Troubleshooting complex codes withTotalView

(now with a new UI)

RogueWave SOFTWARE

Accelerating Great Code

Agenda

- Introduction to Rogue Wave
- TotalView Overview
- Debugging is challenging – how does TotalView help?
  - Getting Started
  - Navigating your debugging session
  - Viewing your data
  - Parallel debugging adds complexity
  - Tricky memory issues
  - Batch can be helpful
- New TotalView UI
- Q&A

Introduction to Rogue Wave

What we do

Rogue Wave helps organizations **simplify** complex software development, **improve** code quality, and **shorten** cycle times

Company snapshot

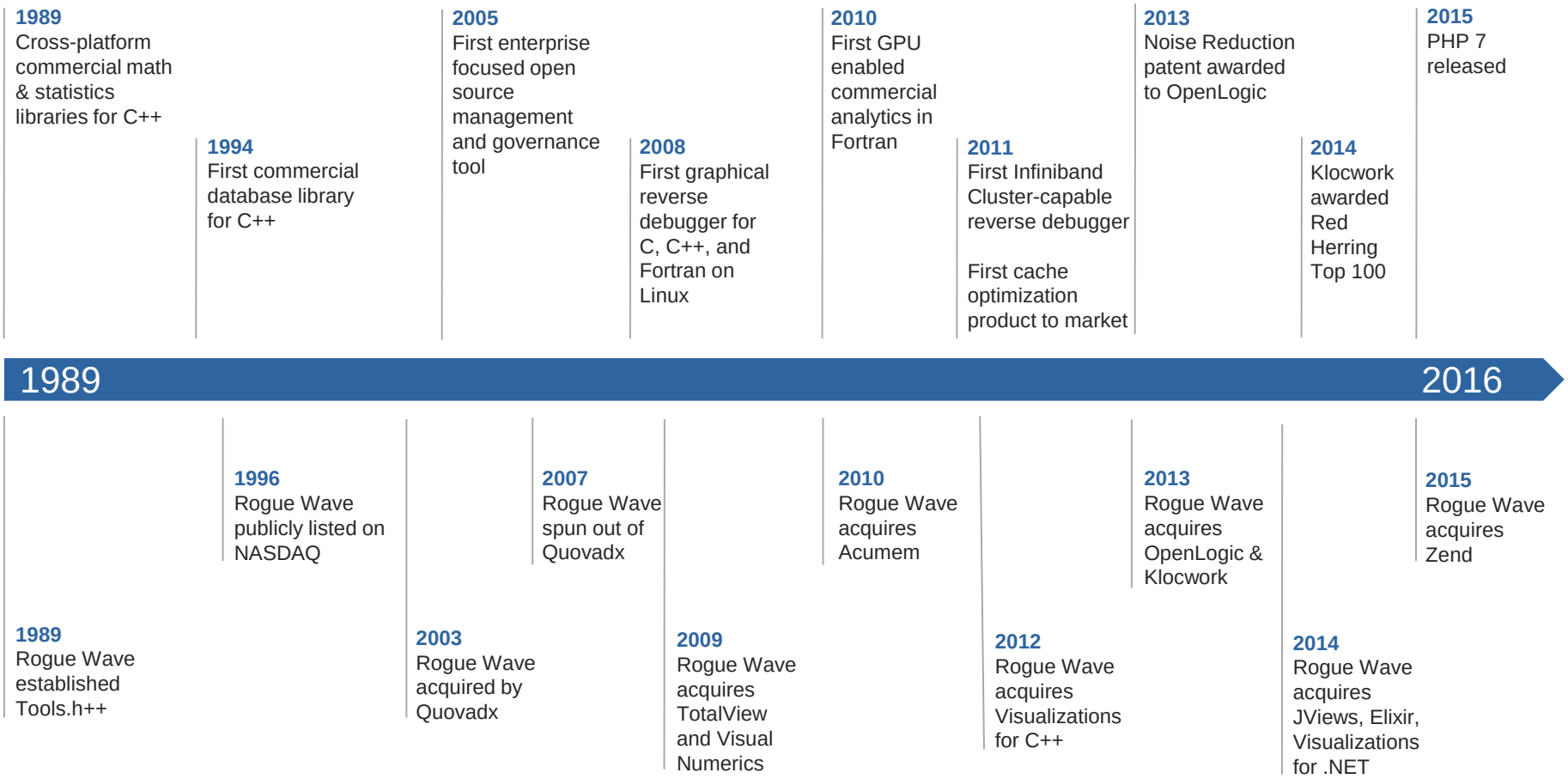
We are the largest independent provider of cross-platform software development tools and embedded components

Founded:	Headquarters:	Employees:	Offices Worldwide:
1989	Louisville, CO	350	11

Our capabilities cover different languages, code bases, and platforms. We meet development where – and how – it happens.

Company timeline

Technology Timeline



Corporate Timeline

Our products

Tools

Klocwork On-the-fly static code analysis for app security

CodeDynamics Commercial dynamic analysis

Open Logic Support Enterprise-grade SLA support for 850+ packages

OpenLogic Audits Detailed open source license and security risk guidance

TotalView for HPC Scalable parallel debugging

Zend Server Enterprise PHP app server

Zend Studio PHP IDE

Zend Guard PHP encoding and obfuscation

Libraries

SourcePro OS, database, network, and analysis abstraction for C++

Visualization Real-time data visualization at scale

PV-WAVE Visual data analysis

IMSL Numerical Libraries Scalable math and statistics algorithms

HydraExpress SOA/C++ modernization framework

HostAccess Terminal emulation for Windows

Stingray MFC GUI components

We enable mission-critical workloads

Used by 3,000 customers in over 57 countries across diverse industries to develop mission-critical applications and software

Financial Services

Bank of America, citibank, Credit Suisse, Liberty Mutual, Goldman Sachs, QUESTRADE, Bloomberg, PRUDENTIAL, Morgan Stanley, UBS, BARCLAYS, MIZUHO, E*TRADE, Deutsche Bank, JPMorganChase

Telecom

at&t, HUAWEI, BT, verizon, Alcatel·Lucent, GenCorp, Sprint, MOTOROLA, Polycom, QUALCOMM, ERICSSON, L3, BlackBerry

Gov't / Defense

BOEING, NASA, NORTHROP GRUMMAN, BAE SYSTEMS, Raytheon, Lawrence Livermore National Laboratory, LOCKHEED MARTIN, Sandia National Laboratories

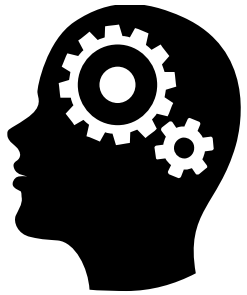
Technology

cisco, ca, ORACLE, IBM, bmcsoftware, DELL, EMC², RSA SECURITY, hp, SUNGARD, fiserv., SAP

Other Verticals

PIXAR, ExxonMobil, McKESSON, Schlumberger, Disney, DELPHI, Chevron, INDUSTRIAL LIGHT & MAGIC, Shell, Cerner, OLYMPUS, SIEMENS, MARATHON, ارامكو السعودية Saudi Aramco, DreamWorks, Schneider Electric, TOTAL, JOHN DEERE

Our vision



Total commitment to the highest quality code

Leveraging the value of code across multiple platforms

Secure code that meets compliance and regulatory standards

Accelerating cycle times

Embracing change and open source in the enterprise

Simplifying the complex – solving industry challenges

How we see software today

Blending languages, commercial, open source, and custom software and platforms is the norm

Demands for faster, better code in enterprise, embedded and web development rises every day

The mix of languages, components, frameworks, and governance add to the **complexity** of building mission-critical, enterprise systems

Software **development and deployment is fragmented** – tools, languages, frameworks, and environments

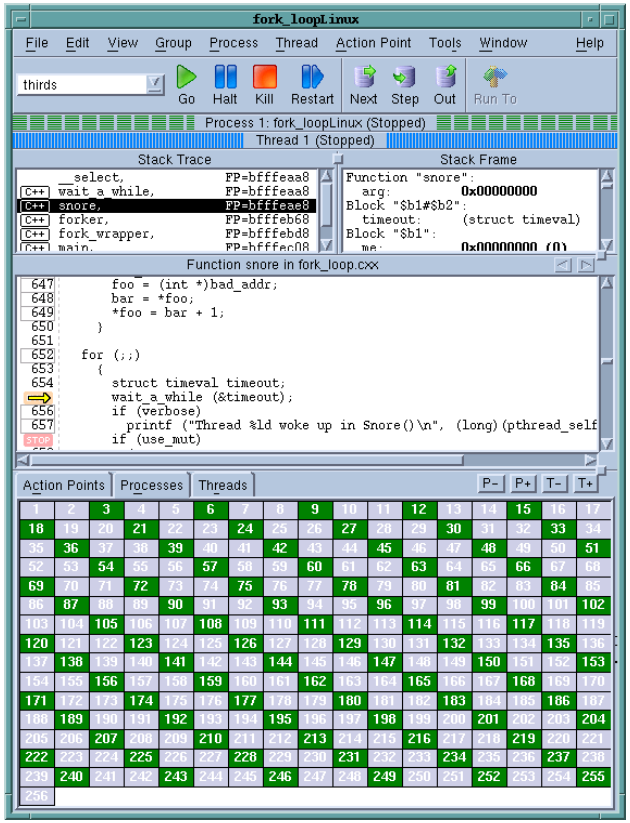
Developers are empowered to use the tools they want

TotalView Overview

What is TotalView®?

Application Analysis and Debugging Tool: Code Confidently

- Debug and Analyse C/C++ and Fortran on Linux™, Unix or Mac OS X
- Laptops to supercomputers
- Makes developing, maintaining, and supporting critical apps easier and less risky



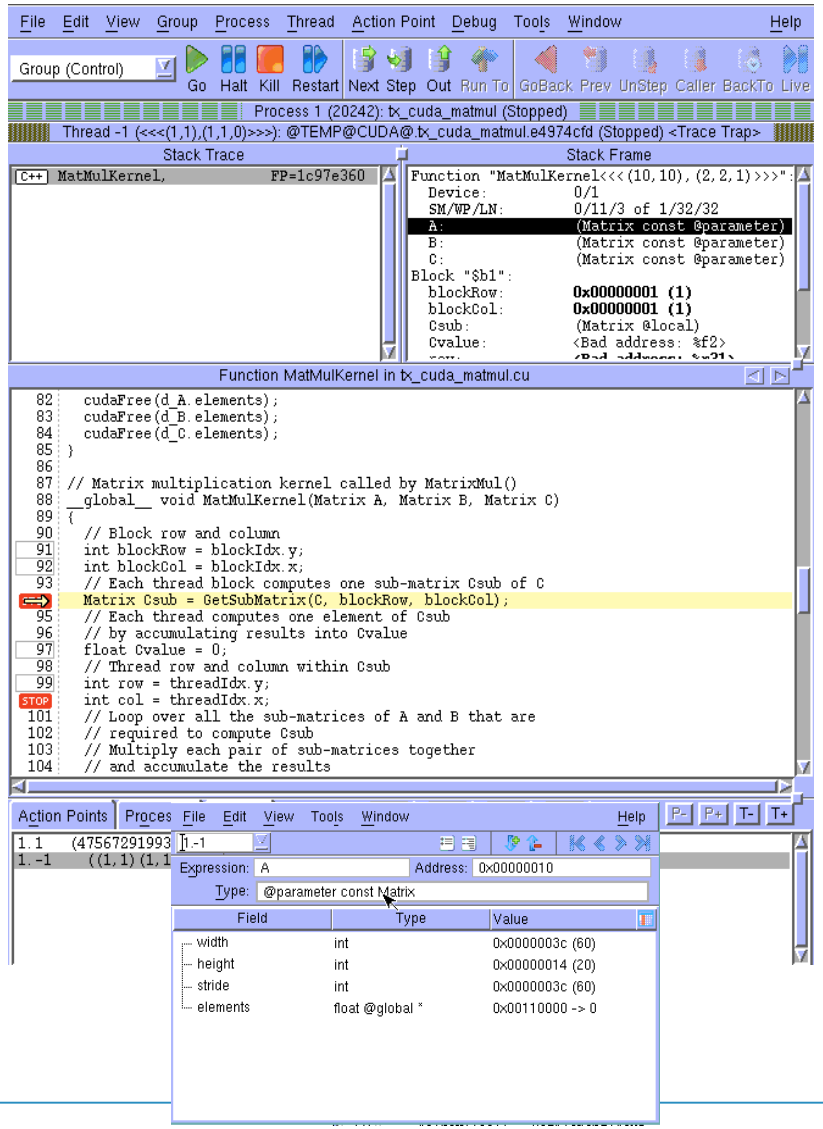
Major Features

- Easy to learn graphical user interface with data visualization
- Parallel Debugging
  - MPI, Pthreads, OpenMP™
  - CUDA™, OpenACC®, and Intel® Xeon Phi™ coprocessor
- Low tool overhead resource usage
- Includes a Remote Display Client which frees you to work from anywhere
- Memory Debugging with MemoryScape™
- Deterministic Replay Capability Included on Linux/x86-64
- Non-interactive Batch Debugging with TVScript and the CLI
- TTF & C++View to transform user defined objects

TotalView for the NVIDIA® GPU Accelerator

- How do you debug software running on CUDA today?
- It is hard because of its specific architecture

- TotalView supports the latest and greatest
- CUDA 6.0, 6.5, 7.0
- Cray CCE OpenACC
- Features and capabilities include
  - MPI-based clusters
  - Multi-card configurations
  - Flexible Display and Navigation
    - Physical (device, SM, Warp, Lane)
    - Logical (Grid, Block) tuples
  - Shows what is running where
  - Support for separate memory address spaces
  - Leverages CUDA memcheck
  - **Supports CUDA dynamic parallelism**



TotalView for Intel® Xeon Phi™

Supports all major Intel Xeon Phi Coprocessor configurations

- Native Mode
  - With or without MPI
- Offload Directives
  - Incremental adoption, similar to GPU
- Symmetric Mode
  - Host and Coprocessor
- Multi-device, Multi-node
- Clusters

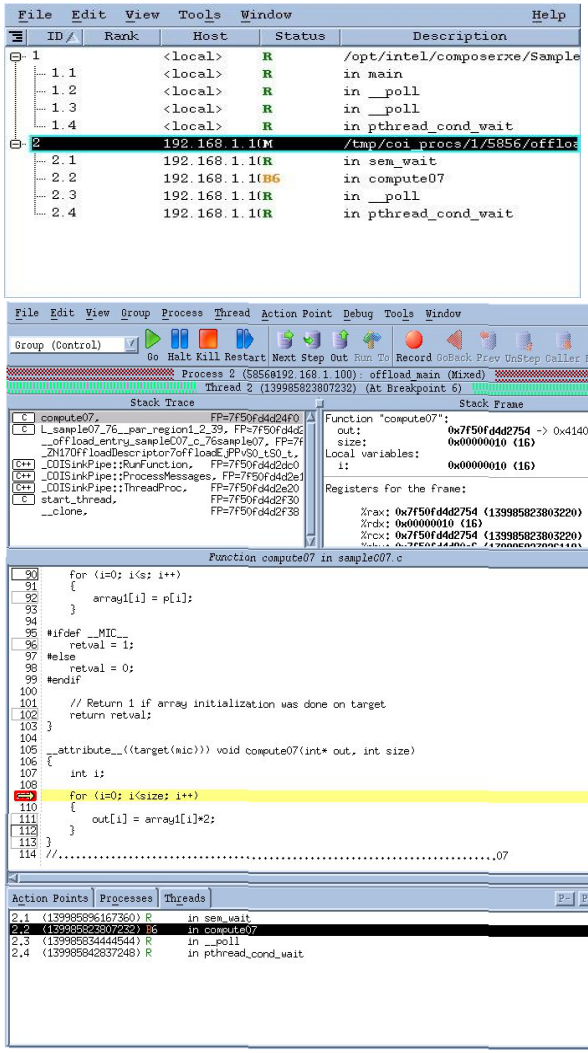
User Interface

- MPI Debugging Features
  - Process Control, View Across, Shared Breakpoints
- Heterogeneous Debugging
  - Debug Both Xeon and Intel Xeon Phi Processes

Memory Debugging

- Both native and symmetric mode

Set up well for supporting KNL when it is available

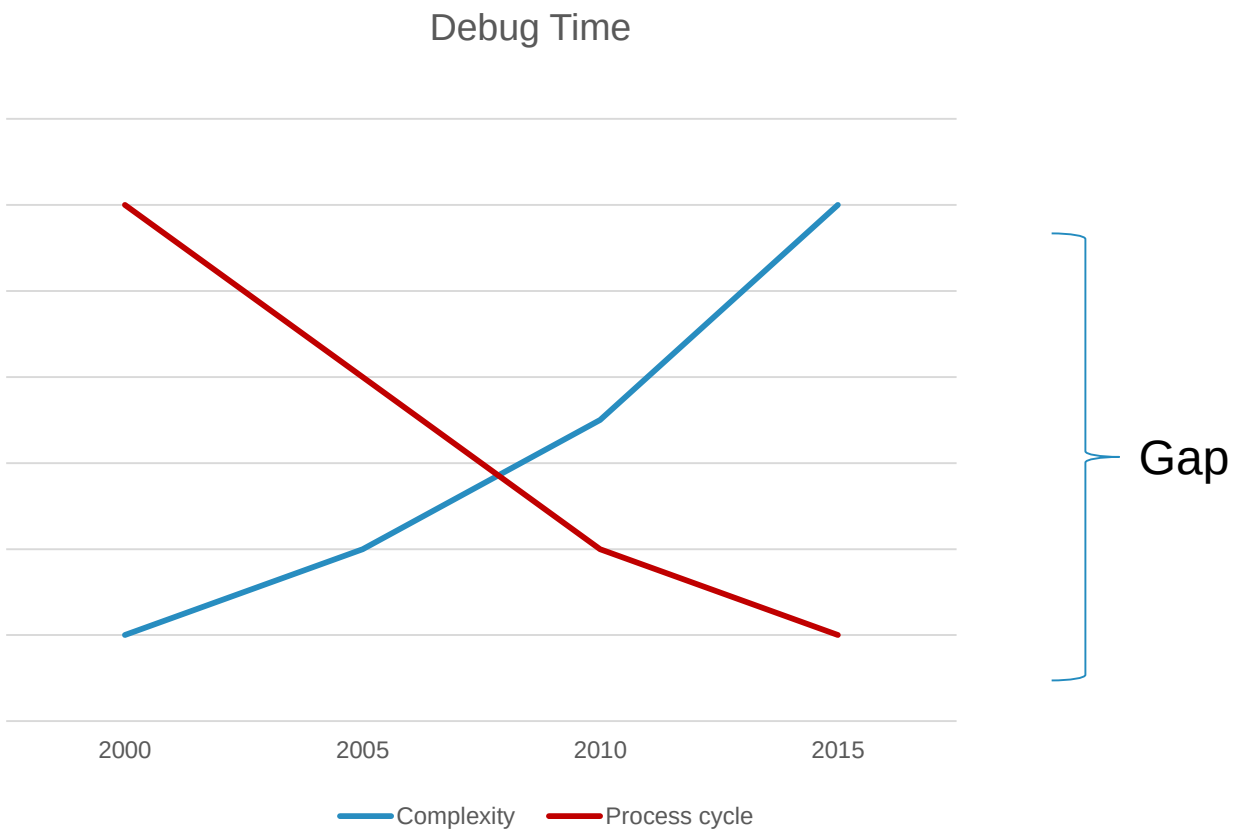


Debugging is challenging

How does TotalView help?

