu item on a selected source line number to create a breakpoint.
- **Bug fixes and improvements**
  Numerous bug fixes and minor improvements have been made to the new UI.

Currently, the next generation UI is supported on Linux x86 64-bit, Linux PowerLE, Linux ARM64, and Apple's Mac OS X platforms. It supports multi-process and multi-threaded debugging as well as a level of parallel, MPI and CUDA debugging. Functionality not yet present in the UI is available through the command line interface (CLI). Please send email to tv-beta@perforce.com with your feedback and feature priorities. We welcome all feedback and feature requests for the new user interface.

### *Improved CUDA Debugging Support*

For 2018.3, TotalView provides a number of advances with its support for debugging CUDA applications:

- CUDA 9.2 and CUDA 10.0  This release provides official support of CUDA 9.2 and CUDA 10.0 as well as CUDA 8.0 and 9.0-9.2.
- Multi-GPU debugging improvements  TotalView 2018.3 improves the capabilities, reliability, and performance of debugging on advanced multi-GPU capable applications.
- CUDA debugging improvements  Continued improvements to the TotalView new UI have been made to improve debugging of CUDA applications. TotalView's 2018.3 enhancements include a new GPU navigation bar that allows for easy navigation between the Logical or Physical coordinates of the GPU, performance improvements when displaying breakpoints for GPU code, and other stability improvements.

### *Platform Updates*

TotalView 2018.3 updates its OpenMPI support to version 3.1.2 and introduces support for the Cray ARM64 architecture.

## TotalView 2018.2

### *The NextGen TotalView User Interface*

With this release, TotalView's new user interface is now officially supported and is no longer an Early Access feature. For new TotalView users, the new UI is displayed by default. For existing TotalView users, the default UI is still the TotalView Classic UI, but any user can change the default using the Preferences dialog on the Display menu, or launch the new UI with the **-newUI** switch:

totalview -newUI

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

- **Welcome to TotalView's New UI page**  When you start the new UI, a new "Welcome to TotalView's New UI" page displays resources about the new UI and how to use it.

- **Set PC** The ability to set the PC to a new line location has been added. To change the PC, select a valid line in the source code, then choose the **Thread |Set PC** menu item or simply hit the "p" key.
- **Data View** A number of enhancements have been added to better display changing data and data collections during program execution.
- **Action Points** Action points now accurately display their state as dynamic code from CUDA kernels or shared libraries are loaded.
- **Multi-process debugging improvements** Various aspects of the new user interface now use the share group of the process in focus for operations and data displayed. For example, only source files and action points related to the current share group in focus are shown. When changing focus to another process in a different share group, the source view and action point view update to show the related files and action points.
- **Bug fixes and improvements** Numerous bug fixes and minor improvements have been made to the new UI.

Currently, the next generation UI is supported on Linux x86 64-bit, Linux PowerLE, Linux ARM64, and Apple's Mac OS X platforms. It supports multi-process and multi-threaded debugging as well as a level of parallel, MPI and CUDA debugging. Functionality not yet present in the UI is available through the command line interface (CLI). Please send email to tv-beta@perforce.com with your feedback and feature priorities. We welcome all feedback and feature requests for the new user interface.

### *TotalView "What's New" Splash Screen*

In order to better promote new features in TotalView, a "What's New" splash screen is displayed after a new version of the product is installed, providing information about, and links to, the latest new features. You can dismiss the splash screen and not show it again until the next new release is installed. To view the What's New dialog again, select the "What's New in TotalView" menu item from the Help menu.

### *CUDA Debugging Model and Unified Display Improvements*

With this release, TotalView improves on the ability to easily set action points within CUDA applications and applications that dynamically load shared libraries with **dlopen**. In either case, until the CUDA or shared library code is loaded, the information required for setting a breakpoint is not available to the debugger. To address this issue, TotalView now allows setting a breakpoint on any line in the Source View, whether or not it can identify executable code for that line. The breakpoint becomes either a pending breakpoint or a sliding breakpoint until the CUDA or shared library code is loaded at runtime.

### *QString Type Transformation*

TotalView now automatically transforms instances of type QString in Qt4 and Qt5 applications so users no longer need to locate and manipulate the underlying character data to a human-friendly format.

### *Manage Single-Stepper Skip Rules*

TotalView now provides the ability to define single-stepper "skip" rules that modify the way source-level single stepping works. These rules identify functions that require no debugging. Skip rules can be defined to *skip over* a function or *through* a function.

- **Skip over rules** direct the debugger to step over a function rather than into it. These are useful for skipping over library functions such as C++ STL code.
- **Skip through rules** direct the debugger to ignore any source-line information for the function, so that single stepping does not stop at source lines within the function. If the function being skipped through calls another function, that call is handled according to the original single-stepping operation. Skip through is most useful for callback or thunk functions.

## TotalView 2018.1

TotalView® for HPC 2018.1 includes the following primary new or updated features. For a complete change history for TotalView, MemoryScape and ReplayEngine, see the document "TotalView_Change_Log.pdf" in the PDF directory of your installation, or the "TotalView Change Log" on the TotalView documentation site.

### *New exec and fork handling controls*

New with 2018.1, TotalView allows you to control how `fork()`, `vfork()`, `clone()` (when used without the CLONE_VM flag), and `execve()` system calls are handled by the debugger. Using the new command line options or CLI state variables, users can now define if the debugger should stop the process, continue the process, or ask whether or not to stop the process when the fork'd or exec'd process is acquired in the TotalView debugging session. For example, to configure TotalView such that when **bash** calls exec, the process automatically continues with no questions asked, but for other processes TotalView does ask to continue, use the following **dset** CLI command or TotalView command option:

```
dset TV::exec_handling {{{^bash$} go} {. ask}} totalview -exec_handling '{{^bash$}
go} {. ask}'
```

Above, the `<regexp>` is wrapped in an extra set of curly braces to make sure that Tcl did not process the "$" as a variable reference. Setting up **fork_handling** rules work in a similar manner.

See the **TV::exec_handling** and **TV::fork_handling** command line option and state variable documentation in the *TotalView Reference Guide* for more information.

### *NextGen TotalView User Interface*

TotalView's new UI continues to add new or updated features. Try out the new user interface with the **-newUI** switch:

totalview -newUI

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

**Multiple Data Views**The Data View is used to explore variables and debug data in your program. To better manage the display of your data, multiple Data Views can now be created by clicking the **Duplicate View** icon in the toolbar of the Data View. This creates a new Data View and clones any selected items in the original Data View into the new one.

**Data View Informational Drawer Tab**To find out more information about an expression in the Data View, a Drawer was added to the view with a new Information tab. The Information tab displays the expression, scope, thread, type, address, language, and other information for the selected item in the Data View. Adjust the size of the Drawer by clicking on the grab bar and moving it up or down. Double clicking on the grab bar snaps it open or closed.

**Bug Fixes and Improvements** Numerous bug fixes and minor improvements have been made to the new UI.

### *Team Plus License Support for Linux PowerLE and Linux ARM64*

TotalView 2018.1 adds proper Team Plus license support for the Linux PowerLE and Linux ARM64 platforms. Prior to this release, customers could only share tokens between these two platforms. With 2018.1, users can now use a Team Plus style license and properly share tokens between Linux PowerLE, Linux ARM64, and Linux x86-64. Please send email to tv-beta@perforce.com to request additional platforms for Team Plus.

### *New/Classic User Interface Preference*

A preference has been added to TotalView's preference panel that allows you to choose which user interface, the new or classic UI, TotalView should use at start up. To set this preference, click on the **Settings** gear icon in the main window toolbar or select **Preferences** from the top-level **File** menu.

### *Platform Updates*

TotalView 2018.1 introduces support for the Fedora 27 system and Clang 5 and Intel 2018 compilers, as well as support for the HPE MPI 2.

## TotalView 2018

TotalView® for HPC 2018 includes the following primary new or updated features. For a complete change history for TotalView, MemoryScape and ReplayEngine, see the document "TotalView_Change_Log.pdf" in the PDF directory of your installation, or on the TotalView documentation page on the web.

### *The NextGen TotalView User Interface*

TotalView's new UI continues to add new or updated features. To try out the new user interface, start TotalView with the **-newUI** switch:

totalview -newUI

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

**Python Debugging Support for ctypes**Python debugging in NextGen now supports filtering of ctypes "glue" frames that tie together function calls between Python and C/C++. This allows developers to see a clean stack trace between the two languages as they would expect, without the unnecessary noise of the layers required to shepherd data and make the calls between the languages.

