RogueWave SOFTWARE

Accelerating Great Code

TOTALVIEW® CHANGE LOG

Versions 8.0 to 8.15.10

Copyright © 2010-2015 by Rogue Wave Software, Inc. All rights reserved.
Copyright © 2007-2009 by TotalView Technologies, LLC
Copyright © 1998–2007 by Etnus LLC. All rights reserved.
Copyright © 1996–1998 by Dolphin Interconnect Solutions, Inc.
Copyright © 1993–1996 by BBN Systems and Technologies, a division of BBN Corporation.
No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without the prior written permission of Rogue Wave Software, Inc. ("Rogue Wave").
Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.
Rogue Wave has prepared this manual for the exclusive use of its customers, personnel, and licensees. The information in this manual is subject to change without notice, and should not be construed as a commitment by Rogue Wave. Rogue Wave assumes no responsibility for any errors that appear in this document.
TotalView and TotalView Technologies are registered trademarks of Rogue Wave Software, Inc. TVD is a trademark of Rogue Wave.
Rogue Wave uses a modified version of the Microline widget library. Under the terms of its license, you are entitled to use these modifications. The source code is available at http://kb.roguewave.com/kb/.
All other brand names are the trademarks of their respective holders.

ACKNOWLEDGMENTS

Use of the Documentation and implementation of any of its processes or techniques are the sole responsibility of the client, and Rogue Wave Software, Inc., assumes no responsibility and will not be liable for any errors, omissions, damage, or loss that might result from any use or misuse of the Documentation

ROGUE WAVE SOFTWARE, INC., MAKES NO REPRESENTATION ABOUT THE SUITABILITY OF THE DOCUMENTATION. THE DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ROGUE WAVE SOFTWARE, INC., HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE DOCUMENTATION, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT SHALL ROGUE WAVE SOFTWARE, INC., BE LIABLE, WHETHER IN CONTRACT, TORT, OR OTHERWISE, FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT, PUNITIVE, OR EXEMPLARY DAMAGES IN CONNECTION WITH THE USE OF THE DOCUMENTATION.

The Documentation is subject to change at any time without notice.

Rogue Wave Software, Inc.

Product Information: (303) 473-9118 (800) 487-3217
Fax: (303) 473-9137
Web: http://www.roguewave.com

Contents

TotalView 2
TotalView 8.15.10 2
TotalView 8.15.7 2
TotalView 8.15.4 3
TotalView 8.15.0 3
TotalView 8.14 4
TotalView 8.13 5
TotalView 8.12 6
TotalView 8.11 7
TotalView 8.10 7
TotalView 8.9.2 8
TotalView 8.9.1 8
New Platforms and Compilers for Version 8.9.1 8
New Features for TotalView 8.9.1 8
Platform Changes in Previous Version 8 Releases 9
New and Changed Features in Previous Version 8 Releases 11
Version 8.9 Features 11
Version 8.8 Features 11
Version 8.7 Features 11
Version 8.6 Features 12
Version 8.5 Features 13
Version 8.4 Features 13
Version 8.3 Features 13
Version 8.2 Features 16
Versions 8.0 and 8.1 Features 16
MemoryScape 21
Version 3.15.4 21
Version 3.5 21
Version 3.2.2 21
Updates in Earlier 3.n Versions 21
ReplayEngine 24
Version 2.15.4 24
Version 2.1 New Platforms and Features 24

Previous 1.n Versions 24
ReplayEngine 1.0 Summary 25

TotalView Change Log, 8.0 - 8.15.10

This document lists updates made to the TotalView product for versions 8.0.0 through Version 8.15.10, including MemoryScape and ReplayEngine. For documentation about the current release, please see the documentation provided with your TotalView installation or TotalView Online Help and Documentation on the Rogue Wave website.

Note that beginning with releases after 8.15.0, the pattern of the version numbering has changed to:

<major-version-number>.<two-digit-year-of-release>.<month-of-release>

So a release in April of 2015 with the same major version number would be 8.15.4.