What is the problem?

- Complexity
  - Threading, Cores
  - Data types
  - LOC
  - Memory Usage
- Iteration time
  - Growth in agile



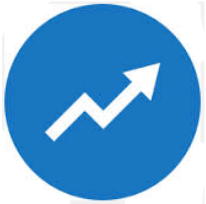
Gap needs closure or majority of development spent fixing defects

Defect analysis is a bottleneck

- Code Correctness, Bug troubleshooting, Memory Analysis
  - Some defects can can weeks to analyze
- Techniques include
    - Logging, Print statements, Debuggers
- Bugs are costly
  - The pressure is high
  - Releases slip
  - Customers get frustrated

The solution

Reduce complexity



Increase productivity

Address new technologies

How does TotalView help?

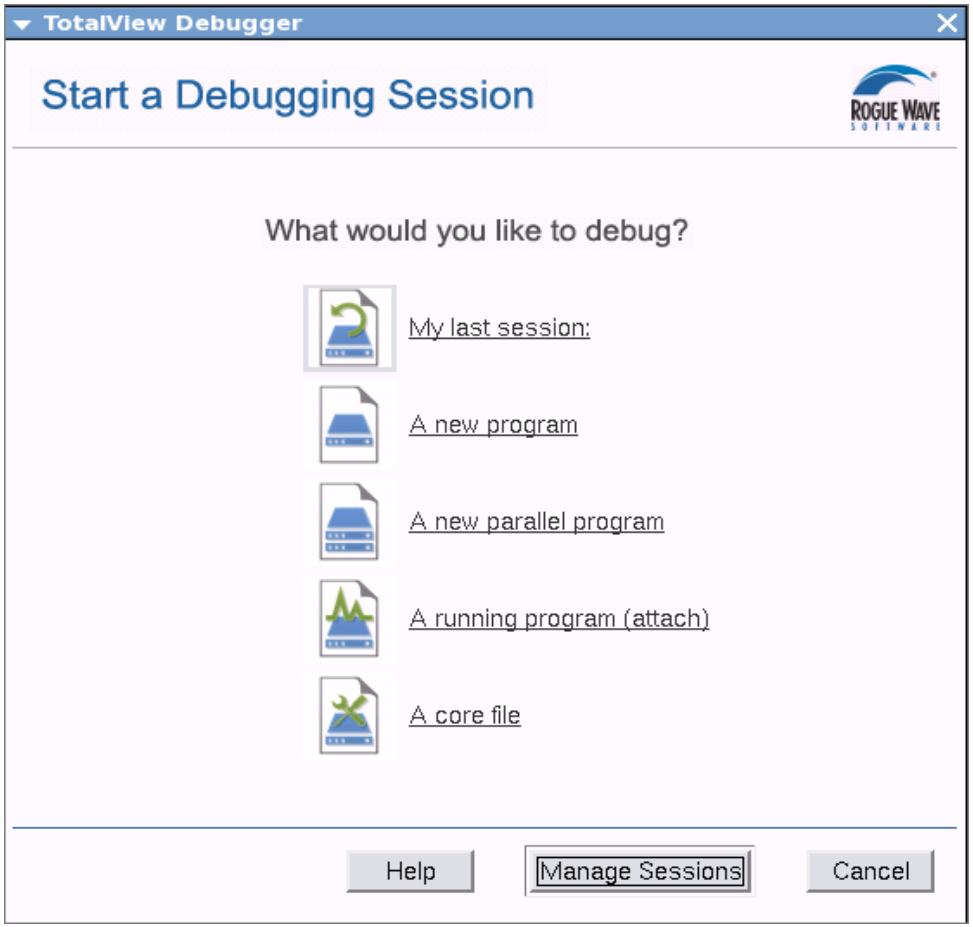
TotalView debugger

- Troubleshooting and analysis tool
  - Visibility into applications
  - Control over applications
- Scalability
- Usability
- Support for HPC platforms and languages