**Launching parallel sessions**Launch your parallel job easily through NextGen's user interface using the new Parallel Session dialog. Simply click on Debug a Parallel Program from the Start Page or from the top-level File menu and select the Parallel System, specify the Program Details and then launch.

**Managing Action Points from a New Source View Context Menu**The Source view now supports the ability to enable, disable, delete and view properties of action points through a context menu accessed by right-clicking on the action point line number. This supports more streamlined debugging sessions.

**Easier Tooltip Viewing**TotalView has improved how it displays very long string within tooltips.

**Bug Fixes and Improvements** Numerous bug fixes and minor improvements have been made to the new UI.

### *License Server Support for Linux PowerLE and Linux ARM64*

On Linux PowerLE and Linux ARM64 platforms, TotalView requires the use of FlexNet Embedded license technology. With 2018.0, license server support has been added, enabling the sharing of team-based tokens across multiple systems of the same architecture. Contact license@perforce.com if you need to convert your existing single node FlexNet Embedded style license to a license server version.

## TotalView 2017.3

TotalView® for HPC 2017.3 includes the following primary new or updated features. For a complete change history for TotalView, MemoryScape and ReplayEngine, see the document "TotalView_Change_Log.pdf" in the PDF directory of your installation, or the "TotalView Change Log" on the TotalView documentation site

### *Early Access to the NextGen TotalView User Interface*

This early version of the NextGen interface continues to add features and is available so we can gather feedback from TotalView users on its updated capabilities. To try out the new user interface, start TotalView with the **-newUI** switch:

totalview -newUI

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

- **Watchpoint Expressions** Added support for watchpoint expressions, which allow you to define a short expression to run when the watchpoint triggers. Use the expression to check the new value of a variable, display information, or execute some other type of custom logic.
- **Modifying Properties on Action Points** Added support for editing properties on each Action Point type, including breakpoints, watchpoints, and evaluation points. Use properties to make minor changes to an existing action point without having to delete it and recreate it.
- **Improvements to Lookup View Search Algorithm** Enhanced and refined the search algorithm used by the Lookup View in order to remove duplicates and better prioritize the found results.
- **Incremental Display of Data in the Data View** Performance and scalability improvements have been made to the Data View allowing for the incremental display of very large structures.
- **Python Debugging** Removed the need for a debug build of the Python interpreter for Python debugging. You may use the standard Python interpreter that was distributed with your operating system. Read more about Python debugging capabilities in Chapter 7 of the *NextGen TotalView User Guide*.
- **Bug Fixes and Improvements** Numerous bug fixes and minor improvements have been made to the new UI.

Currently, the next generation UI is supported on Linux x86 64-bit, Linux PowerLE, Linux ARM64, and Apple's Mac OS X platforms. It supports multi-process and multi-threaded debugging as well as a level of parallel, MPI and CUDA debugging. Functionality not yet present in the UI is available through the command line interface (CLI). Please send email to tv-beta@perforce.com with your feedback and feature priorities. We welcome all feedback and feature requests for the new user interface.

### *Improved Inline and Optimized Code Debugging*

Significant improvements were made to the way TotalView debugs inline functions. TotalView is now able to step the debugging session through the functions and show the local variables for each inline function call.

### *Evaluation Points Performance*

TotalView now evaluates most evaluation point expressions within the TotalView server, providing faster performance. Some expressions may still need to be evaluated on the front-end portion of the debugger, depending on the complexity of the expression or variable data being accessed.

### *CUDA 9*

TotalView has been validated against the latest release of the CUDA SDK, CUDA 9.

### *Platform Updates*

TotalView 2017.3 introduces support for the Fedora 26 platform and GCC 7.1 compiler, as well as Mac OS X High Sierra.

# TotalView 2017.2

TotalView® for HPC 2017.2 includes the following primary new or updated features. For a complete change history for TotalView, MemoryScape and ReplayEngine, see the document "TotalView_Change_Log.pdf" in the PDF directory of your installation, or the "TotalView Change Log" on the TotalView documentation site.

### *Early Access to the NextGen TotalView User Interface*

This early version of the NextGen interface continues to add features and is available so we can gather feedback from TotalView users on its updated capabilities. To try out the new user interface, start TotalView with the **-newUI** switch:

totalview -newUI

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

- **Official Support - Mixed Language Debugging with Python and C/C++**Mixed language debugging of Python and C/C++ applications enables you to easily see a fully integrated call stack across the language barriers and to examine data passed between the layers. Read more about the Python and C/C++ debugging capabilities in Chapter 7 of the *NextGen TotalView User Guide*.

Currently supported platforms:

Python 2.7 on Linux x86 64-bit, Linux PowerLE, and Linux ARM64

- **Evaluation Points Support Added to the New UI**The ability to create and modify evaluation points, which are snippets of code that run when a point of code is hit, has been added to the new UI. Evaluation points are an excellent resource to add conditional logic for stopping execution of your program or trying out new execution logic without modifying your program. Read more about using evaluation points in Chapter 5 of the *NextGen TotalView User Guide*.

- **Data View Improvements for Deep Structures**Improvements have been made to the Data view to better handle structures with many pointers to other structures, creating deep trees of information. The depth of the tree is clipped to keep performance in check but the branches can easily be explored by right clicking on the data element and selecting **Dive** from the context menu.

Currently, the next generation UI is supported on Linux x86 64-bit, Linux PowerLE, Linux ARM64, and Apple's macOS/Mac OS X platforms. It supports multi-process and multi-threaded debugging as well as a level of parallel, MPI and CUDA debugging. Functionality not yet present in the UI is available through the command line interface (CLI). Please send email to tv-beta@perforce.com with your feedback and feature priorities. We welcome all feedback and feature requests for the new user interface.

### *ReplayEngine Reverse Debugging Added to tvscript*

**tvscript** enables unattended debugging functionality that is often useful in test, continuous integration, and batch environments where interactive debugging is not always feasible. With this release, users can now enable reverse debugging through **tvscript** and use an event driven approach to generate ReplayEngine recording files for later analysis. A common use is to enable reverse debugging as part of an overnight test run and if a test failure occurs, generate a ReplayEngine recording file and attach it to a bug report for later analysis. See Chapter 4 in the *NextGen TotalView Reference Guide* for more information about **tvscript**.

### *.gdb_index Section Support*

TotalView now supports processing **.gdb_index** sections contained within executable and shared library files. Compiling with DebugFission and linking with the gold linker to produce a **.gdb_index** can greatly reduce link time, disk usage, and debugger start-up time for large applications. See the *Compiling and Linking Split DWARF* section in Chapter 9 of the *TotalView Reference Guide* for more information on compiling programs with Split DWARF.

### *Solaris Split DWARF Support*

Starting with the Solaris Studio 12.4 compilers, Oracle® supports a Split-DWARF variant (also know as excluded DWARF) that can greatly reduce link time, disk usage, and debugger start-up time for large applications. See the *Compiling and Linking Split DWARF* section in Chapter 9 of the *TotalView Reference Guide* for more information on compiling programs with Split DWARF.

### *dwz Optimized DWARF Support*

TotalView supports debugging ELF shared libraries and ELF executables that are processed with the dwz tool, a tool for optimizing and removing duplicate debug symbols. For more information about dwz, see the dwz (1) Linux man page.

### *Platform Updates*

TotalView 2017.2 introduces support for the ARM64 platform and Absoft 17 compiler.

### *Bug Fixes & Performance Improvements*

A significant number of bug fixes and improvements have been added to the 2017.2 release.

## TotalView 2017.1

TotalView® for HPC 2017.1 includes the following primary new or updated features. For a complete change history for TotalView, MemoryScape and ReplayEngine, see the document "TotalView_Change_Log.pdf" in the PDF directory of your installation, or the "TotalView Change Log" on the TotalView documentation site.

### *Early Access to the NextGen TotalView User Interface*

This early version of the NextGen interface continues to add features and is available so we can gather feedback from TotalView users on its initial capabilities. To try out the new user interface, start TotalView with the **-newUI** switch:

totalview -newUI

For more details on the new UI, see the in-product help through the **Help | Contents** menu item. New features added to the NextGen user interface in this release include:

- **Replay Bookmarks** For ReplayEngine supported platforms, Replay Bookmarks allow you to easily mark a point during your program's execution history and then jump back to that point in time. See the *Replay Bookmarks* section of in Chapter 9 of the *NextGen for TotalView User Guide.*
- **Early Access to Mixed Language Debugging with Python and C/C++** This release introduces mixed debugging of Python and C/C++ applications, enabling you to easily see a fully integrated call stack across the language barriers and to examine data passed between the layers. Read more about the Python and C/C++ debugging capabilities in Chapter 7 of the *NextGen TotalView User Guide*.