TotalView

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 `support@roguewave.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, 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 Getting Started Guide and User Guide for information on this new 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 *TotalView 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
- 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 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.

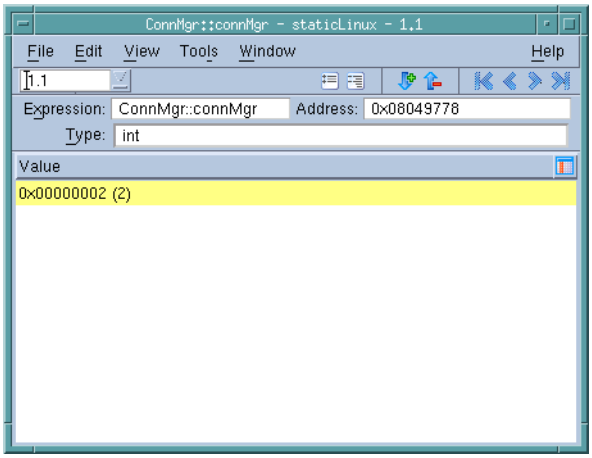
- If you have more than one TotalView license, you can control which kind of license TotalView uses by adding one of the following command-line options: -**team**, **-noteam**, **-teamplus**, **-noteamplus**, **-ent** or **-noent**.
- The TotalView Memory Debugger can now write light-weight memory debug 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.

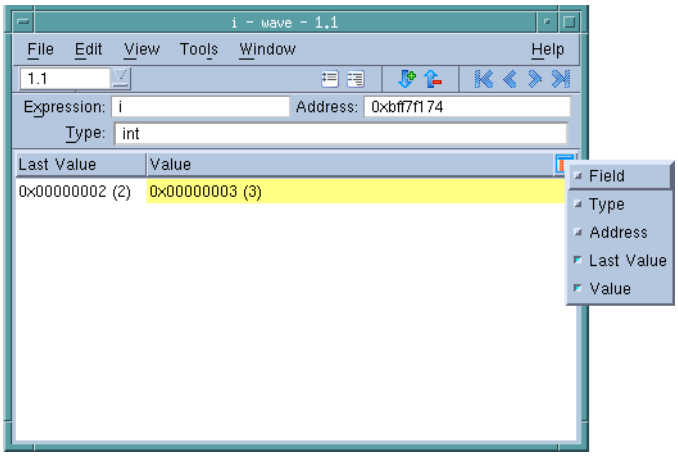
Figure 1 – Highlighted Change in the Variable Window



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

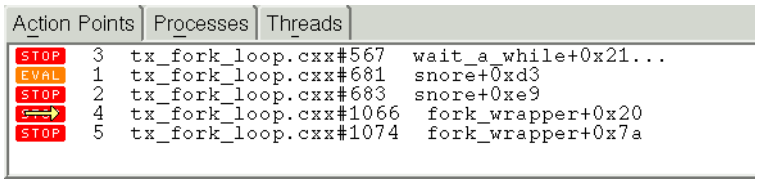
- Values in the **Variable** and **Expression List** windows have a **Previous value** hidden column that you can display. Use the control on the right side of the column headings to display a list of columns that you can display or hide.