Getting Started

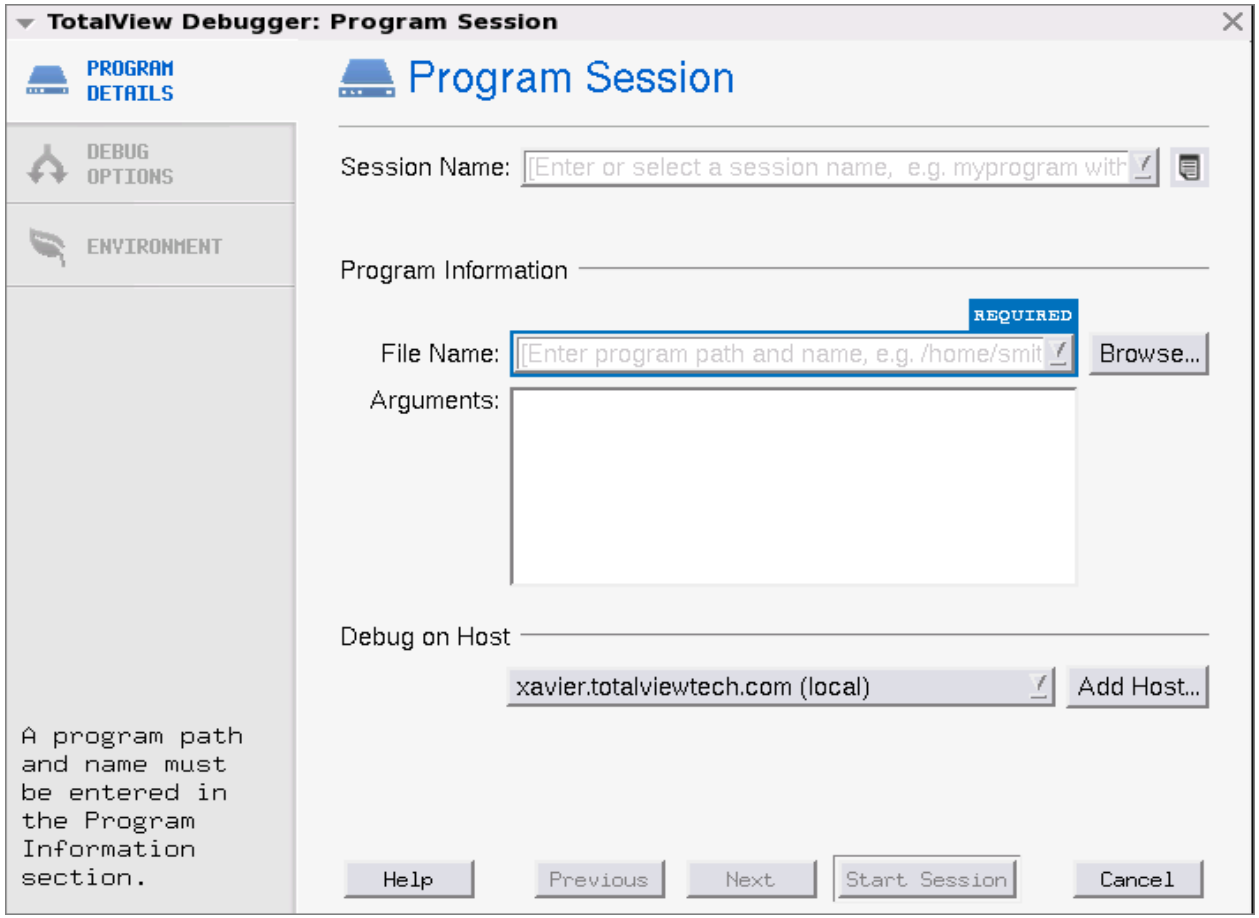
Starting TotalView

Sessions Manager



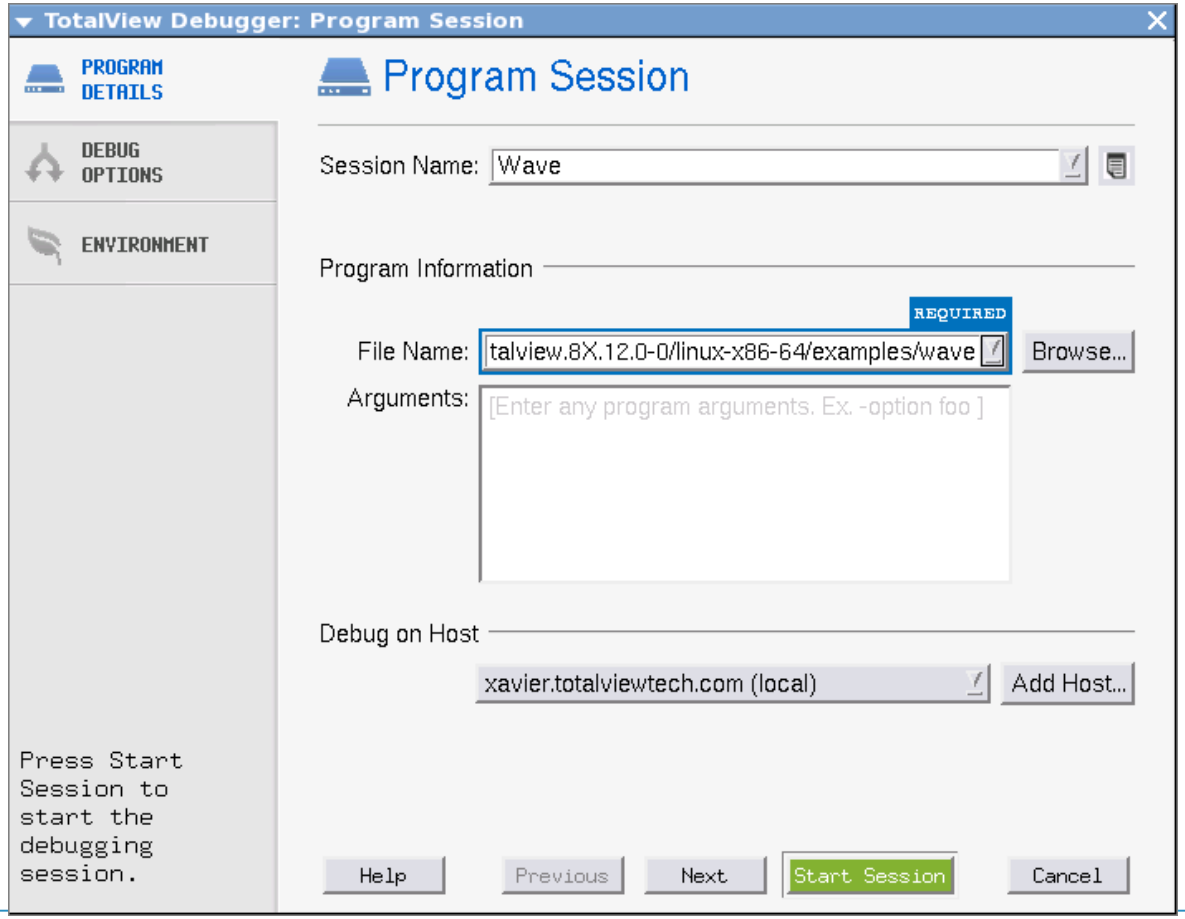
Starting TotalView

Debugging a new Program



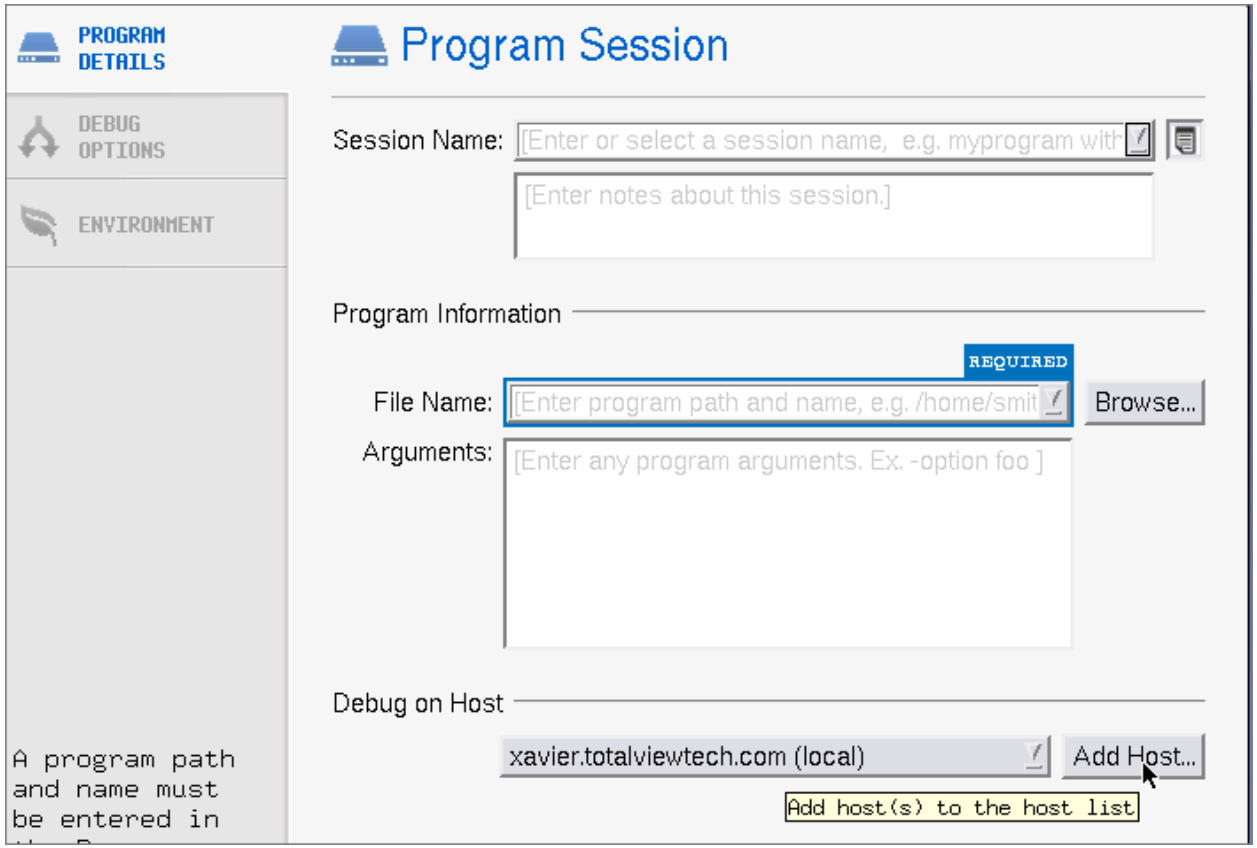
Starting TotalView

Debugging a new Program



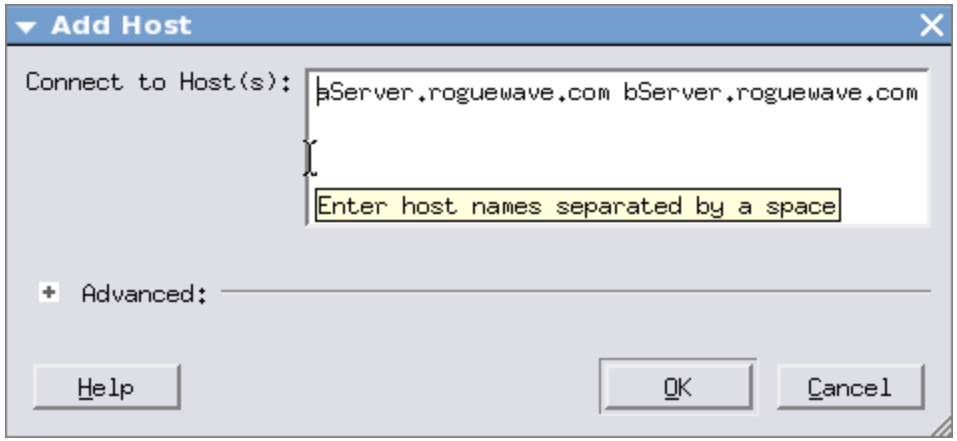
Starting TotalView

Adding a remote host



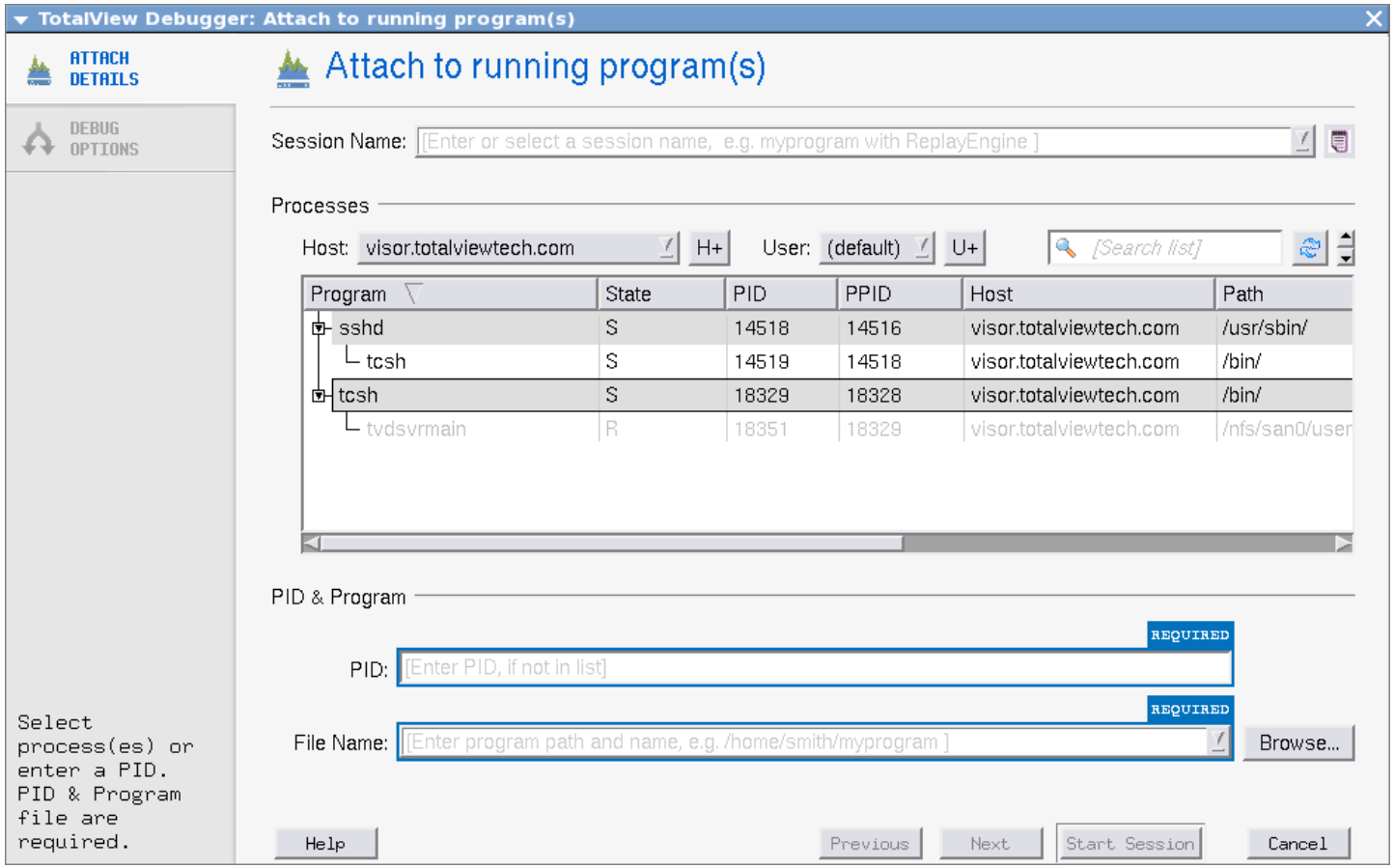
Starting TotalView

Adding a remote host



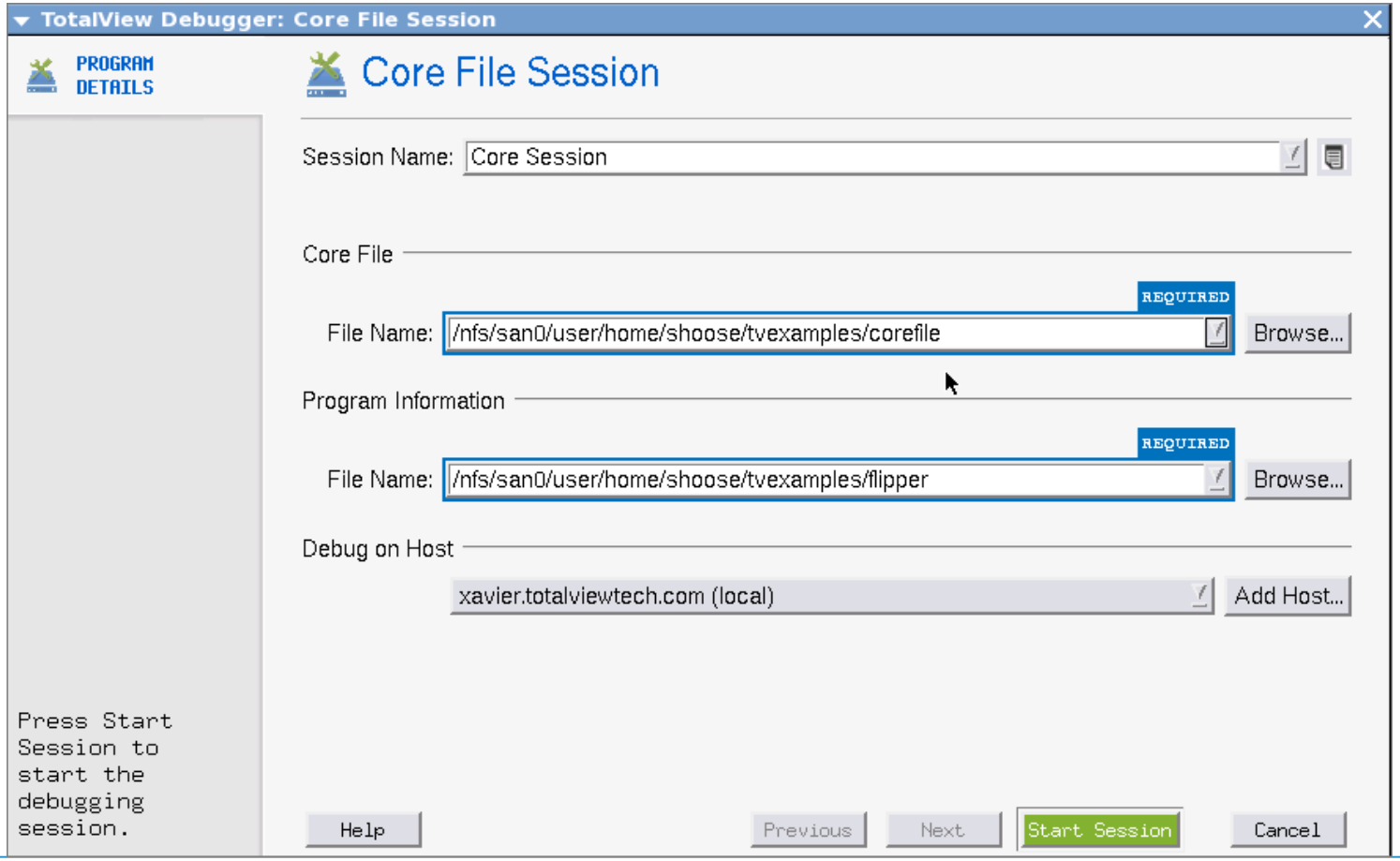
Starting TotalView

Attach to running program(s)



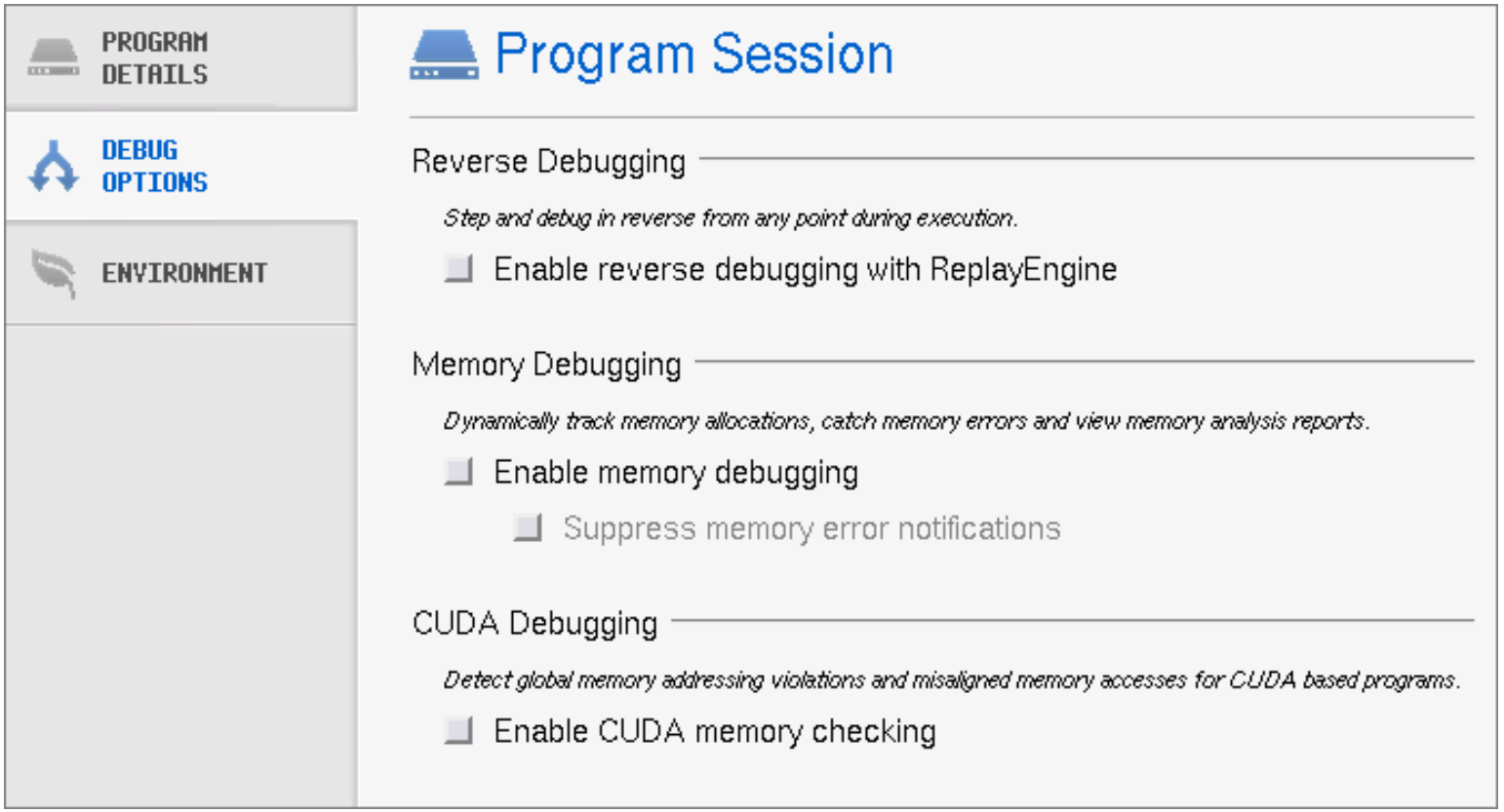
Starting TotalView

Open a Core File



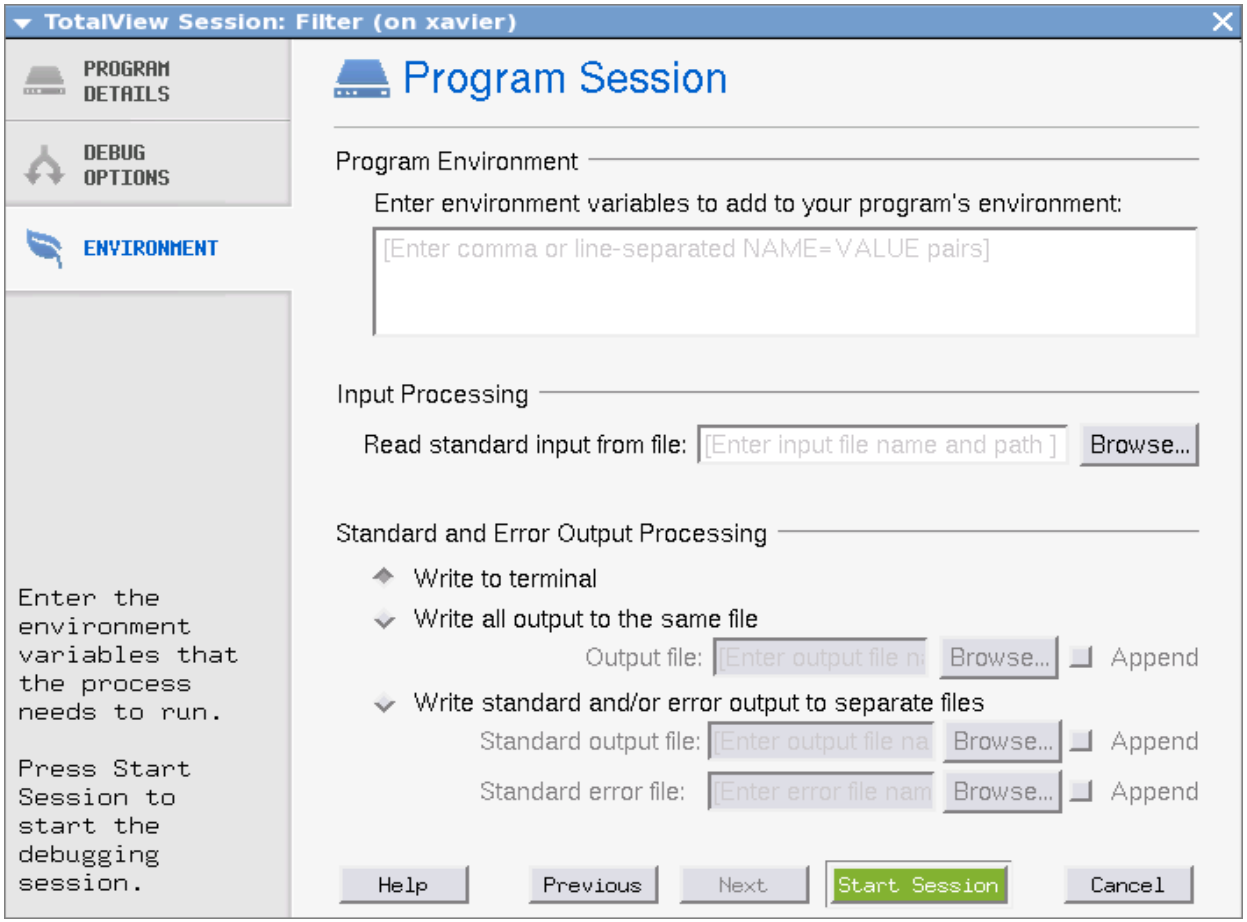
Starting TotalView

Debug options



Starting TotalView

Environment



Starting TotalView

Via Command Line

Normal

totalview [tv_args] prog_name [–a prog_args]

Attach to running program

totalview [tv_args] prog_name –pid PID# [–a prog_args]

Attach to remote process

totalview [tv_args] prog_name –remote name [–a prog_args]