Currently, the next generation UI is supported on Linux x86 64-bit, Linux PowerLE, Linux ARM64, and Apple's macOS/Mac OS X platforms. It supports multi-process and multi-threaded debugging as well as a level of parallel, MPI and CUDA debugging. Functionality not yet present in the UI is available through the command line interface (CLI). Please send email to tv-beta@perforce.com with your feedback and feature priorities. We welcome all feedback and feature requests for the new user interface!

Distribution Tar Bundle Name Change for 32-bit Linux and Mac OS

- **Linux**: The names of the Linux x86-based 32-bit tar bundles now append "-32" to more clearly identify 32-bit versus 64-bit distributions.
- **Mac**: The names of the Mac OS distributed tar bundles now append "-64" to more clearly identify the architecture.

### *Bug Fixes & Performance Improvements*

A significant number of bug fixes and improvements have been added to the 2017.1 release.

## TotalView 2017.0

### *Updates to the NextGen TotalView User Interface*

New features added to the NextGen user interface in this release include:

- Creating unconditional watchpoints

- Creating breakpoints “At Location”

Currently, the next generation UI is supported on Linux x86 64-bit, Linux PowerLE, Linux ARM64, and Apple’s macOS/Mac OS X platforms. It supports multi-process and multi-threaded debugging as well as a level of parallel, MPI and CUDA debugging.

### *ReplayEngine Performance Improvements*

Performance for the ReplayEngine **GoBack** operation has been substantially improved. The magnitude of the improvement depends on the nature of the program being debugged, but at least a 10X factor has been measured when running a process backward until it hits either an action point or the beginning of recorded Replay history.

### *NVIDIA® Tesla® P100 GPU with NVIDIA Pascal™ GPU architecture*

This release enables debugging support on NVIDIA’s new Tesla P100 GPU featuring its Pascal GPU architecture.

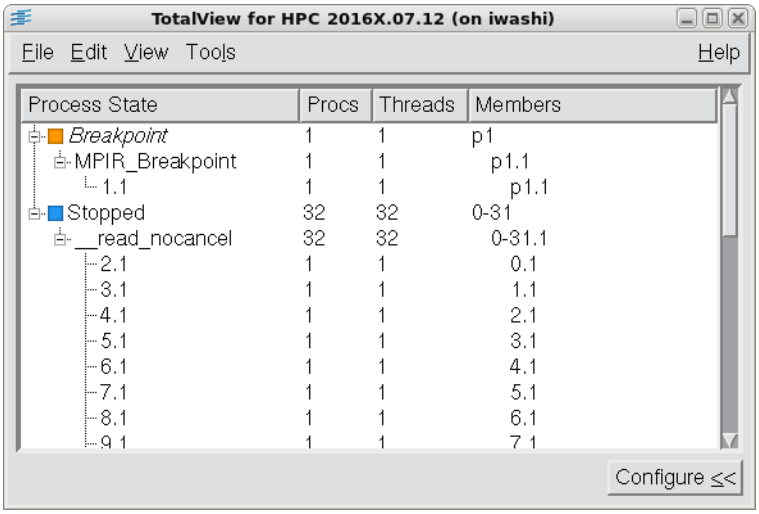
## TotalView 2016.07

### *Early Access Support for Linux ARM64*

The TotalView 2016.07 release adds support for Apple’s macOS Sierra.

### *Root Window Enhancements*

TotalView’s Root Window displays grouped threads and processes using common properties and allows you to quickly see the state of all processes and threads in your debugging session. For this release, the Root Window now features color coded icons to help you quickly distinguish between the different process and thread states.



### *Bug Fixes and Improvements*

- Improved stepping and debugging through inlined functions
- Significant startup and stability improvements when debugging parallel CUDA jobs
- Performance and stability enhancements for TotalView's reverse debugging feature, ReplayEngine

## TotalView 2016.06

### *Early Access to the NextGen TotalView Interface*

The TotalView team continues to develop the next generation user interface. This release adds two new features to the NextGen interface:

1. **An environment variable `TVNEWUI`** to easily launch the NextGen interface, so you can now try out the new user interface in two ways:
   - To launch the new UI for a single instance of TotalView, simply add the `-newUI` switch after the totalview command, for example:

     ```
     totalview -newUI
     ```

   - To routinely use the new UI, set the environment variable `TVNEWUI` to `True` and run the `totalview` command, like so:

     ```
     setenv TVNEWUI True
     totalview
     ```

2. **Source Search Path Management**

If TotalView cannot find the source for your program, you can now adjust the source search path directly through the Search Path preferences panel in the new UI. To change the source search path click on the **Settings** gear icon in the main window toolbar or select **Preferences...** from the top-level **File** menu. Use the controls in the Search Path Configuration panel to change where TotalView finds sources for your program.

### *TotalView License Software Update*

All versions prior to 11.13.1.2 of the Flexera Software FlexNet Publisher, the licensing software used by TotalView, contain a buffer overflow vulnerability that may be leveraged to gain code execution.

TotalView has been updated to use the latest version and so contains an updated vendor daemon. You must update your license server installation to at least 11.13.1.2 to use this version of TotalView in order to eliminate the security vulnerability. See the *TotalView Installation Guide* for more information about setting up the license server. The updated licensing software is included in the distribution.

### *CUDA 8 Support*

TotalView has been tested against the latest CUDA 8 release candidate and works as expected for CUDA debugging. When the final version CUDA 8 is released, TotalView's CUDA 8 support will be revalidated, but no changes are anticipated.

### *Early Access Support for Native SLURM Using MRNet on Cray CTI*

TotalView is providing Early Access support for the preliminary implementation of scalable debugging on Cray systems running native SLURM. Specifically, you can debug a job launched by SLURM's srun command, and TotalView will use the MRNet scalable infrastructure by default.

Note that, for this release, you cannot also use ReplayEngine, due to Cray file staging issues. If the -replay option is used, TotalView cannot connect to any parallel process started by srun. We'll be looking further into this issue in subsequent releases. See the release notes for more details about the Early Access support of native SLURM.

### *Platform Updates*

TotalView 2016.06 introduces support for the following platforms, compilers and parallel environments.

- Operating Systems:

  None

- Compilers:

  IBM XLF 15.1.3 and XLC 13.1.3 on AIX and Linux Power

- Parallel:

  None

# TotalView 2016.05

# TotalView 2016.04

### *Early Access to the NextGen TotalView User Interface*

The TotalView team continues to develop the next generation user interface. To try out the new user interface, start TotalView with the `-newUI` switch:

```
totalview -newUI
```

Each release will include additional functionality based on a priority list that you can help influence. Please send email to tv-beta@perforce.com with your feedback and feature priorities. For information about the new user interface, check out the in-product help through the **Help | Contents** menu item.

Here are the new features added to the new interface in this release:

- **New platform support: Linux PowerLE**
- **Initial MPI and OpenMP debugging support**

Parallel jobs launched through the command line interface now start a debugging session. See the MPI and parallel debugging documentation in the *TotalView User Guide* for information on how to launch a parallel job from the command line. An upcoming release will bring the Parallel Session capability into the new user interface.

- **Support for evaluation and barrier points**

While barrier and evaluation points must be initially created from the command line, once created they are displayed in the new interface Source and Action Points views. They can then be enabled, disabled, deleted, and suppressed like breakpoints.

- **Improved source search algorithms**

The locating of source files through the Source view and the Find Files and Functions view uses improved algorithms. The interface also responds to changes to the EXECUTABLE_SEARCH_PATH variable. See the *TotalView Reference Guide* for information on setting this variable through the command line.

- **Improved support for CUDA debugging**

The interface correctly displays CUDA source code and threads, and supports the setting of breakpoints in CUDA code.

### *Continued Performance Improvements*

This release of TotalView has numerous improvements to better handle very large executables with very extensive debug information.

### *Calling Functions when Running ReplayEngine Recorded Execution History*

TotalView now supports the calling of functions through the TotalView expression system when the ReplayEngine reverse debugging engine is activated. This allows for calls to functions such as printf or your own functions while evaluating expressions in the debugger during playback mode. See the *Reverse Debugging with ReplayEngine* document for more information on how memory-related side-effects are handled during the evaluation of expressions.