Figure 2 – Last Value Column



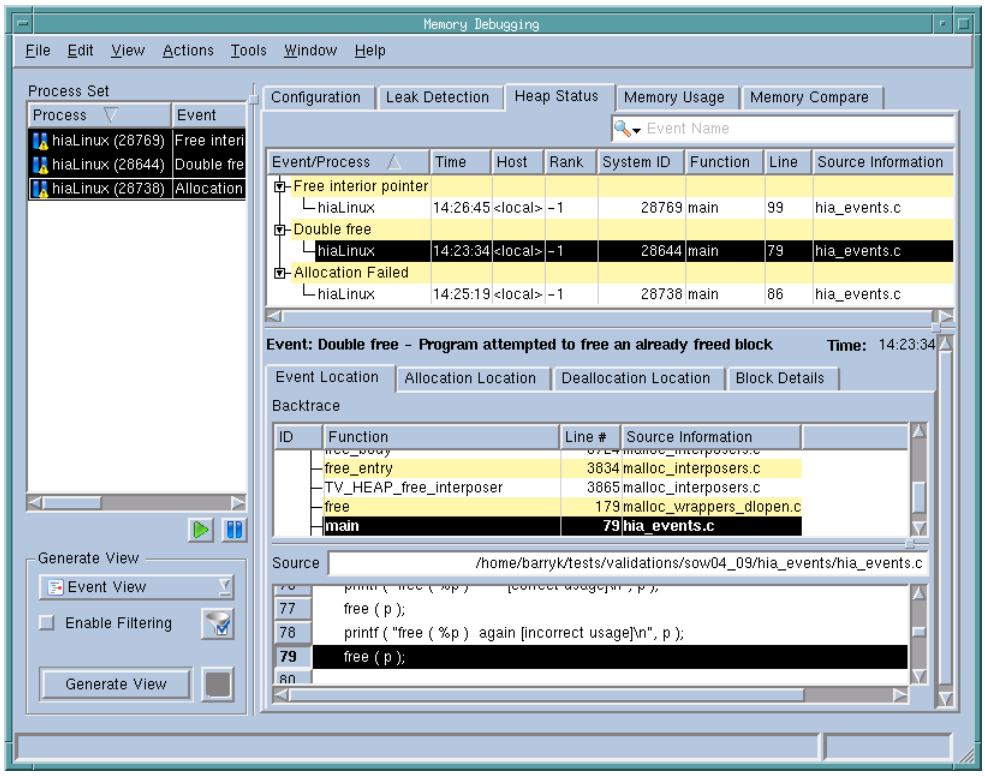
- When a process hits a breakpoint, TotalView highlights the breakpoint by placing an arrow over the breakpoint ID in the Action Points pane.

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.

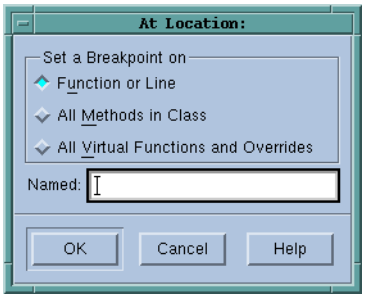
Versions 8.0 and 8.1 Features

Breakpoint Changes

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

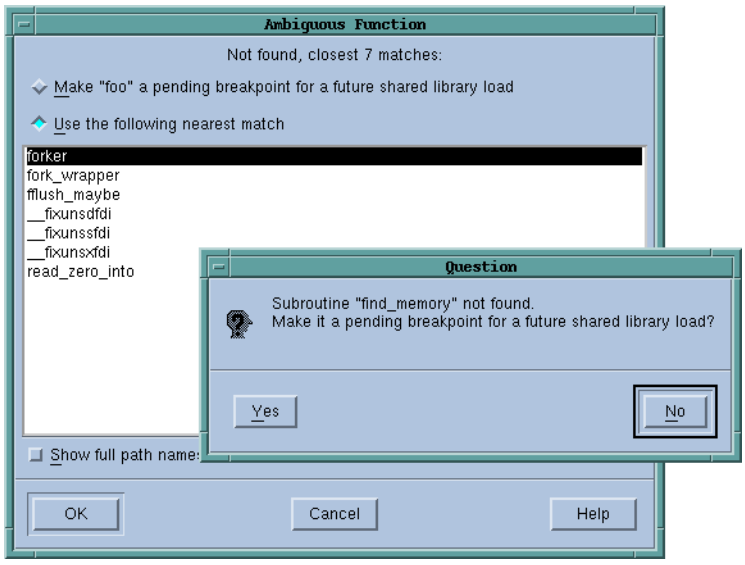
- The **Action Point > At Location** command now has three choices:
- **Function or Line**: lets you set a line number or a function name. This choice is what occurred in previous TotalView releases.
- **All Methods in Class**: lets you set a breakpoint on all methods in a class. This can set more than 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.

Figure 5 – Breakpoint > At Location Dialog Box



- 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*.