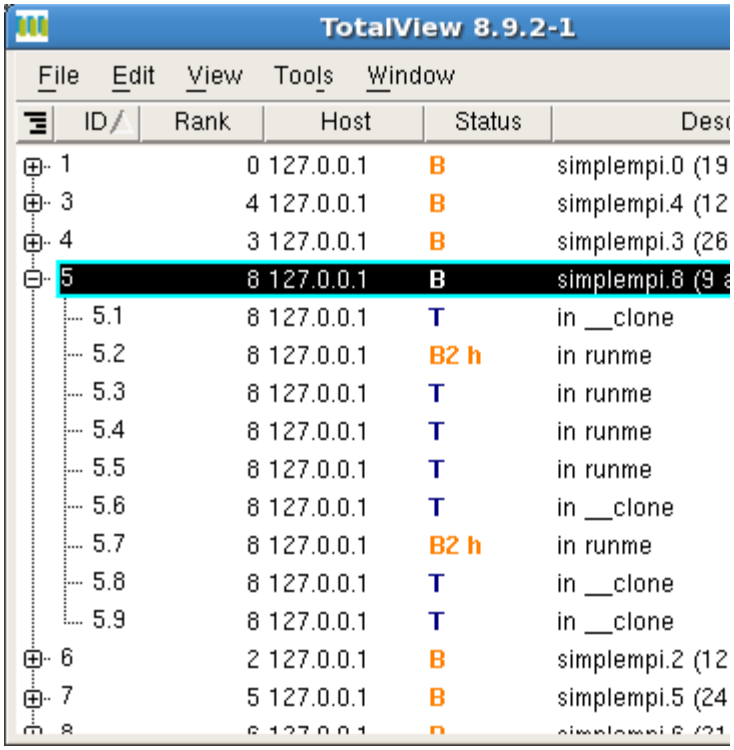
Attach to a core file

totalview [tv_args] prog_name corefile_name [–a prog_args]

Navigating your debugging session

TotalView Root Window

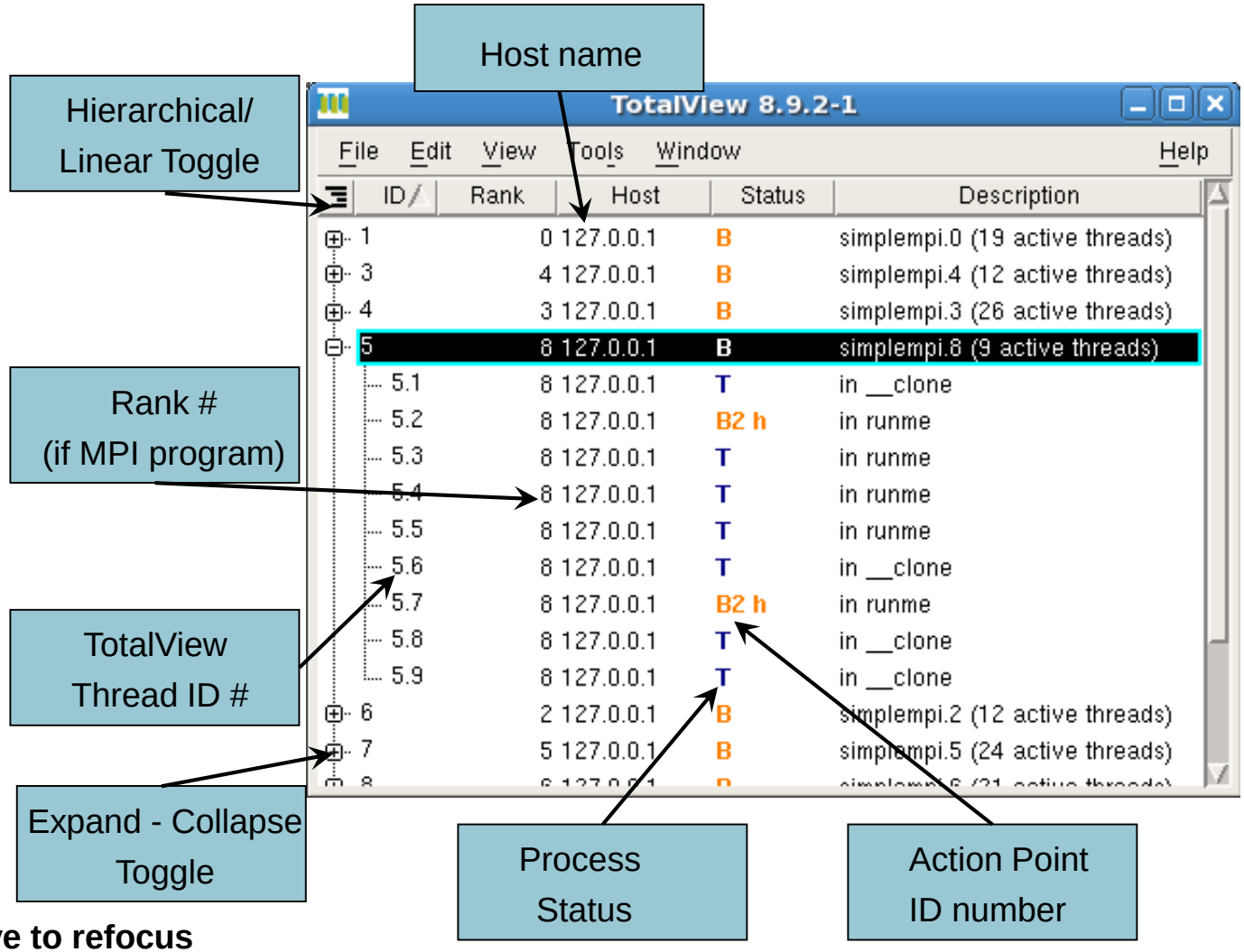
- State of all processes being debugged
- Process and Thread status
- Instant navigation access
- Sort and aggregate by status



↗Status Info

- T = stopped
- B = Breakpoint
- E = Error
- W = Watchpoint
- R = Running
- M = Mixed
- H = Held

TotalView Root Window

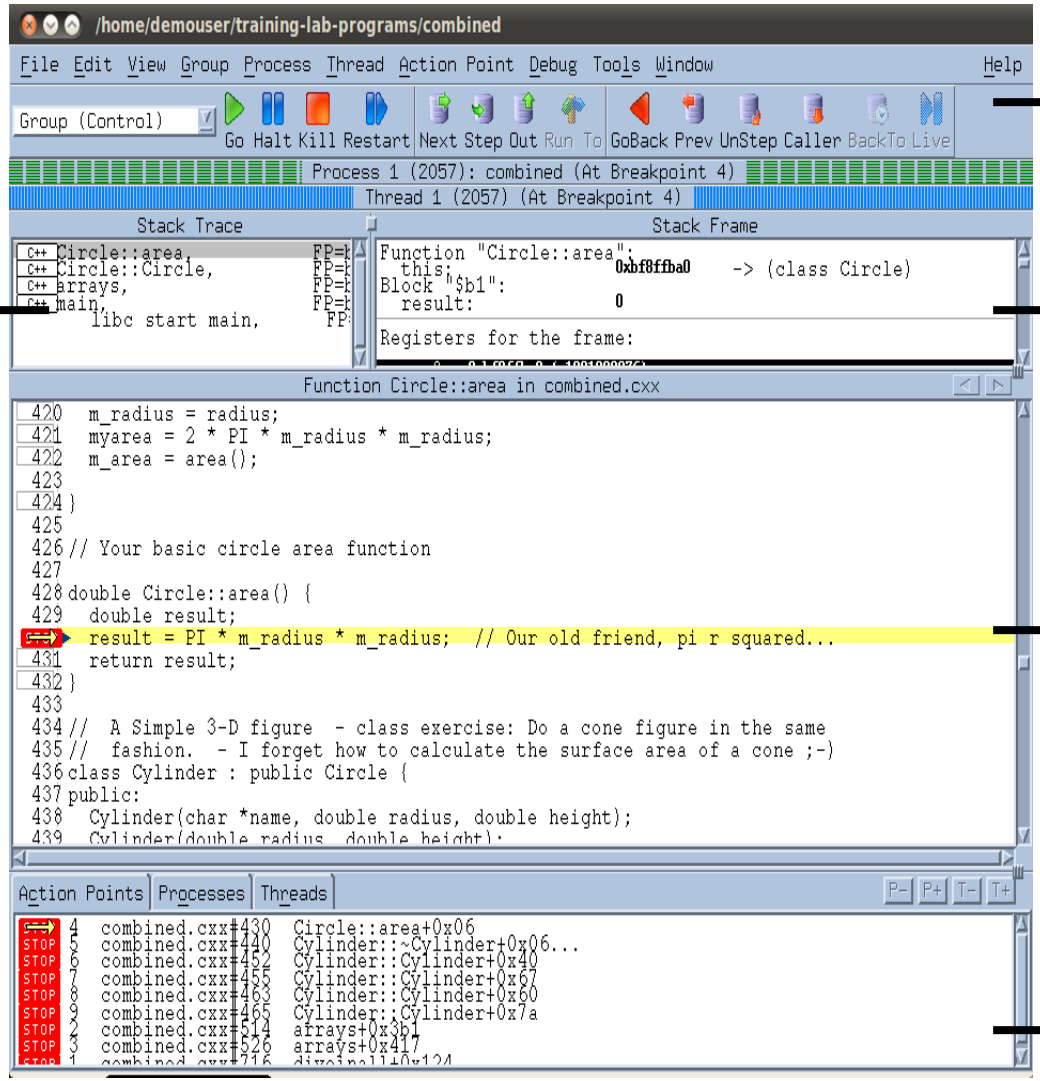


- **Dive to refocus**
- ***Dive in new window* to get a second process window**

Process Window Overview

Provides detailed state of one process, or a single thread within a process

A single point of control for the process and other related processes



Stack Trace Pane

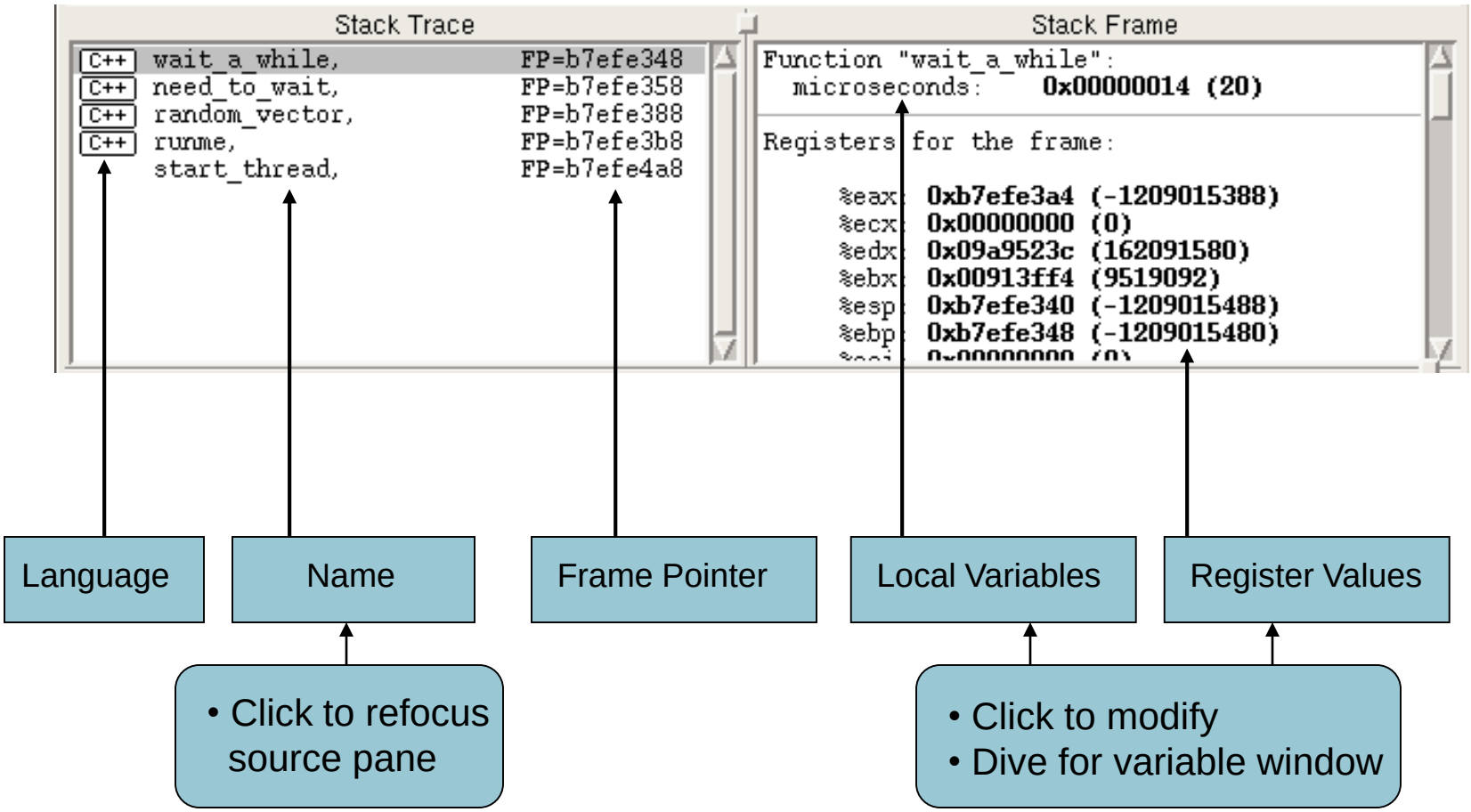
Toolbar

Stack Frame Pane

Source Pane

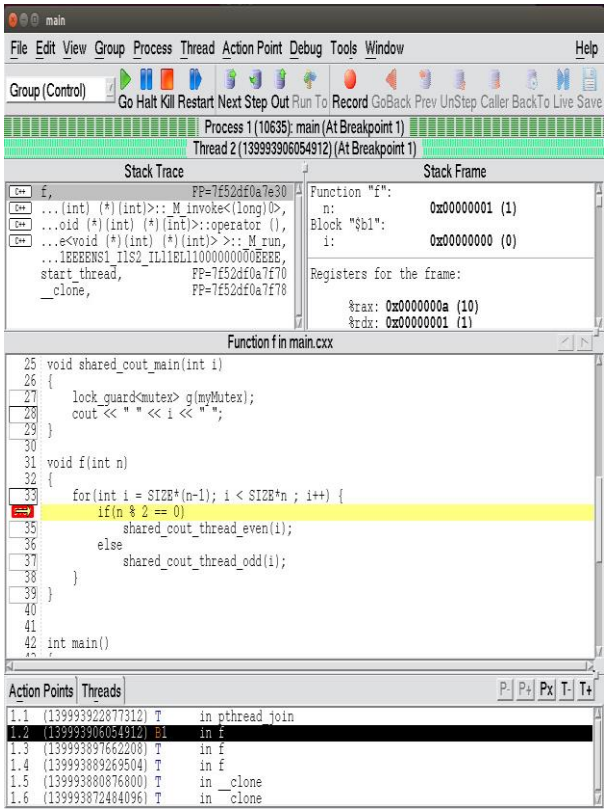
Tabbed Area

Stack Trace and Stack Frame Panes

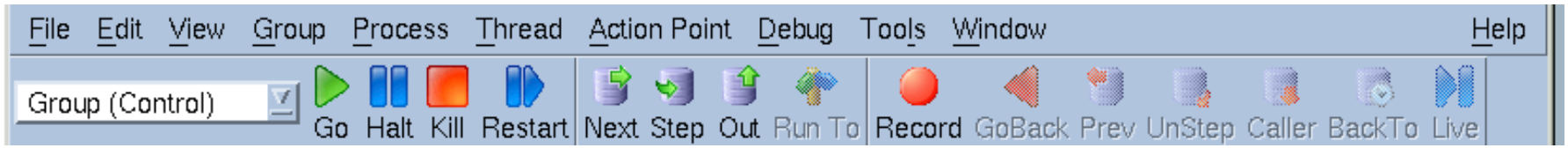


Multi-threading made easier

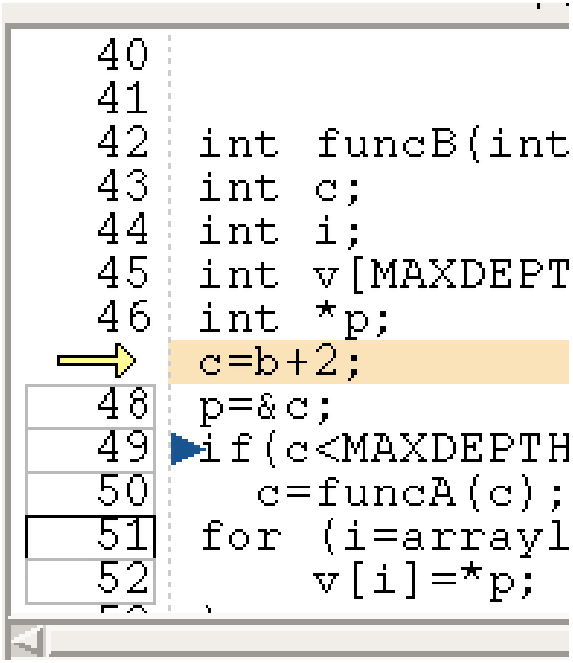
- How do you debug a problem that occurs in 1 thread in a 50 thread application?
  - Without TotalView,
    - Set a breakpoint in code
    - Run and hope you hit the right thread
  - With TotalView
    - set thread specific breakpoints
- Better multithreaded debugger
  - Understand the state of all of your threads
  - Focus on specific threads
    - View stack and data
- Built to scale to HPC and leveraging that in mainstream commercial envs