### *Platform Updates*

TotalView 2016.04 introduces support for the following platforms, compilers, and parallel environments.

- **Compilers**:

PGI 16.1

Absoft 16

# TotalView 2016.01

### *Early Access to the NextGen TotalView User Interface*

While the interface is in its initial development stage, we are pleased to provide an early version so that we can gather feedback from TotalView users on its existing capabilities. To try out the new user interface, start TotalView with the `-newUI` switch:

```
totalview -newUI
```

Check out the in-product help through the **Help | Contents** menu item for more details about the new user interface. Currently, the next generation UI is available only on Linux x86 64-bit platforms. It fully supports multi-process and multi-threaded debugging but does not yet support parallel, MPI or CUDA debugging. It also does not yet include some features like evaluation, barrier and watchpoints. Each new release will include additional functionality based on a priority list that you can help influence. Please send email to tv-beta@perforce.com with your feedback and feature priorities. We welcome all feedback and feature requests for the new user interface.

### *Continued Performance Updates*

Each TotalView release incorporates performance improvements. For this release, improvements were made to leverage the GNU_HASH section to quickly read and process debug information.

### *Fortran Improvements*

A number of bug fixes and enhancements have been made to TotalView's Fortran support, including the functionality to display variables within nested modules as well as to display modules and the module data for the Intel Fortran compiler.

### *Platform Updates*

TotalView 2016.01 introduces support for the following platforms, compilers, and parallel environments.

- **Operating Systems**:

Mac OS X 10.11 (El Capitan)

- **Compilers**:
- **Parallel:**

Open MPI 1.10.0

# TotalView 8.15.10

### *C++11 Support*

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

### *CUDA 7.5 Support*

Support for CUDA version 7.5. TotalView continues to support CUDA versions 7.0 and 6.5.

### *Platform Updates*

- **Operating Systems**: Linux PowerLE
- **Compilers**: GCC 5.2 and Intel 16.0

# TotalView 8.15.7

### *Early Access Support for Linux PowerLE*

TotalView 8.15.7 provides early access support for the new Linux PowerLE (Little Endian) systems. This support includes major debugger functionality, including support for CUDA 7.0. Contact Rogue Wave support at tsupport-TotalView@perforce.com to try this early access version of TotalView on your Linux PowerLE system.

### *CUDA Core File Support*

TotalView now supports debugging core files generated from a GPU core dump when a GPU exception is encountered. TotalView loads GPU-generated core files in the same way as traditional core files, making it possible to inspect the state of GPU code and the point at which it crashed.

### *PGI OpenACC Support*

TotalView has been updated to work with the latest versions of the PGI compiler and has been verified to debug code written with the OpenACC Application Program Interface. This enables easy debugging of PGI OpenACC applications that offload code from a host CPU to an attached accelerator device.

### *Platform Updates*

- **Operating Systems**: Solaris 11, Ubuntu 15.04
- **Compilers**: PGI 15.5

# TotalView 8.15.4

### *Support for CUDA 7.0*

Support for CUDA version 7.0. TotalView continues to support CUDA versions 6.5 and 6.0.

### *ReplayEngine*

GUI support for saving and loading Replay Recording files

### *Platform Updates*

- **Operating Systems**: OpenSuSE 13.2, SuSE SLES 12, Red Hat Fedora 21
- **MPIs**: SGI MPT 2.12, OpenMPI 1.8.4, Intel MPI 5.0
- **Compilers**: Clang 3.5 for Linux-x86 (32-bit), Linux-x86-64

# TotalView 8.15.0

### *Significant scalability and performance improvements*

This release focuses on a new scalability infrastructure on Linux, Blue Gene/Q and Cray platforms that uses efficient broadcast and reduction operations for a quantum leap in scalability. TotalView can now be used across hundreds of thousands of processes and millions of threads. This change is particularly noticeable in the reduction in time between when you launch a job and when you can begin actual debugging.

### *Improved MemoryScape startup performance*

Scalability, performance, and reduced startup times have also been implemented for memory debugging. At very large scales, focusing on a single process is likely to result in the greatest improvements.

### *Root window aggregate display improvements*

The root window now displays an aggregated tree of information about the state of the processes and threads you are debugging, rather than the former list-based display. The new display offers flexibility in the type of data that is aggregated, and maintains the full ability to dive on processes and threads.

### *New compiler support*

TotalView now supports the Intel 15.0 and PGI C++ 14.4 compilers.

# TotalView 8.14

### *CUDA 6.0*

Support for CUDA version 6.0, including debugging applications that utilize the new Unified Memory feature. TotalView continues to support CUDA versions 5.5 and 5.0. Starting at CUDA 5.5 there is limited support for dynamic parallelism.

### *Performance and Usability Improvements*

Critical performance improvements, and usability enhancements that cancel long running operations when processing delayed symbols and when creating certain types of breakpoints.

### *Type Transformation of Additional Datatypes*

Added support for transforming the raw implementations of the **unordered_map**, **unordered_set**, **unordered_multimap** and **unordered_multiset** STL collection classes. This data is transformed into readable name/value pairs, making it much more intuitive for you to understand the data in your application.

### *FlexLM License Upgrade*

FlexLM software has been upgraded to the latest FlexNet v11.12.1 version. This eliminates the need to work around the Flexera `-INCREMENT` bug on most platforms. The platforms Linux-IA64, Linux Power, and AIX (RS6000) remain on FlexNet v11.11.1 either because v11.12.1 is not available or because of other, more serious bugs in the new FlexLM version. On these platforms, the workaround developed for TotalView 8.13 still applies, as described in the Release Notes.

### *Replay Save and Restore*

A Replay session can now be stored to disk and later retrieved, at which time you can perform all of the replay functionality that was available at the time of the save. See the section "Saving the Execution History" in Chapter 1 of *Reverse Debugging with ReplayEngine*. This is early access functionality.

### *Platform Updates*

PGI Compiler version 14.4, Red Hat Fedora 20, Argonne MPICH 3.1, Cray CCE 8.3.1, Intel Composer XE for Linux 2013 SP1 Update 2 (14.0.2), Red Hat Enterprise Linux 6u5 x86 64-bit and 32-bit, GNU GCC 4.8.2.

# TotalView 8.13

### *CUDA 5.0 and 5.5*

TotalView now supports CUDA versions 5.0 and 5.5. With CUDA 5.0 TotalView does not support dynamic parallelism at all. With CUDA 5.5 we have limited support for dynamic parallelism. You should be able to use TotalView in the CUDA 5.5 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.

### *Xeon Phi Memory Debugging*

TotalView's support for Intel Xeon Phi (MIC architecture) has been extended to include memory debugging with MemoryScape for Native and Symmetric modes. MemoryScape functionality is not yet available for Offload mode programs.

### *Xeon Phi Symmetric Mode*

Xeon Phi support has been further extended to include Xeon Phi Symmetric Mode across a Xeon Phi cluster.

### *Mac OS X Mavericks*

TotalView now supports the Macintosh OS X Mavericks platform. Please check the TotalView Reference Guide, Part III, “Platforms and Operating Systems,” for special installation instructions.

### *Scalable data aggregation*

Some CLI commands, including `dwhere` and `dstatus`, now provide options to aggregate the data they display, making it much easier to understand the status and location of all the processes and threads being debugged.

### *Improved breakpoint performance*

There is a significant improvement in performance when creating and using breakpoints to debug very large applications.

### *Early Access: Tree-Based Scalable Debugging Infrastructure*

TotalView's early access support for tree-based scalability has noticeably improved performance. This support is based on the MRNet overlay tree network, and is available to selected customers on request. This release also includes a new scalable root window that aggregates data about many processes rather than listing them out one per line.

### *FLEXLM and Security*

The version of FLEXLM being used with TotalView has been upgraded to a more current version, which offers improved security.

### *Platforms and Compilers*

Support for new versions of operating systems and compilers. For a complete listing of supported platforms, please see the document *TotalView -Platforms and -System Requirements*.

## TotalView 8.12

### *Sessions Manager*

This release introduces the Sessions Manager, which allows you to set the configuration for a debugging session and preserve it from session to session. The Sessions Manager also provides a single, centralized interface for initiating debugging sessions. See the *TotalView Getting Started Guide* and *TotalView User Guide* for information on this feature.

### *Xeon Phi Support*

TotalView now fully supports Xeon Phi (MIC architecture) using the Knights Corner implementation. This is a separately licensed feature.