Figure 6 – Setting Pending Breakpoints



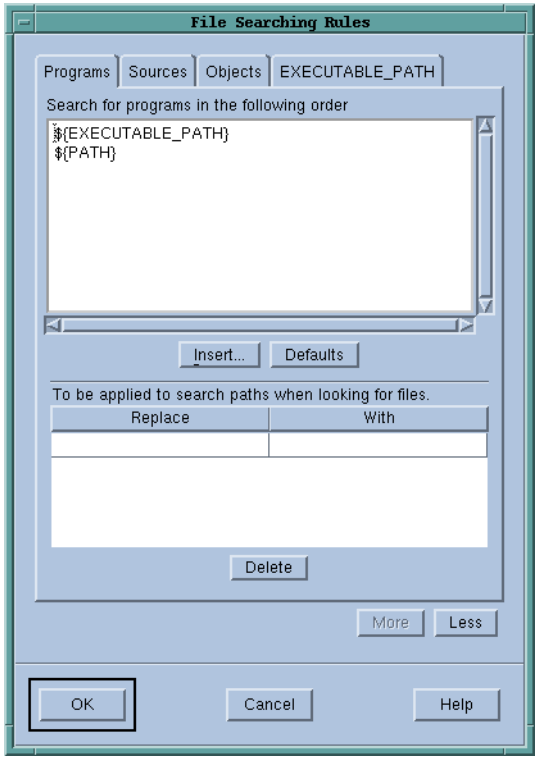
- When you create a barrier breakpoint or change a breakpoint to a barrier point, the same new features are available.
- The CLI **dbreak**, **dbarrier**, and **dlist** commands have been extended to use these features. The argument to these commands can now be a breakpoint expression. Understanding 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.

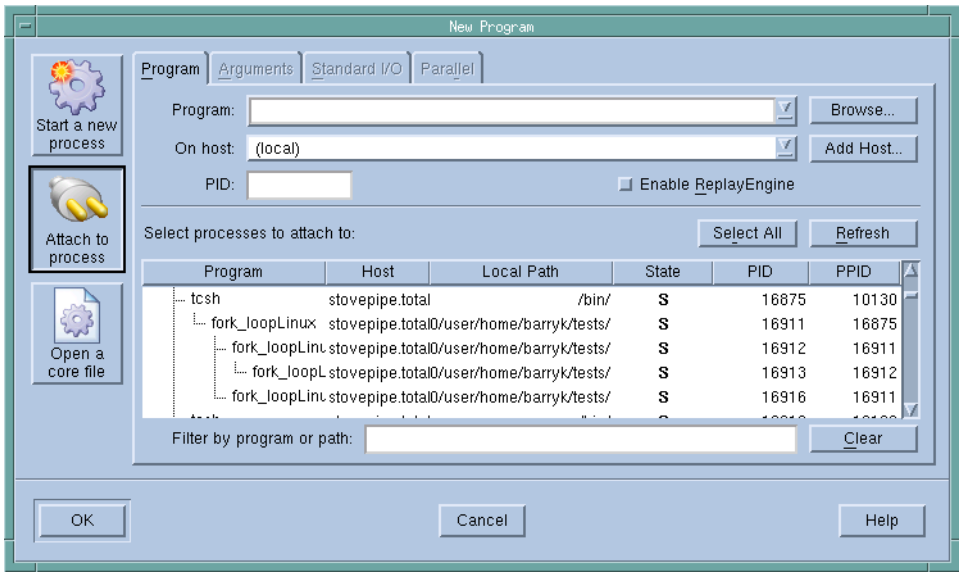
Figure 7 – File > Search Path Dialog Box



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 8 – 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

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
	Linux x86-64
	Linux Power 32
	Linux Power 64 / Cell
	SiCortex
	Cray XT
Linux x86	Linux x86
	Linux Power 32
	Linux Power 64 / Cell
Linux Power 64 (including Linux Cell)	Linux Power 32
	Linux Power 64 / Cell
	Blue 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
- 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 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 Replay-Engine 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.