Deterministic Replay Debugging



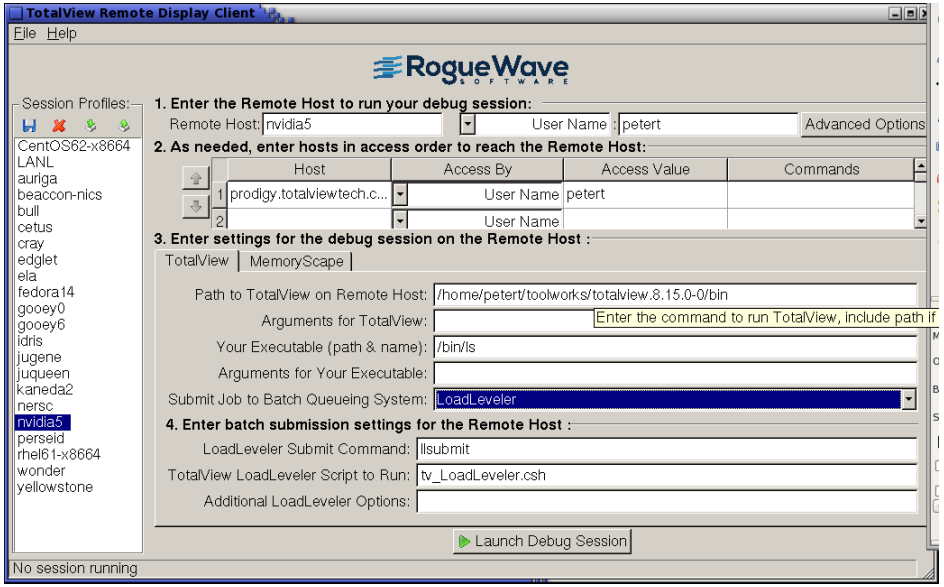
- Reverse Debugging: Radically simplify your debugging
  - Captures and Deterministically Replays Execution
    - Not just "checkpoint and restart"
  - Eliminate the Restart Cycle and Hard-to-Reproduce Bugs
  - Step Back and Forward by Function, Line, or Instruction



- Specifications
  - A feature included in TotalView on Linux x86 and x86-64
    - No recompilation or instrumentation
    - Explore data and state in the past just like in a live process, including C++View transformations
  - Replay on Demand: enable it when you want it
  - Supports MPI on Ethernet, Infiniband, Cray XE Gemini
  - Supports Pthreads, and OpenMP
  - New: Save / Load Replay Information

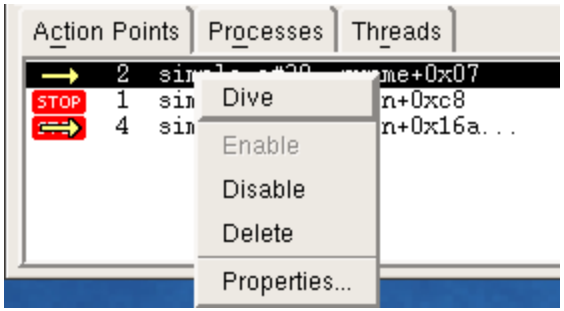
Remote Display Client

- Offers users the ability to easily set up and operate a TotalView debug session that is running on another system
- Consists of two components
  - Client – runs on local machine
  - Server – runs on any system supported by TotalView and “invisibly” manages the secure connection between host and client
- Remote Display Client is available for:
  - Linux x86, x86-64
  - Windows XP, Vista, 7
  - Mac OS X



Viewing your data

Breakpoints



Breakpoints

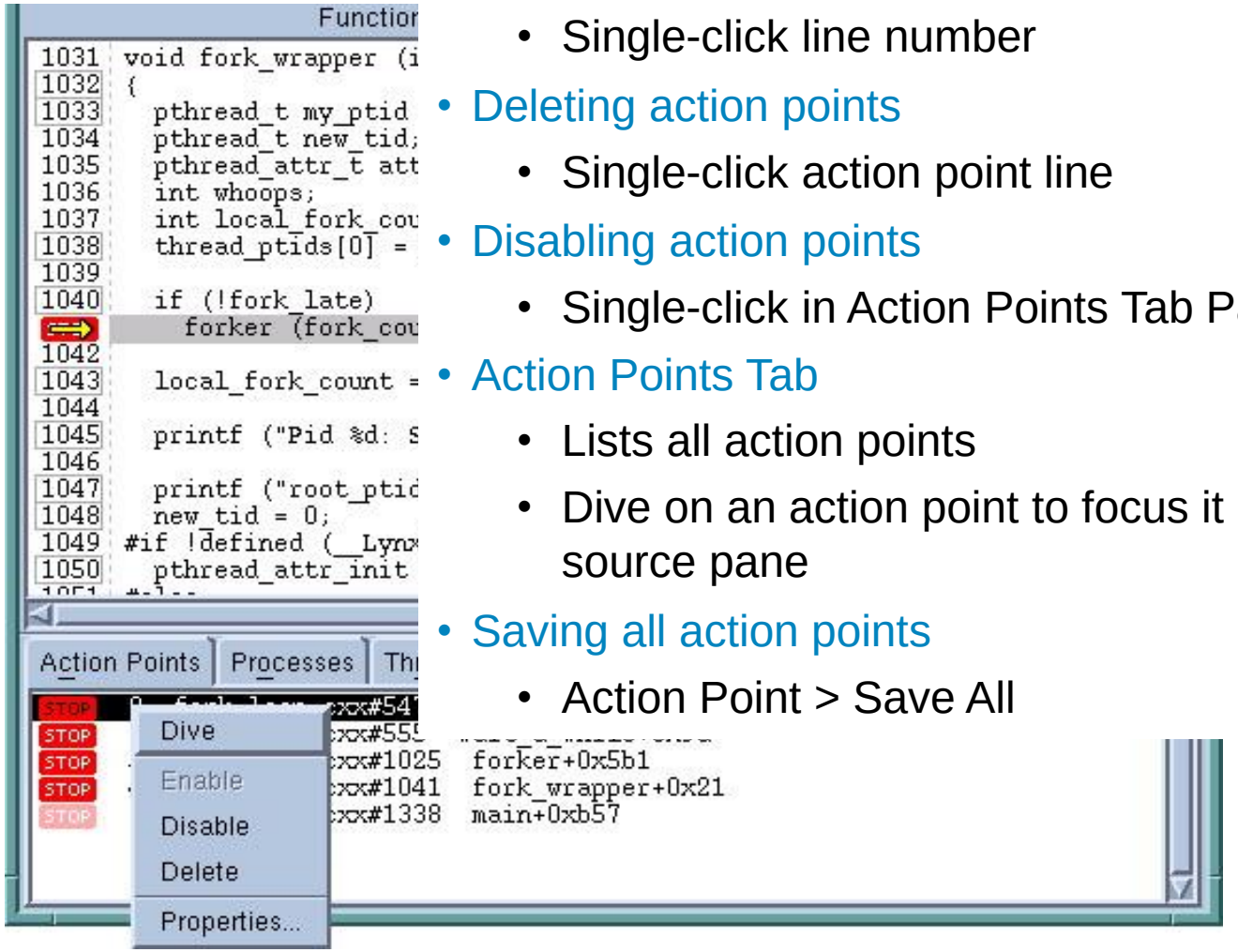
Barrier Breakpoints

Conditional Breakpoints

Evaluation Breakpoints

Watchpoints

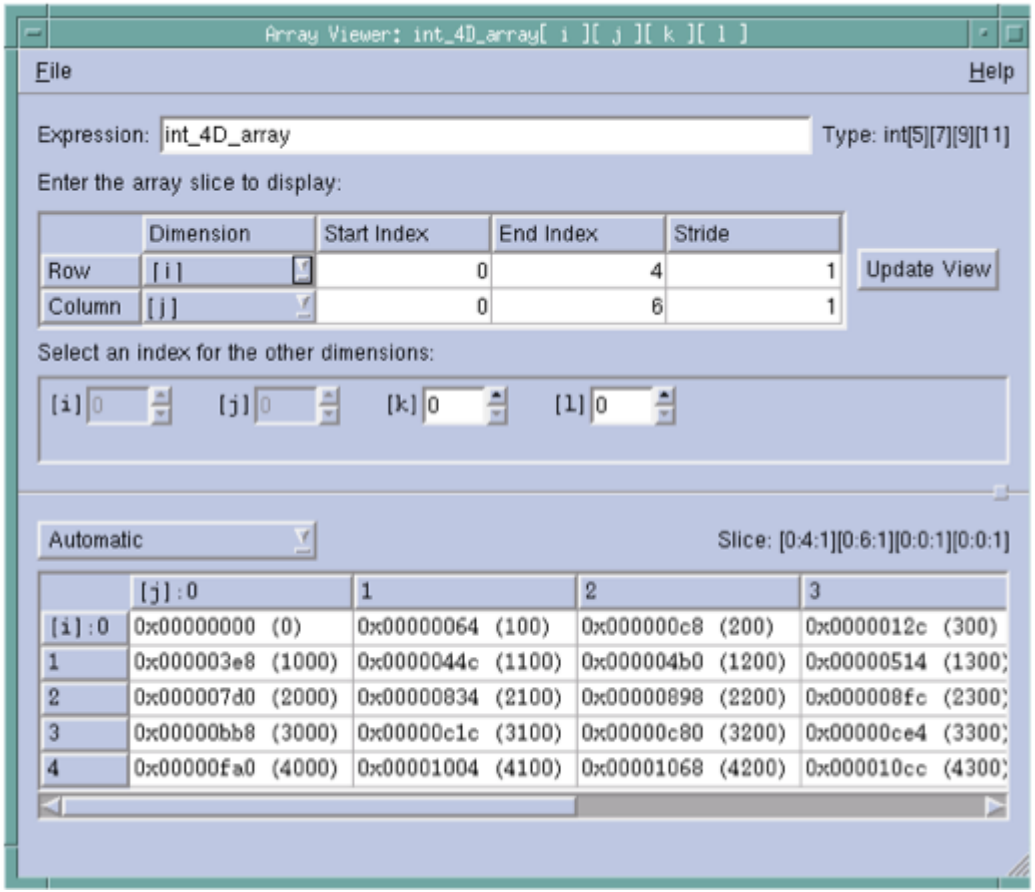
Setting Breakpoints



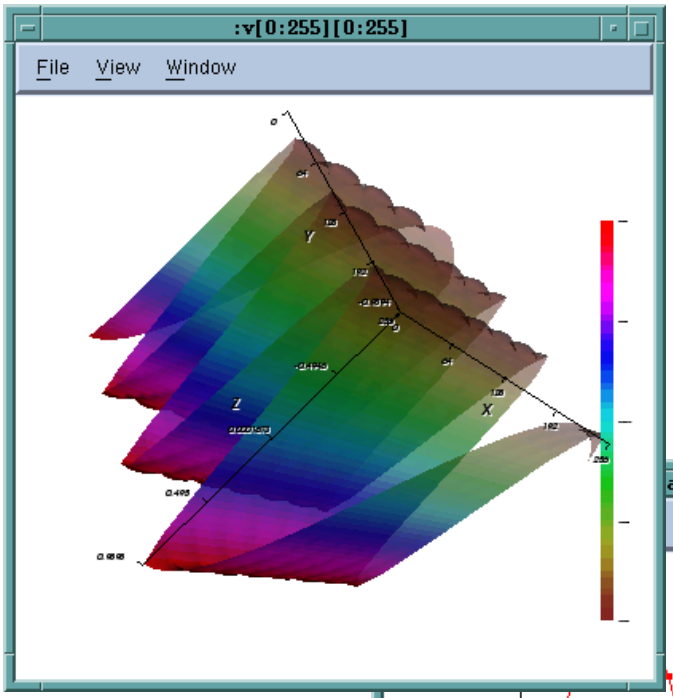
- Setting action points
  - Single-click line number
- Deleting action points
  - Single-click action point line
- Disabling action points
  - Single-click in Action Points Tab Pane
- Action Points Tab
  - Lists all action points
  - Dive on an action point to focus it in source pane
- Saving all action points
  - Action Point > Save All

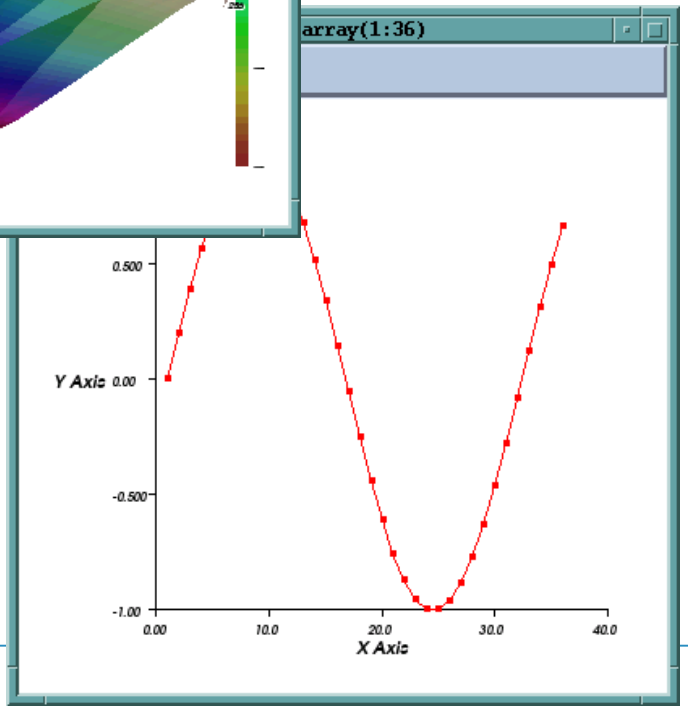
Multi-dimensional array viewer

- See your arrays on a Grid display
- 2-D, 3-D, …N-D
- Arbitrary slices
- Specify data representation
- Windowed data access – Fast



Visualizing Arrays



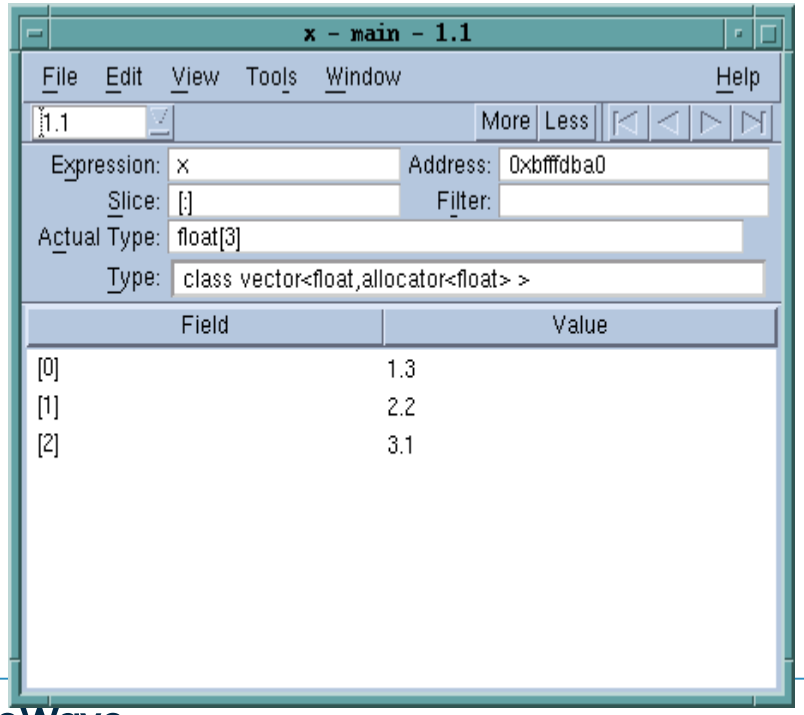


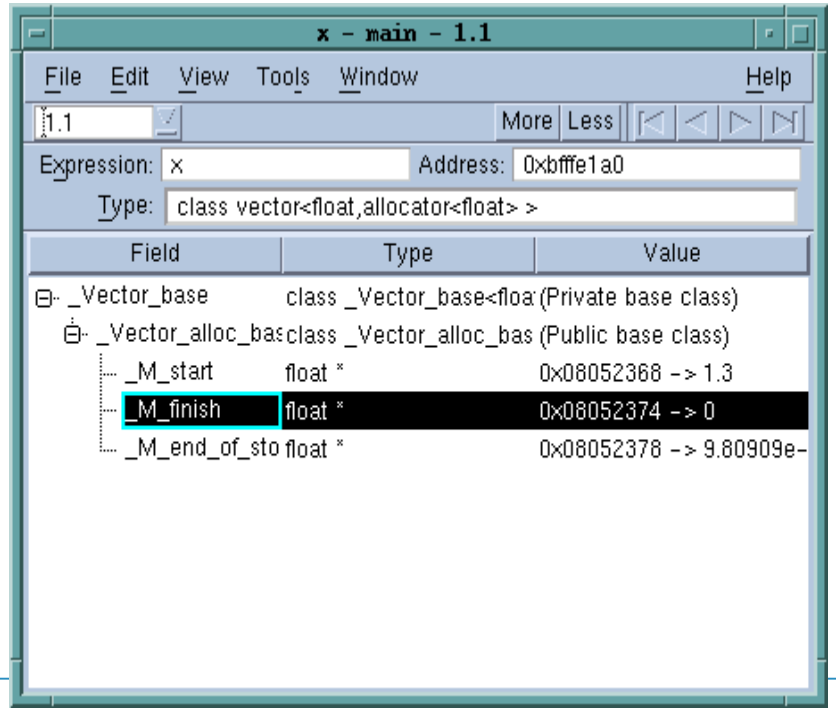
- Visualize array data using Tools > Visualize from the Variable Window
- Large arrays can be sliced down to a reasonable size first
- Visualize is a standalone program
- Data can be piped out to other visualization tools
- Visualize allows to spin, zoom, etc.
- Data is not updated with Variable Window; You must revisualize
- $visualize() is a directive in the expression system, and can be used in evaluation point expressions.