For more information, see the PDF document TotalView_Intel_Xeon_Phi_Debugging.pdf.

### *Support for Intel AVX and AVX2, and AMD XOP instruction sets*

On Linux for the x86 and x86-64 architectures, TotalView's disassembler now supports most instructions from Intel's AVX and AVX2 set, and AMD's XOP set. As a result, the assembler code view will display these instructions, and single-stepping will work correctly for code containing these instructions.

ReplayEngine does not yet support these instructions, so reverse debugging will fail on code containing them.

### *Enhanced Addresses Dialog*

Setting action points on templatized or overloaded functions can result in a large number of individual action points, particularly in massively parallel programs. The new Addresses dialog for action points helps you to enable a subset of these action points that pertain to the problem you are trying to debug.

### *Cray ATP Support*

Cray Abnormal Termination Processing (ATP) stops program execution at the time of a crash so you can debug the problem. TotalView now makes it easy to attach to such a held process.

### *STL Container Support*

The TotalView STLView feature now includes support for the STL containers set, multiset, and multimap.

### *Support for Mac OS X Lion and Mountain Lion*

TotalView now supports the Macintosh OS X Lion and Mountain Lion platforms. Please check the TotalView Reference Guide, Part III, “Platforms and Operating Systems,” for special installation instructions.

# TotalView 8.11

### *Blue Gene/Q*

TotalView now supports the Blue Gene/Q platform.

### *CUDA 4.2*

Support added for the NVIDIA CUDA SDK 4.2 tool chain on Linux x86 64-bit and Cray XK CPU-based systems.

### *OpenACC and OpenMP Directives Programming*

Support on the XK6 platform for Cray's OpenMP Accelerator Directives and Cray's OpenACC Directives. For information on this support, see the section Directive-Based Accelerator Programming Languages in the T*otalView for HPC User Guide.*

### *Xeon Phi*

This release provides Early Access support for Xeon Phi (MIC architecture) using the Knights Corner implementation. This is a separately licensed feature.

# TotalView 8.10

### *Enhanced and Extended CUDA Support*

Support added for the NVIDIA CUDA SDK 4.1 tool chain on Linux x86 64- bit and Cray XK CPU-based systems. Support for the Cray platform represents the first extension of CUDA support beyond the Linux x86 64-bit platform. For more information on this support, see the *TotalView Platforms and System Requirements* guide.

### *ReplayEngine on Demand*

ReplayEngine can now be enabled on a running application. Formerly, ReplayEngine had to be enabled when the application was started. This enhancement includes the addition of a Record button on the Process Window toolbar, which allows you to easily enable Replay on a running process.

### *ReplayEngine on Cray XE*

Support for ReplayEngine has been extended to the Cray XE platform.

### *C++View in ReplayEngine*

C++View type transformations now work in the context of ReplayEngine. Note, however, that there are some specific behavioral considerations. Please see Using C++ View with ReplayEngine in the *TotalView Reference Guide* for details.

#### *Enhanced TVScript Scalability*

Batch debugging of large-scale MPI applications through TVScript has been fully certified to the level of 1024 process jobs, and 2048 threads per process.

#### *Enhanced Dive Visibility*

When the cursor hovers over a divable object in the TotalView Source Pane, a red, dotted-line box appears around the object text, clearly indicating that this is a divable object.

# TotalView 8.9.2

#### *Enhanced CUDA Support*

TotalView 8.9.2 adds support for the NVIDIA CUDA SDK 4.0 tool chain on Linux x86 64-bit systems. For information on the specific platforms supported, see the *TotalView Platforms and System Requirements* guide.

# TotalView 8.9.1

## New Platforms and Compilers for Version 8.9.1

New platforms supported are Red Hat Enterprise Linux 6, Red Hat Fedora 14, and IBM AIX 6.1.5.

New compilers supported are GCC/GFortran 4.5.2, Intel Composer XE 2011 for Linux and Mac, and PGI 11.2. New MPI environments are Intel MPI 4.0.1 and POE 5.2. See the *TotalView Platforms and System Requirements* guide.

## New Features for TotalView 8.9.1

#### *Enhanced CUDA Support*

- Support for CUDA 3.2 on Linux x86 and Linux x86-64 has been added. The following features are included
- Support for apps built with the CUDA 3.0, 3.1, or 3.2 SDK
- Compatibility with CUDA 3.0, 3.1 or 3.2 drivers
- Support for CUDA function calls on the stack (in addition to the inlined function support in previous versions)
- Handles exceptions in CUDA code
- Display of variables in GPU hardware registers

- Display of variables in GPU hardware registers
- Support for host pinned memory regions
- Support for CUDA contexts
- New CLI commands for CUDA functionality

### *Array Statistics CLI Commands*

You can now use a **dprint -stats** to programmatically gather the array statistics that were previously only available in the GUI. These can be directly printed to the TCL prompt or placed in a TCL associative array with the **-data** option.

### *Array Type Inheritance*

In the multi-dimensional array display, changes to types made in the variable window are picked up prior to launching the array display, resulting in the most up-to-date information.

### *Enhanced Parallel Backtrace*

We've improved the way we store stack trace data to improve the representation of recursive function calls in the parallel backtrace view. In addition, parallel backtrace data is now available through the CLI with the **dcalltree** command.

## Platform Changes in Previous Version 8 Releases

### *8.9 Changes*

TotalView supports Red Hat Enterprise Linux 5 Update 4 and 4 Update 8, Fedora 13, and Ubuntu 9.10. New compilers supported are GCC 4.5.0, GFortran 4.5.0, PGI 10.6, Intel 11.1, IBM C/C++ 11.1, and IBM Fortran 13.1. New parallel runtimes (MPIs) supported are MPICH 1.2.1p1, OpenMPI 1.4.1, Intel MPI 4.0, MVAPICH 1.2, MVAPICH2 1.4.1, SGI MPT 1.27, and IBM POE 5.1.

### *8.8 Changes*

TotalView supports the Fedora 12 and Ubuntu 9.10 platforms and compiler PGI 10.1.

### *8.7 Changes*

TotalView supports the Fedora 9 and 10 platforms; compilers XLF 12.1, XLC 10.1, Intel 11.1, GCC 4.4, gfortran 4.4, and Berkeley UPC 2.8; and parallel environments MPICH 2.1 and OpenMP 1.3.

### *8.6 Changes*

TotalView supports C/C++ compilers for the IBM Cell Broadband Engine, GNU Fortran from Red Hat, and the Sony BCU-100 Zego.

### *8.5 Changes*

TotalView supports the IBM Cell Broadband Engine.

### *8.4.1 Changes*

This release has updated the compilers TotalView supports. Consult the *TotalView Platforms and System Requirements* guide for more information.

### *8.4 Changes*

TotalView supports Apple Mac OS X 10.5 (Leopard).

### *8.3 Changes*

New operating system versions include:

- Apple OS X 10.4.5, 10.4.8, and 10.4.9
- Fedora Core 7
- Ubuntu 6.06

As always, we have added support for new versions of existing compilers and parallel runtime environments.

### *8.2 Changes*

TotalView 8.2 has added support for the following systems and compilers:

- SiCortex supercomputer
- Cray XT4 support and APLs integration
- Fedora Core 6
- Expanded Mac support

Preliminary Mac OS X Leopard, 64-bit Mac-Intel, and Mac Universal Binaries

- Ubuntu

Ubuntu is a community-developed Linux-based operating system for the desktop, laptop, thin client and server. TotalView will support applications developed on this platform.

You'll find a complete list of supported platforms and compilers in the *TotalView Platforms and System Requirements* guide.

## New and Changed Features in Previous Version 8 Releases

## Version 8.9 Features

Support for CUDA 3.0 on Linux x86 and Linux x86-64 has been added. The following features are included:

- Tesla and Fermi hardware support
- Device (not emulator) support
- CUDA function inlining support
- CUDA memcheck functionality support
- Linux threads and GPU device threads visibility
- Representation of hierarchical memory with type qualification
- Display and navigation using logical thread and block coordinates and hardware coordinates
- Breakpoints and stepping code running on the device
- Interoperability with MPI for use in accelerated clusters

A **2D array viewer** enables viewing a multi-dimensional array in a two-dimensional grid, allowing for setting the slice and stride within the dimensions to limit the amount of data in view. It includes a mechanism for controlling the numerical precision of the data displayed.

**C++View** is an extension of TotalView's type transformation facility, to allow formatting functions to be written in the user's target code and preferred language rather than in TCL code in the debugger.

The **parallel backtrace feature** provides a single view to the state of every process/thread in a parallel job. It displays the host, status, process ID, rank and the line of code being executed.

**TVScript** support has been expanded to include the Cray XT, Blue Gene/L and Blue Gene/P platforms.

The **Data Window** supports display of very long C++ expressions.

## Version 8.8 Features

This section lists the changes made for version 8.8 of TotalView.

- A Remote Display client for the Mac and Windows 7 operating systems has been added.
- Runtime performance at scale has been improved.

## Version 8.7 Features

This section lists the changes made for version 8.7 of TotalView.

- TotalView includes the MemoryScape GUI for memory debugging, with Red Zones on Linux platforms for detection and immediate notification of errors in heap arrays.
- Support for various heterogeneous debugging combinations and Power PC 32 cross debugging is introduced.
- Remote debugging is enhanced to support IP-only networks and nodes with multiple interfaces with different IP addresses.
- TotalView allows subset attach from the command line and via the CLI.
- New server launch CLI state variables and shared library search and mapping state variables are added, and the search path and mapping rules are changed.

## Version 8.6 Features

This section lists the changes made for version 8.6 of TotalView.

- Remote Display: TotalView can open a window on your machine that displays TotalView executing on a remote system. We provide installers for Windows running XP or Vista, Linux x86 and Linux x86-64. While Remote Display can run only on these operating systems, the remote TotalView can execute on any platform that TotalView supports.
- TVScript / Batch debugging: TotalView can run unattended using the **tvscript** shell command. The commands that tvscript executes can be entered as command-line options or by creating a file for **tvscript** to execute. See the *TotalView Reference Guide* for how to use the **tvscript** command.
- **Debug** is added to the TotalView menubar if you are running on Linux-x86 (32-bit) and Linux-x86-64. All memory commands are now within **Debug**.
- The current line is highlighted in yellow. If you have purchased a Replay license, an orange highlight line shows where you've gone back to. A separate marker shows the PC that existed when you entered replay mode.
- The **dhistory** command lets you invoke ReplayEngine from within the CLI. The spurs command displays information on spurs library use.
- Options supporting ReplayEngine are added to **dattach**, **dload**, and stepping commands such as **dstep**, **dnext**, **duntil**, etc.
- Options to the **dload** command let you start MPI programs. These options are **-mpi**, **-nodes**, **-starter_args**, **-np**, **-procs**, and **-tasks**.
- The **Process > Startup Parameters** dialog is rearranged and contains options for enabling memory debugging and ReplayEngine. This window comes up automatically when you start TotalView using a program's name as an argument.

- Other command-line options added for this release include **-local_interface** (most often used with Blue Gene), **-memory_debugging**, and **-replay.**
- New variables added at this release are TV::ask_on_cell_spu_image_load,**TV::cell_spu_image_ignore_regexp**, **TV::cell_spu_images_stop_regexp**, **TV::cell_spurs_jm_name**, **TV::cell_spurs_kernel_dll_regexp**, **TV::cell_spurs_ss_name**, **TV::cell_spurs_tm_name**, **TV::data_format_int128**, **TV::local_interface**, and **TV::GUI::heap_summary_refresh**.

## Version 8.5 Features

No changes are listed for version 8.5 of TotalView.

## Version 8.4 Features

This section lists the changes made for version 8.4 of TotalView.

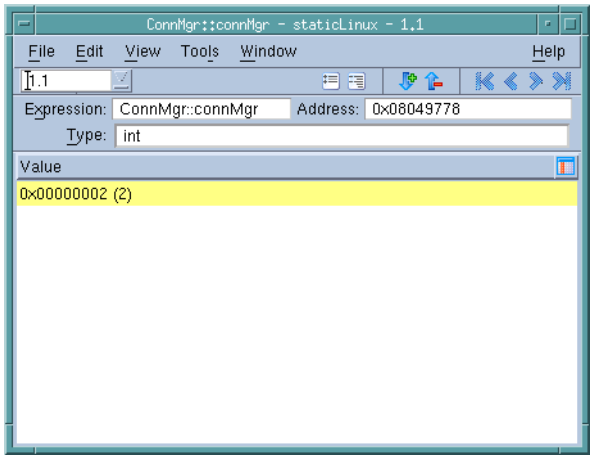
- If you have more than one TotalView license, you can control which kind of license TotalView uses by adding one of the following command-line options: -**team**, **-noteam**, **-teamplus**, **-noteamplus**, **-ent** or **-noent**.
- The TotalView Memory Debugger can now write light-weight memory debug files when an event occurs. These files are similar to the memory debugging files (**.mdbg**) files that you can write using the **File > Export** command. They differ in that they are designed to be written when the event occurs and in such a way that the program's behavior is minimally disturbed.
- Improved support for C++ templates.
- Improved support for Fortran modules on Apple Mac OS X.

## Version 8.3 Features

This section lists the changes made for version 8.3 of TotalView.

- Improvements to the way TotalView launches MPI programs let you use TotalView with virtually every MPI library, even with those that were not configured for debugging.
- TotalView now highlights changes to values displayed in the **Variable** and **Expression List** windows with a colored background.

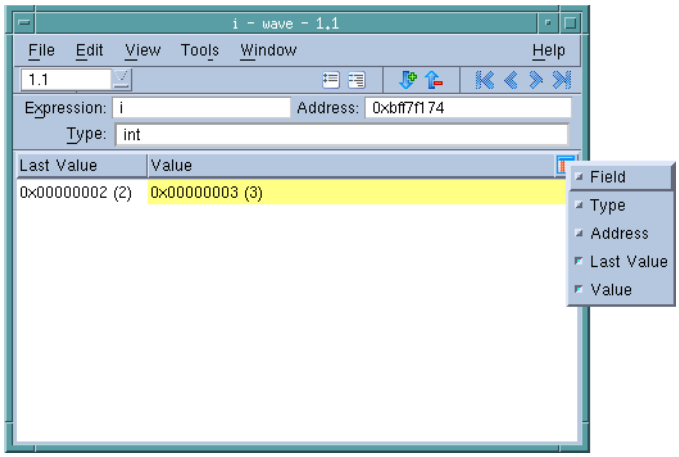
## Figure 1, Highlighted Change in the Variable Window




While this figure shows a simple variable, TotalView also highlights changed elements within compound variables such as structures and arrays.

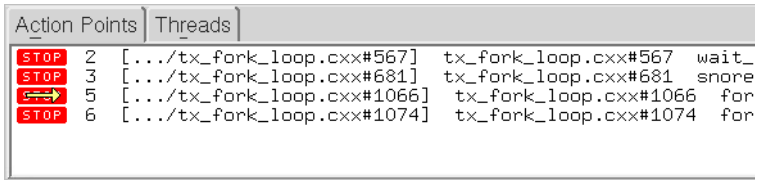
- Values in the **Variable** and **Expression List** windows have a **Previous value** hidden column that you can display. Use the control on the right side of the column headings to display a list of columns that you can display or hide.

## Figure 2, Last Value Column



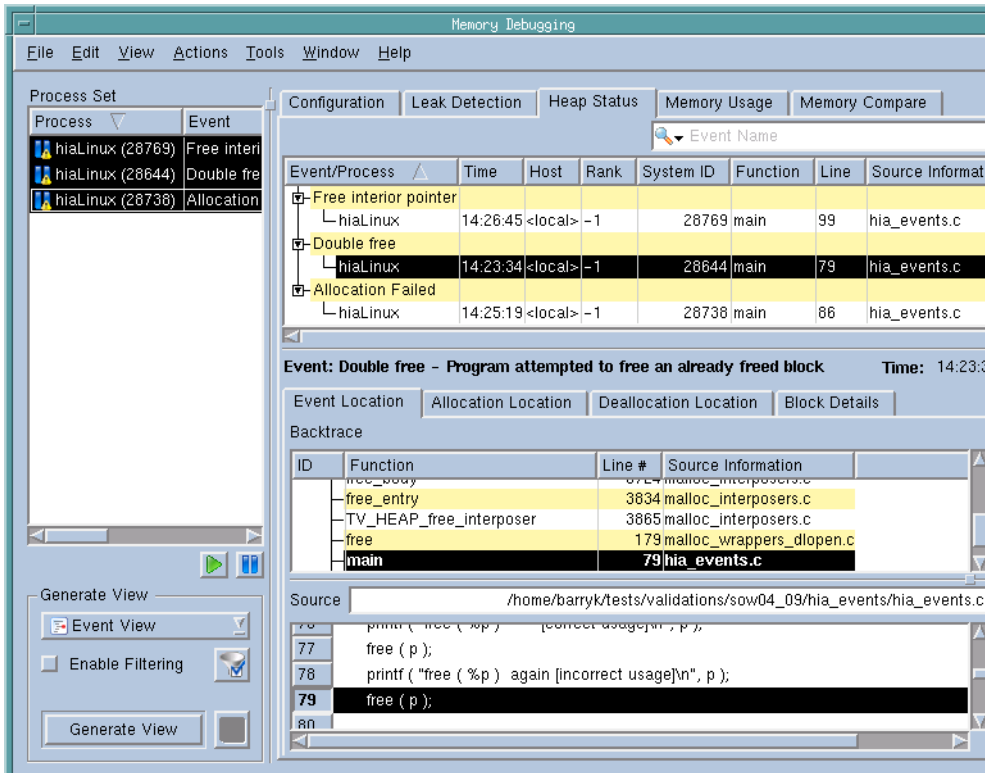
- When a process hits a breakpoint, TotalView highlights the breakpoint by placing an arrow over the breakpoint ID in the Action Points pane.