STLView

STLView transforms templates into readable and understandable information

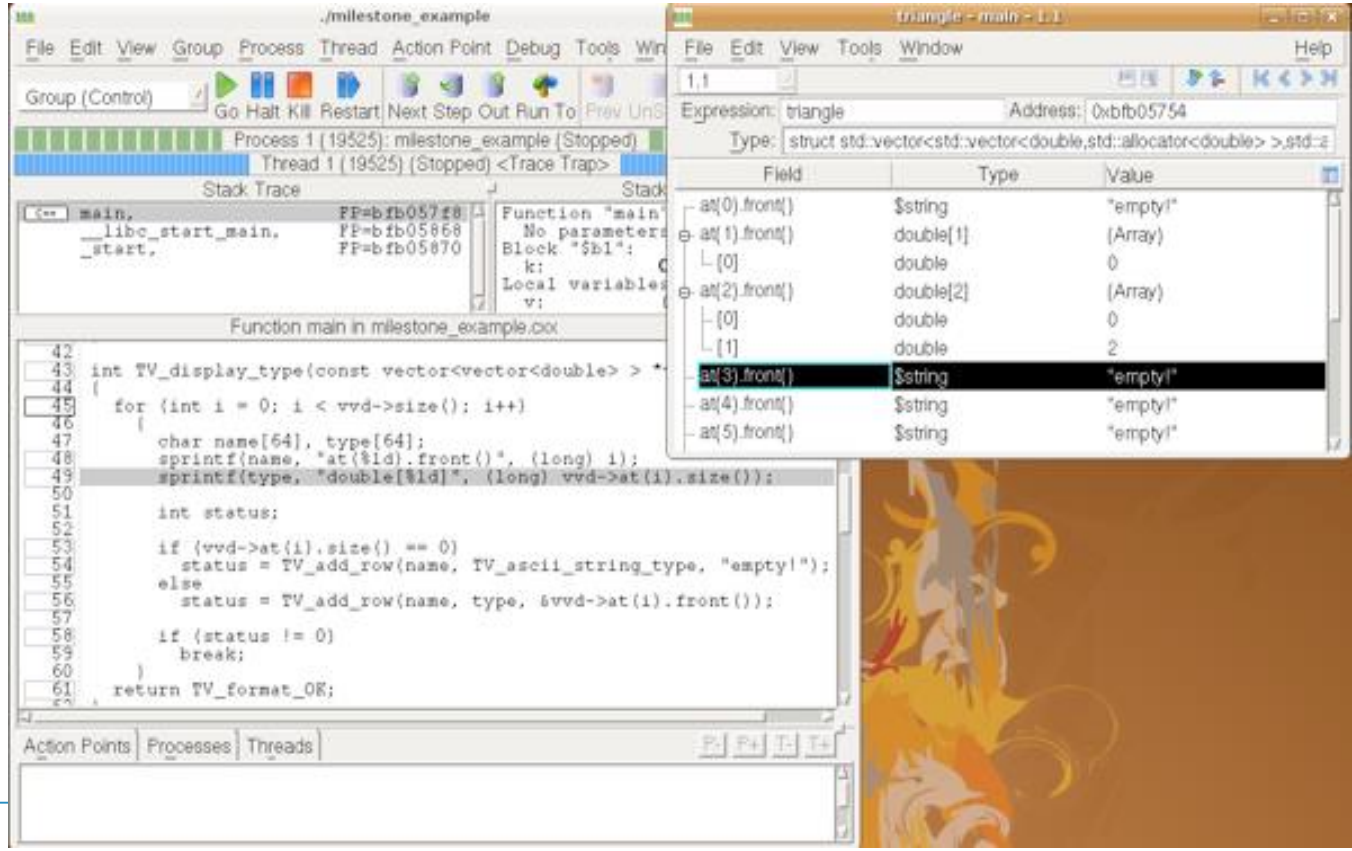
vector, list, map, string, set, multiset and multimap





C++View

- C++View is a simple way for you to define type transformations
  - Simplify complex data
  - Aggregate and summarize
  - Check validity
- Transforms
  - Type-based
  - Compose-able
  - Automatically visible
- Code
  - C++
  - Easy to write
  - Resides in target
  - Only called by TotalView



C++ View API

```
#include "tv_data_display.h"


int TV_ttf_display_type ( const T * );

int TV_ttf_add_row (
                        const char *field_name,
                        const char *type_name,
                        const char *address );
```

C++ View Example

```
class A {
        int i;
        char *s;
};

class B {
        A a;
        double d;
};

int TV_ttf_display_type ( const A *a )
{
        (void) TV_ttf_add_row ( "i", TV_ttf_type_int, &(a->i) );
        (void) TV_ttf_add_row ( "s", TV_ttf_type_ascii_string, a->s );
}

int TV_ttf_display_type ( const B *b )
{
        (void) TV_ttf_add_row ( "a", "A", &(b->a) );
        (void) TV_ttf_add_row ( "d", "double", &(b->d) );
}
```

Type Transformation Facility (TTF)

TTF is a TotalView subsystem that

- Allows you to transform the way information appears
- Doesn’t require changes to your code
- TTF transforms can be stored in a .tvd file for referencing during a debugging session or at TotalView startup

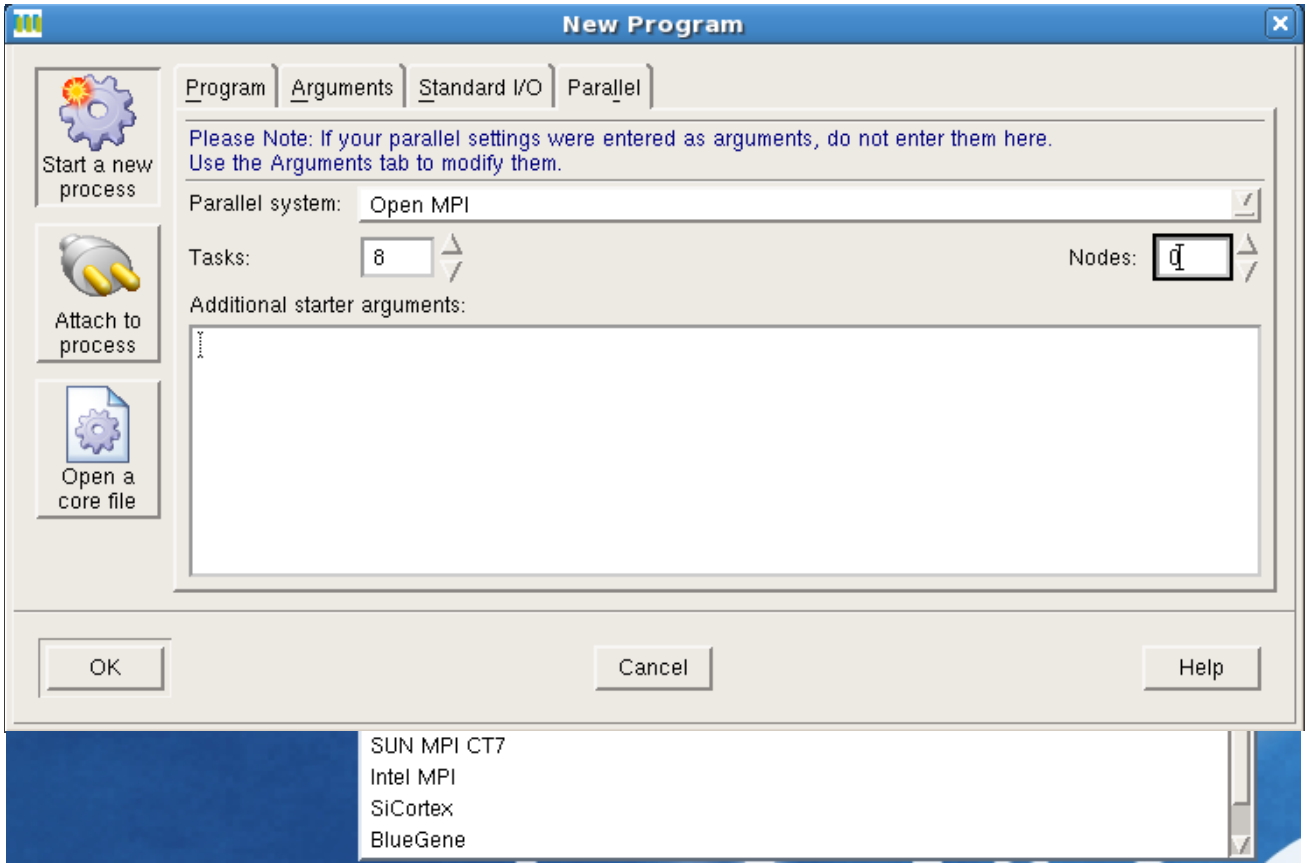
QT Qstring class transform

```
::TV::TTF::RTF::build_struct_transform {
    name {^class QString$}
    members {
       { ascii { $wstring_u16 cast {* {d -> data} } } }
    }
}
```

Parallel debugging adds complexity

TotalView Startup with MPI

In the Parallel tab, select:



your MPI preference, number of tasks, and number of nodes.
… then add any additional starter arguments

Command line MPI Startup

Using OpenMPI

mpirun -tv *args prog prog_args*

As an alternative, you can invoke TotalView on mpirun.

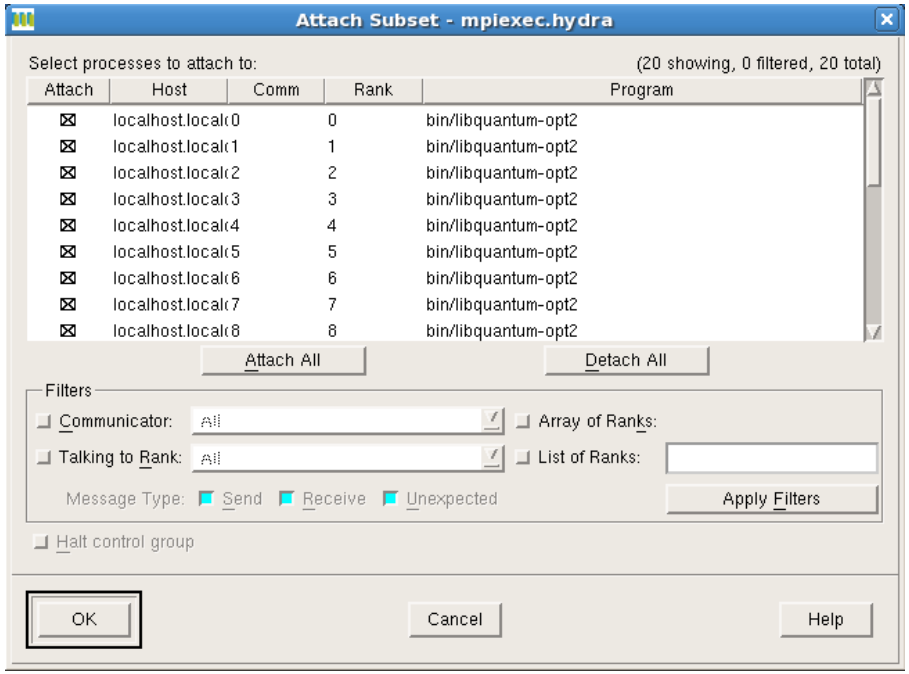
totalview -args mpirun *args ./prog*

For example, to start TotalView on a four-process MPI program:

totalview -args mpirun -np 4 ./mpi_program

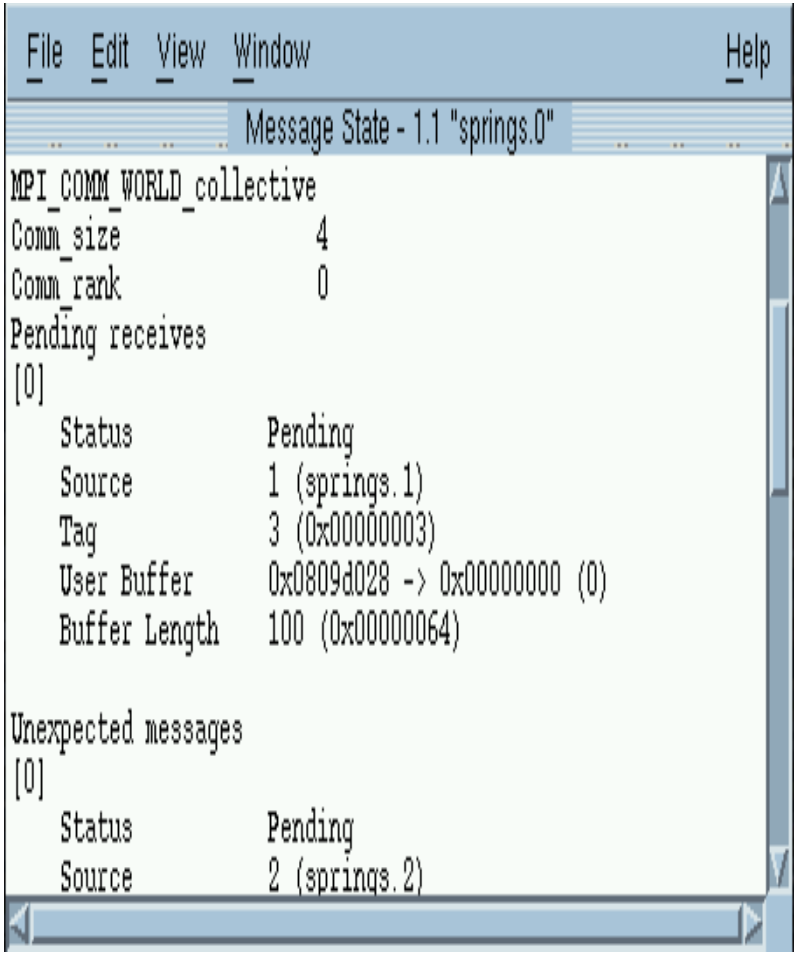
Subset Attach

- Connecting to a subset of a job reduces tokens and overhead
- Can change this during a run
- Groups->Subset Attach



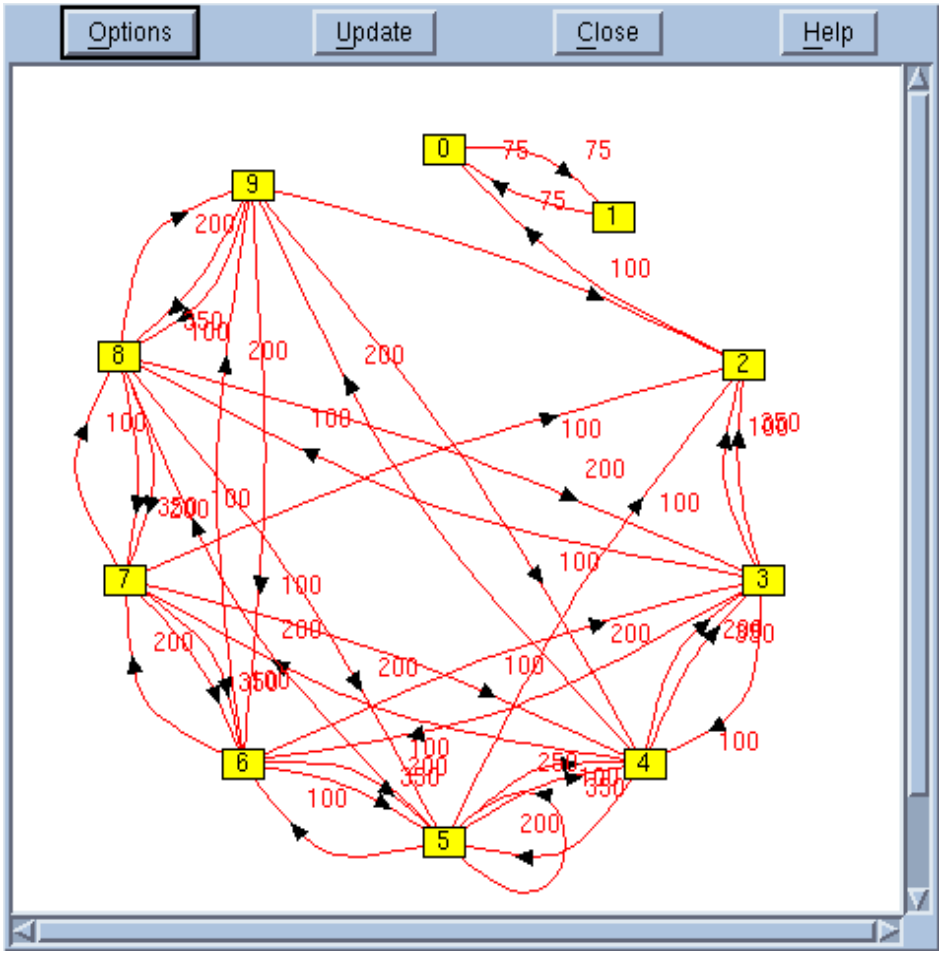
View MPI Message Queues

- Information visible whenever MPI rank processes are halted
- Provides information from the MPI layer
  - Unexpected messages
  - Pending Sends
  - Pending Receives
- Use this info to debug
  - Deadlock situations
  - Load balancing
- May need to be enabled in the MPI library
  - --enable-debug