## Figure 3, Highlighted Action Point ID




- **View > Show Across** replaces **View > Laminate** in the **Variable** window's menus. This means the commands you will now use are **View > Show Across > Process** and **View > Show Across > Thread**.
- You can now tell TotalView to show a variable across processes or threads by right-clicking on it in the Source Pane, then selecting either **Across Processes** or **Across Threads** from the context menu.
- The **Create Watchpoint** command was added to the Action Points menu. As always, you can create a watchpoint from within the Variable window by selecting **Tools > Create Watchpoint**.
- You can now set a watchpoint upon a variable's memory address by right-clicking on the variable in the Source pane and then selecting **Create Watchpoint** from the context menu.
- You can completely expand or collapse information in the Variable window by selecting an icon in the toolbar. The accelerators for these commands are **Ctrl++** (that's the control key and the **+** symbol) and **Ctrl+-** (which is the control key and the - symbol).
- TotalView no longer stops by default when your program loads a library.
- You can specify more than one core file on the command line and you can use wildcards in core file names.
- There is a new Events View report within the Memory Debugger Heap Status tab.

## Figure 4, Event View




- Within the Memory Debugger's **File > Import Data** dialog box, you can select multiple memory debugging (**.mdbg**) files.

## Version 8.2 Features

This section looks at changes that have occurred within TotalView.

- Early-Access GUI Installer

You can now install TotalView from tar files as you've always done or install it using our new graphical installer. We are calling this an early-access release in that we want you to tell us what you think of it and how we can improve it.

- Fortran Parameter Display

TotalView now displays the value of Fortran parameters. Parameters can be used like variables in expressions but could not previously be examined within the debugger.

## Versions 8.0 and 8.1 Features

### *Breakpoint Changes*

Action points are considerably more powerful. Here is a summary:

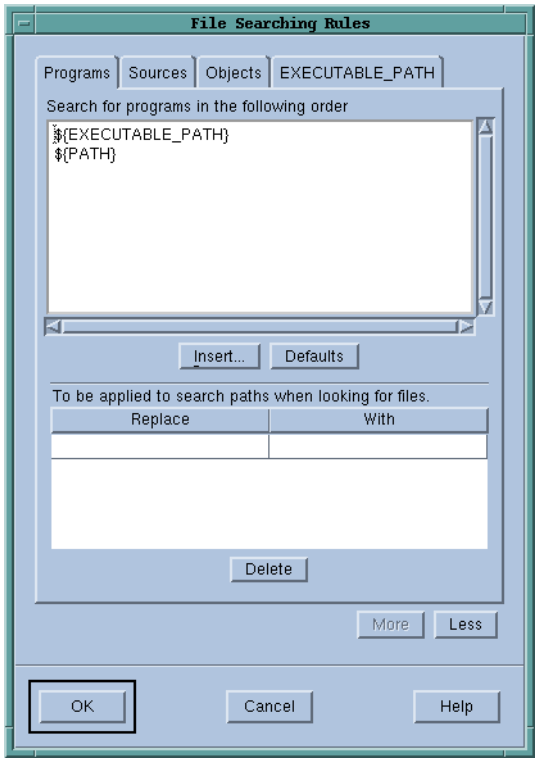
- The **Action Point > At Location** command now has three choices:
- **Function or Line**: lets you set a line number or a function name. This choice is what occurred in previous TotalView releases.
- **All Methods in Class**: lets you set a breakpoint on all methods in a class. This can set more than one breakpoint.
- **All Virtual Functions and Overrides**: lets you set breakpoints on virtual functions and their overrides. This too can set more than one breakpoint. You can now tell TotalView that a breakpoint will occur in a library that will be loaded later and that TotalView should retain knowledge of this breakpoint. This allows it to be set when the library is read. Previously, TotalView had to create and set breakpoints at the same time. This meant that when you enter a name into the **At Location** dialog box and the name is not yet known, TotalView, displays either an **Ambiguous Function** or a **Question** dialog box. At this time, you can set the breakpoint's status to *pending*. When you create a barrier breakpoint or change a breakpoint to a barrier point, the same new features are available.
- The CLI **dbreak**, **dbarrier**, and **dlist** commands have been extended to use these features. The argument to these commands can now be a breakpoint expression. Understanding this concept reveals some of the subtleties involved using these new features. These concepts are explained with the **dbreak** pages of the *TotalView Reference Guide*.

### *Other Features*

This section describes improvements and changes made in many different places in TotalView.

- The way in which you set search paths has changed.

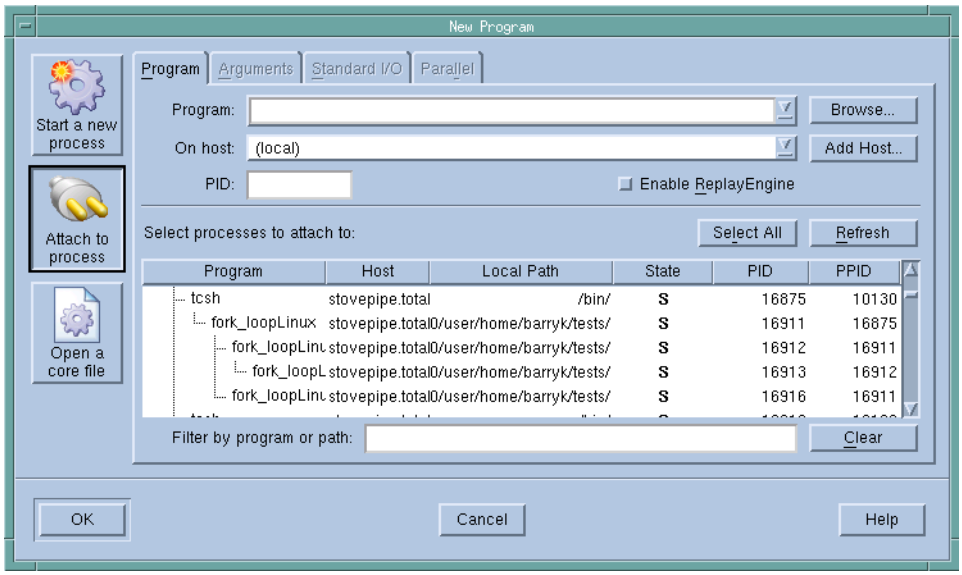
## Figure 5, File > Search Path Dialog Box




You have told us that most of the time, you usually do not need to set search paths. If you do, you just need to enter a couple of paths. This is what the old dialog box was designed for. (You can still do this by typing paths directly into the **EXECUTABLE_PATH** tab.) However, when your programs make the transition from modules being developed on your workstation to an place where the work of development teams is brought together, setting search paths was tedious and difficult to get right. This new dialog box, extensively documented in the help, lets you solve this problem.

- The **File > New Program** dialog box has changed. Much of what you will see makes the dialog box more usable. Most notable is the way you attach to processes. The dialog box now lets you select more ethan one process at a time. The one new feature is that you can now enable memory debugging from this dialog box.

## Figure 6, File > New Dialog Box




- The **View> Freeze** command is added to the Variable window. This command tells TotalView that it should freeze the contents of a Variable window. That is, as your program executes and as data values change, the contents of this window does not change. In most cases, you will also create a second Variable Window so that you can see old and new values at the same time.
- The **View > Lock** command is added to the Variable Window. This command tells TotalView that it should not change the address from which the Variable Window is obtaining information.
- New **dheap -compare** CLI command options. This options lets you compare the result of two different memory states.
- New **dkill -remove** CLI command option. Using this option to tell TotalView that, in addition killing the process, it should remove knowledge of the process. This is seldom necessary. However, if you are using TotalView Team, using this option makes the token used by a process available to another process in your program.
- The following CLI variables were added for this release
  - **TV::env**: sets an environment variable.
  - **TV::bluegene_server_launch_string**: sets the Blue Gene server launch string.

- **TV::default_sterr_append**: tells TotalView to append **stderr** information to **stdout**.
- **TV::default_stderr_filename**: tells TotalView to write **stderr** information to a file.
- **TV::default_stderr_is_stdout**: tells TotalView to write **stderr** information to **stdout**.
- **TV::default_stdin_filename**: Tells TotalView to read **stdin** information from a file.
- **TV::default_stdout_append**: Tells TotalView to append **stdout** information to a file.
- **TV::default_stdout_filename**: Tells TotalView to write **stdout** information to a file.

# MemoryScape

## Versions 2016.01 - 06

No updates.

## Version 3.15.4

No updates.

## Version 3.5

### *Xeon Phi Support*

TotalView's support for Intel Xeon Phi (MIC architecture) has been extended to include memory debugging with MemoryScape for Native and Symmetric modes. MemoryScape functionality is not yet available for Offload mode programs.

### *Platforms and Compilers*

Support for new versions of operating systems and compilers. For a complete listing of supported platforms, please see the document *TotalView -Platforms and -System Requirements*.

## Version 3.2.2

### *Platforms and Compilers*

We have added support for new versions of operating systems and compilers. For a complete listing of supported platforms, please see the *TotalView, MemoryScape, and ReplayEngine Platforms and System Requirements* guide.

### *New Features*

There are no major new features added in MemoryScape 3.2.2.

## Updates in Earlier 3.n Versions

### *Interoperability with TotalView*

MemoryScape offered greatly increased interoperability with the TotalView debugger. Launch TotalView from within an already running MemoryScape session to examine specific variables or exert more precise control over programs that you are debugging. Enabling memory debugging within a TotalView session will bring up the MemoryScape GUI.

Effective with TotalView 8.7 and later, you can run MemoryScape with a TotalView license that allows memory debugging.

### *Red Zone platforms*

Red Zones (heap memory read and write bounds checking with event generation at read/write time) are available on Solaris and Mac.

### *Support for heterogeneous debugging*

MemoryScape supports several forms of heterogeneous debugging, where the operating system and/or architecture differ. For example, from a Linux x86-64 session you can debug remote processes on Linux Cell.

This table shows the supported combinations:

| Host System | Target System |
|---|---|
| Linux x86-64 | Linux x86 (32-bit)Linux x86-64Linux Power 32Linux Power 64 / CellSiCortexCray XT |
| Linux x86 (32-bit) | Linux x86 (32-bit)Linux Power 32Linux Power 64 / Cell |
| Linux Power 64 (including Linux Cell) | Linux Power 32Linux Power 64 / CellBlue Gene |
| SiCortex Linux x86-64 | Linux MIPS 64 |

### *Support for Power PC32 cross debugging*

MemoryScape now supports debugging PowerPC32 embedded applications. Support is delivered through a cross debugger. The host system (where MemoryScape is running) must be one of the following platforms:

- x86-64 Linux
- x86 Linux (32-bit)

- Power64 Linux
- Cell Linux

### *Red Zones for Linux*

The Red Zones feature added to MemoryScape 3.2 provides instant array bounds detection for Linux systems. MemoryScape can immediately detect when a program tries to access memory beyond the allocation bounds.

- Red Zone events: MemoryScape uses Red Zones to detect access violations before and after allocated memory bounds. It can also detect when a program accesses memory that has been deallocated. MemoryScape will stop the program's execution and raise an event alerting you to the illegal access and allowing you to see exactly where the code overstepped the bounds.
- Red Zone allocation size range controls: Red Zones will increase the memory consumption of your program. To reduce this impact, special controls have been added to give you full control over how and when Red Zones are applied to allocated memory. You can restrict Red Zones to allocations in several user-defined size ranges and easily turn Red Zones on or off at any time during a program's execution.
- Red Zone support in the CLI, TVScript, and MemScript: Red Zones are supported in the CLI and TVScript and MemScript via new commands and command qualifiers. The appropriate product documentation provides the details.

### *Support for malloc zones on Mac OS X*

Mac OS X provides a mechanism for multiple pools of memory called malloc zones. MemoryScape tracks both the allocator and owner of all heap allocations. These properties can be displayed and used for filtering.

### *Hoard low memory detection*

When you ask MemoryScape to hoard deallocated memory, you may increase the risk of running out of available memory earlier than expected. MemoryScape has the capability to reduce this risk and alert you when you are at risk.

- Hoard low memory controls: you can tell the hoard to automatically release its memory when available memory gets low, allowing the program to run longer.
- Hoard low memory events: MemoryScape can stop execution and notify you when the hoard drops below a defined threshold, so that you know the program is getting close to running out of memory.
- Hoard low memory support in the CLI, TVScript, and MemScript: this feature is supported in the CLI, TVScript, and MemScript via new commands and command qualifiers. The appropriate product documentation provides the details.

# ReplayEngine

## Version 2016.06

### *dhistory Bookmarking Options*

The CLI's **dhistory** command used to perform actions on ReplayEngine now has new options that control bookmarks, and has deprecated some options:

- *New options*: -**create_bookmark**, -**goto_bookmark**, -**show_bookmarks**, -**delete_bookmark**
- *Deprecated options*: -**get_time** (use **create_bookmark**); -**go_time** (use **goto_bookmark**)

## Versions 2016.01 - 05

No updates.

## Version 2.15.4

No updates.

## Version 2.1 New Platforms and Features

### *Infiniband Support*

ReplayEngine now has complete coverage for major Infiniband configurations. We introduced support for IP over Infiniband several releases ago and have been gradually improving coverage. It is now possible to use ReplayEngine with native transport mechanisms with the following low-latency Infiniband hardware from market leaders Mellanox and QLogic:

- Mellanox IBverbs transport in Open MPI 1.4.2, MVAPICH2 1.5, MVAPICH2 1.6, and Intel MPI 4.0.
- QLogic PSM transport in Open MPI 1.4.2, MVAPICH 1.2, MVAPICH2 1.5, MVAPICH2 1.6, and Intel MPI 4.0.

For a complete listing of supported platforms, please see the *TotalView, MemoryScape, and ReplayEngine Platforms and System Requirements* guide.

## Previous 1.n Versions

### *Edit During Record*

ReplayEngine allows you to edit values while in Record mode.

### *Shared Memory Support*

Shared memory support includes Unix shmem, frequently used for intranode communication between process in HPC systems.

### *Backwards Continue Functionality*

A Backwards Continue function has been added, with support for action points (breakpoints, watchpoints, and expression points).

### *Controlling recorded history and memory constraints*

Turning on ReplayEngine with long-running applications often failed because of insufficient memory for recorded execution history storage. ReplayEngine 1.5 was modified to discard the oldest recorded history and continue when there is insufficient memory. This means that you will be able to move back in execution time only to the point at which the history has been discarded. This behavior is the default, but there are TotalView preferences that allow you to specify that ReplayEngine should stop the process instead when it runs out of memory.

You can also set the maximum size for the ReplayEngine history buffer. The default size is 'unlimited' and bound by the amount of memory available to the process.

## ReplayEngine 1.0 Summary

- ReplayEngine records the changes to program state as they happen.
- ReplayEngine replays previously executed commands. ReplayEngine commands are added to the TotalView toolbar. Generally, these commands let you specify which previously executed line in your program you want to examine. These commands are analogous to Next, Step, Out, and Run To. They differ in that they move into the program's history. That is, entering replay mode is done by pressing a single button. The commands behind these buttons are located on the **Instrumentation** entry added to the tool bar. (All memory commands have also been moved to **Instrumentation**.)
- When you are in replay mode, you can use ReplayEngine commands to move through your program's assembly code.
- Changing back to record mode is as simple as pressing the **Live** tool bar button. Reentering replay mode is just a button press.
- Multithreaded codes replay in precisely the same sequence as the threads previously executed. This is especially useful for examining race conditions.
- Breakpoints, watchpoints, and some conditional breakpoints can be used when running forward in replay mode.