Message Queue Graph

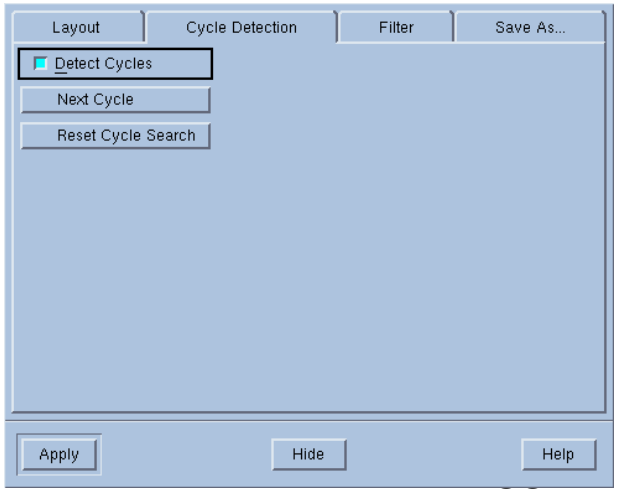
- Hangs & Deadlocks
- Pending Messages
  - Receives
  - Sends
  - Unexpected
- Inspect
  - Individual entries
- Patterns

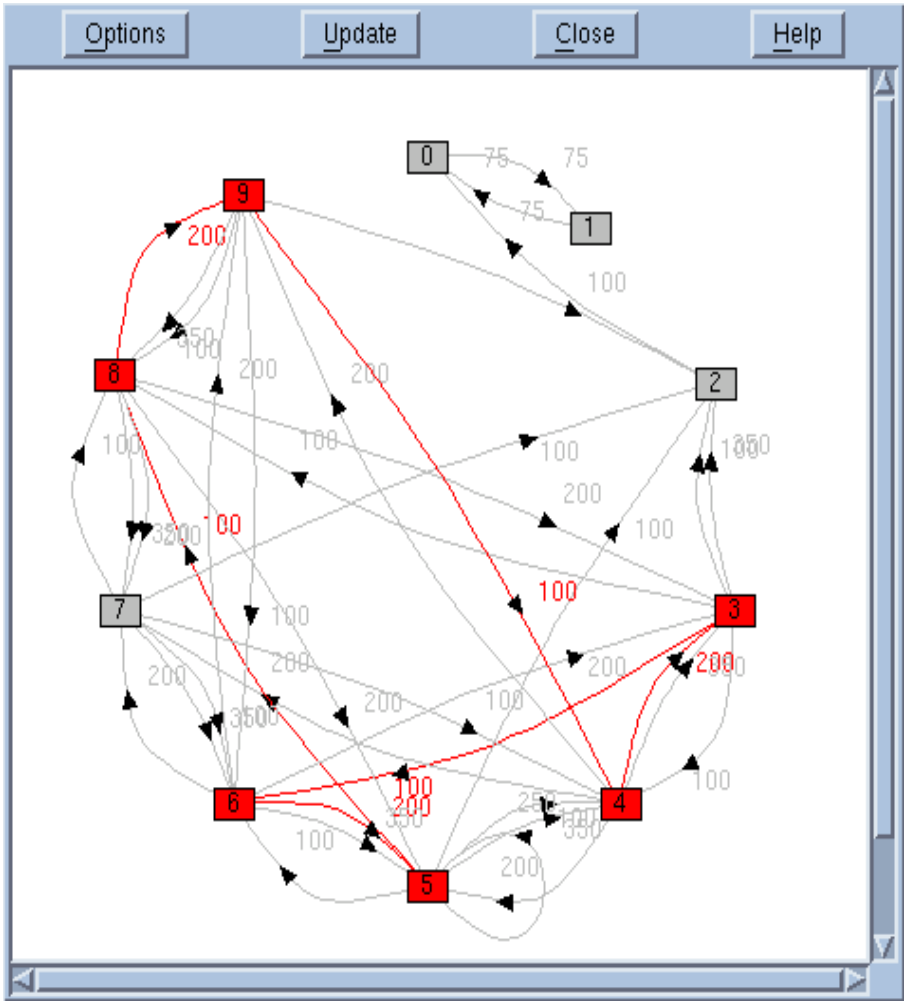


Message Queue Graph

Message Queue Debugging

- **Filtering**
  - Tags
  - MPI Communicators
- **Cycle detection**
  - Find deadlocks





Tricky memory issues

Memory Analysis

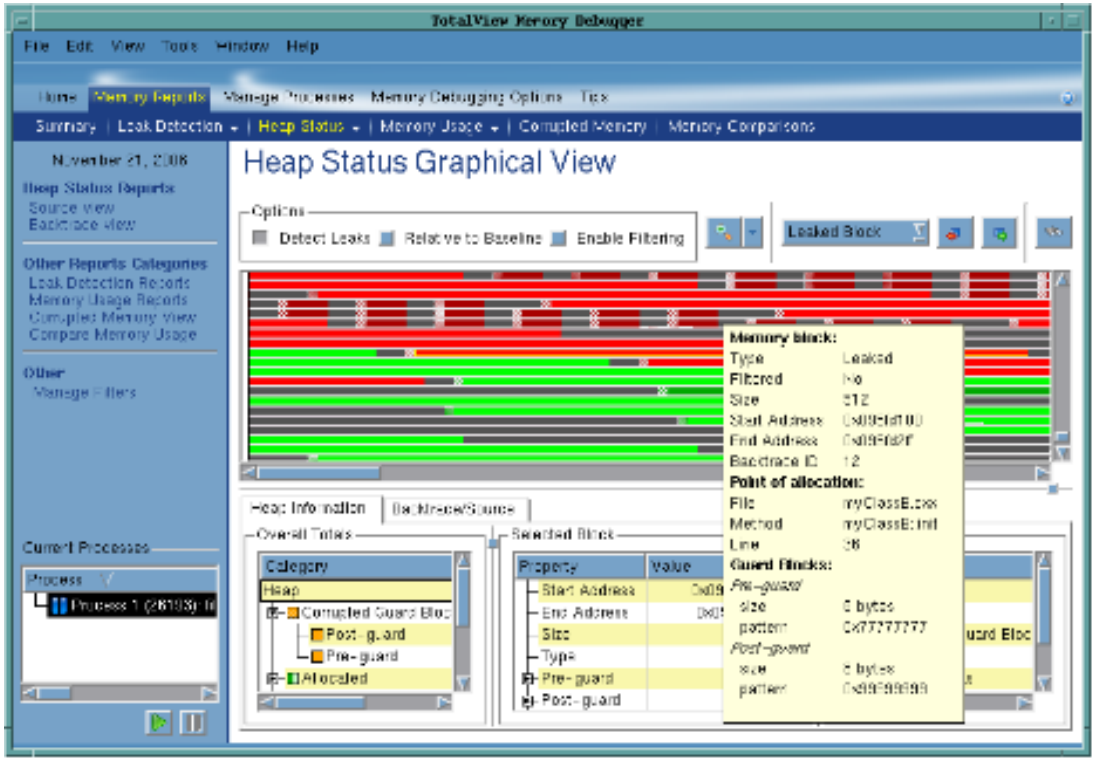
Eliminate memory errors and slowdown hackers from using them

Runtime Memory Analysis : Eliminate Memory Errors

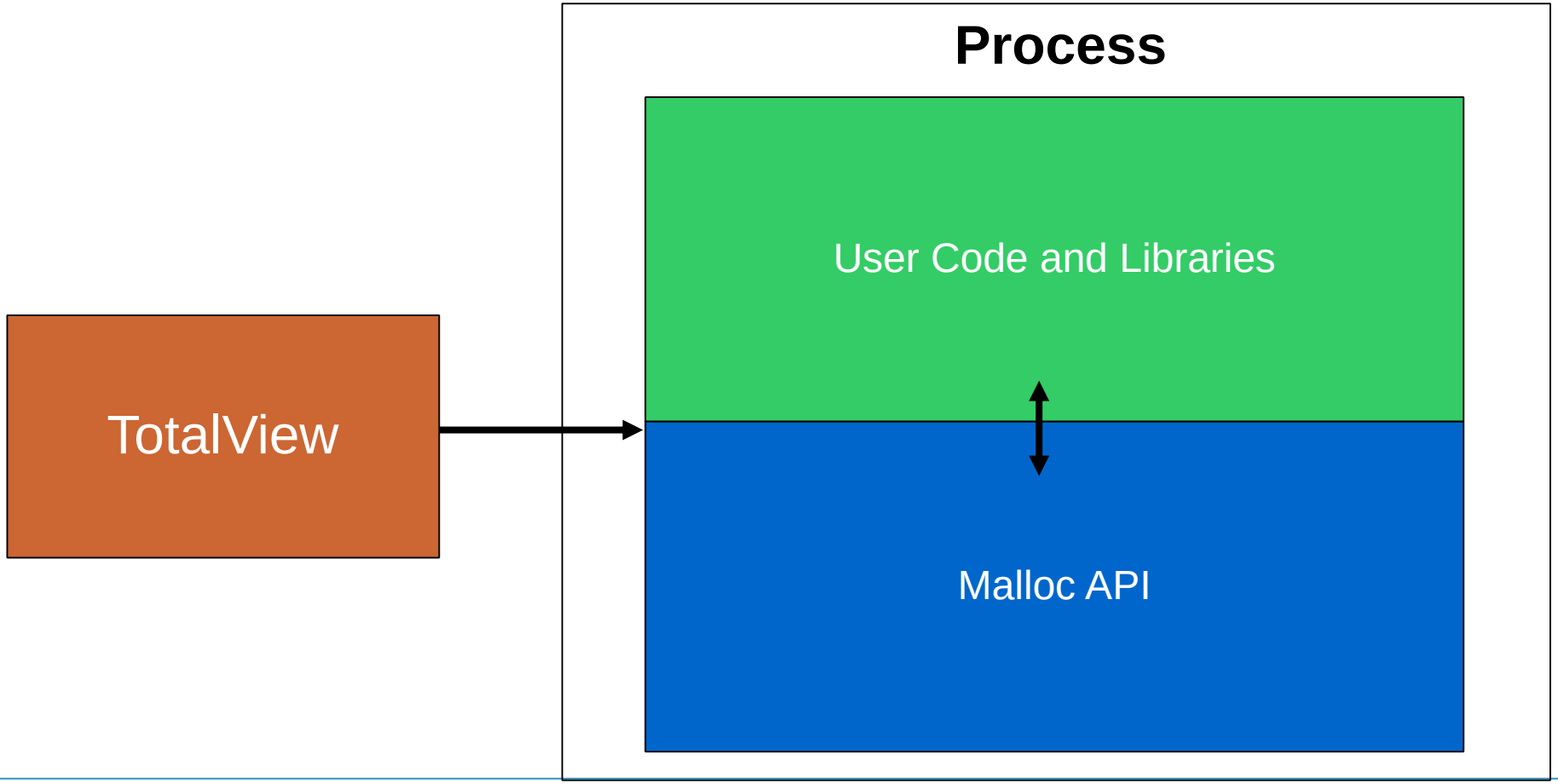
- Detects memory leaks *before* they are a problem
- Explore heap memory usage

Features

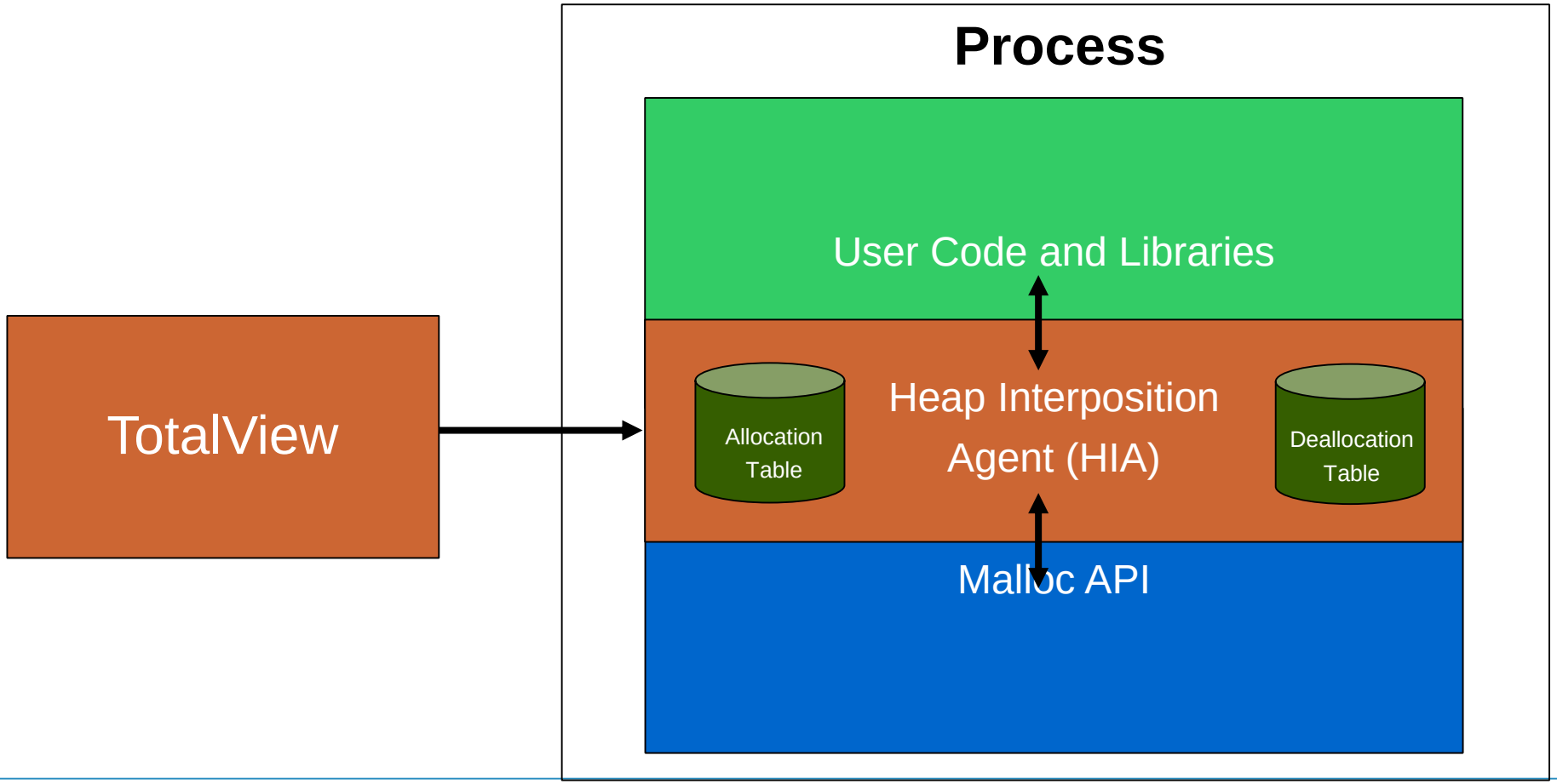
- Detects
  - Malloc API misuse
  - Memory leaks
  - Buffer overflows
- Low runtime overhead
- Easy to use
  - Works with vendor libraries
  - No recompilation
  - No instrumentation



The Agent and Interposition



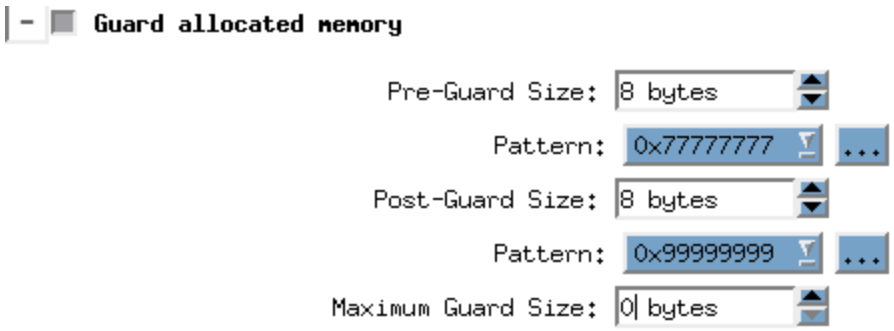
The Agent and Interposition



Guard Blocks

Guard allocated memory

When selected, the Memory Debugger writes guard blocks before and after a memory block that your program allocates



Pre-Guard and Post-Guard Size:

Sets the size in bytes of the block that the Memory Debugger places immediately before and after the memory block that your program allocates

Pattern:

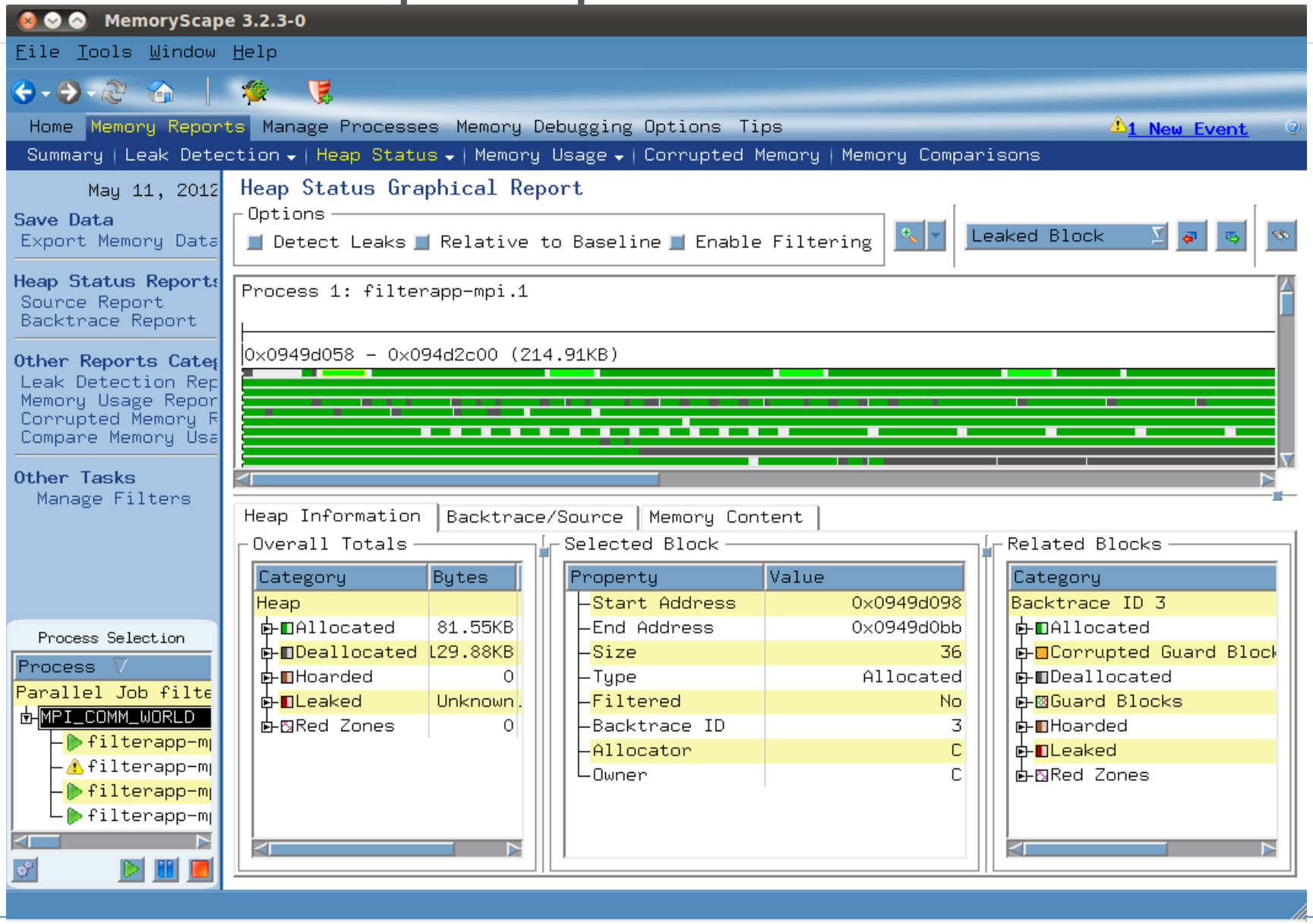
Indicates the pattern that the Memory Debugger writes into guard blocks. The default values are 0x77777777 and 0x99999999

Red Zones

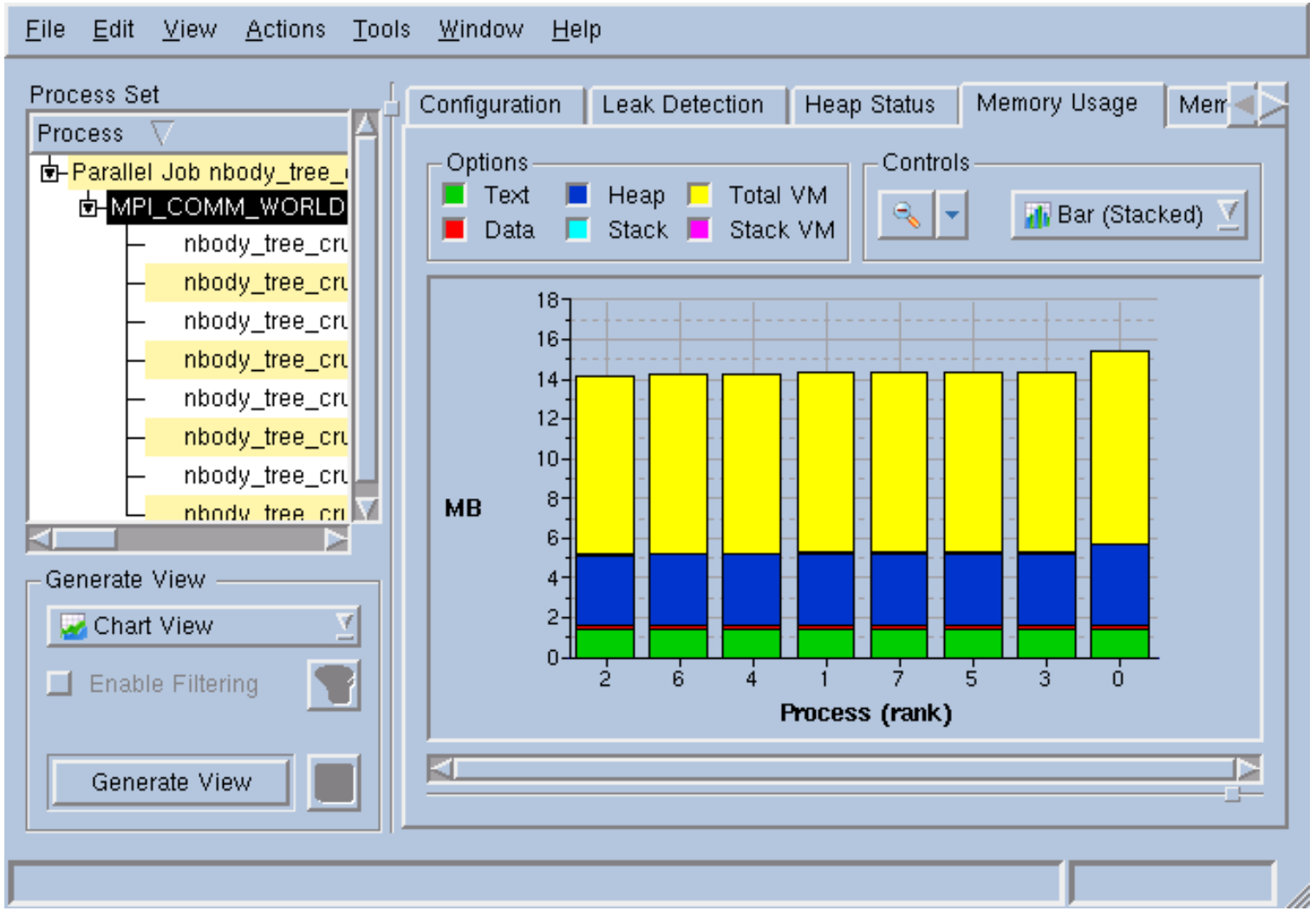
Red Zones : instant array bounds detection

- Red Zones provides:
  - Both read and write memory access violations.
  - Immediate detection of memory overruns.
  - Detection of access violations both before and after the bounds of allocated memory.
  - Detection of deallocated memory accesses.
- Red Zones events
  - MemoryScape will stop your programs execution and raise an event alerting you to the illegal access. You will be able to see exactly where your code over stepped the bounds.

Heap Graphical View

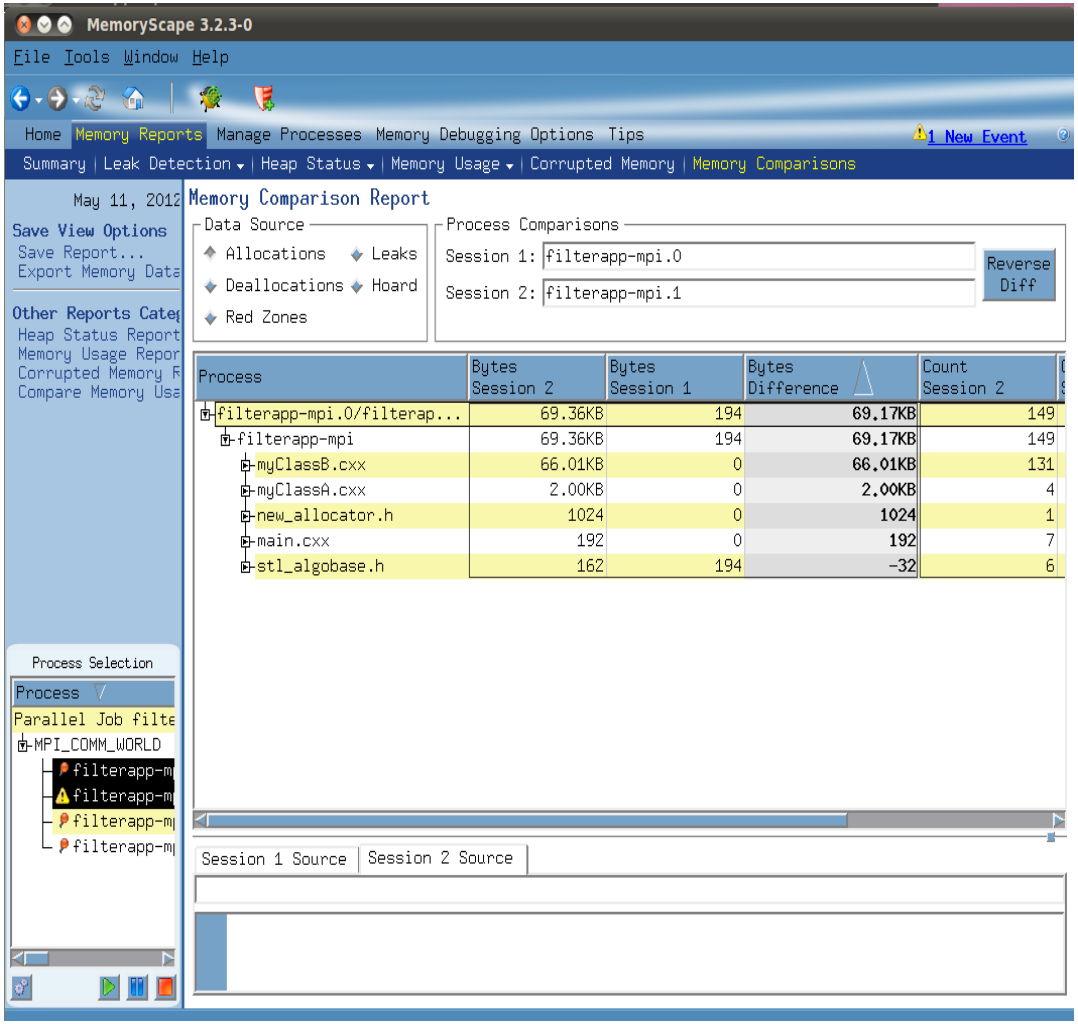


Memory Usage Statistics



Memory Comparisons

- “Diff” live processes
  - Compare processes across cluster
- Compare with baseline
  - See changes between point A and point B
- Compare with saved session
  - Provides memory usage change from last run



Batch debugging can help

Regression Testing

- How do you make sure a bug you fixed never returns?
  - Build a regression test
  - Issue is it typically is time consuming
- What is the method to build a regression test?
  - Use the tools that helped you find it
- How do you run a regression test?
  - Invoke it during your build process

- Enter TotalView scripts
  - Command line driven
  - Access to application internals
  - Same commands as in the debugger

```
•!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!
•! Print
•!
•! Process:
•!   ./server (Debugger Process
ID:  1, System ID:  12110)
•! Thread:
•!   Debugger ID:  1.1, System
ID:  3083946656
•! Time Stamp:
•!   06-26-2008 14:04:09
•! Triggered from event:
•!   actionpoint
•! Results:
•!    foreign_addr = {
•!      sin_family = 0x0002 (2)
•!      sin_port = 0x1fb6 (8118)
•!      sin_addr = {
•!        s_addr = 0x6658a8c0
(1717086400)
•!      }
•!      sin_zero = ""
•!    }
•!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!
```

TVScript Overview

- Gives you non-interactive access to TotalView's capabilities
- Useful for
  - Debugging in batch environments
  - Watching for intermittent faults
  - Parametric studies
  - Automated testing and validation
- TVScript is a script (not a scripting language)
  - It runs your program to completion and performs debugger actions on it as you request
  - Results are written to an output file
  - No GUI
  - No interactive command line prompt

TVScript Syntax

- tvscript syntax:
  - tvscript [options] [filename] [-a program_args]

- Options express (“event”,”action”) pairs
  - Typical events
    - Action_point
    - Any_memory_event
    - Guard_corruption
    - error
  - Typical actions
    - Display_backtrace
    - List_leaks
    - Save_memory
    - Print [-slice {*slice_exp*] {*variable* | *exp*}

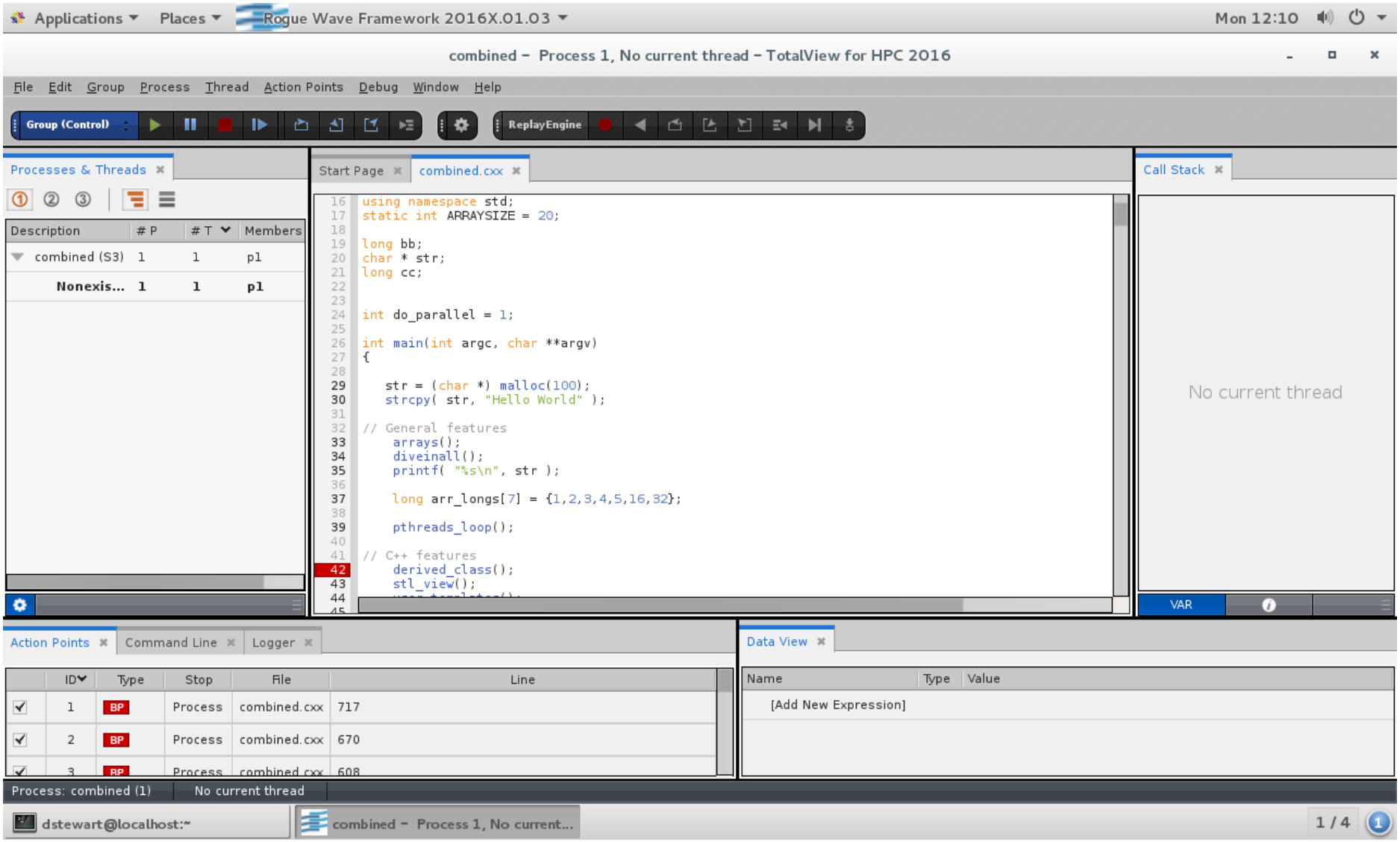
- Example

```
•!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!
•! Print
•!
•! Process:
•!   ./server (Debugger Process ID:  1, System ID:  12110)
•! Thread:
•!   Debugger ID:  1.1, System ID:  3083946656
•! Time Stamp:
•!   06-26-2008 14:04:09
•! Triggered from event:
•!   actionpoint
•! Results:
•!    foreign_addr = {
•!      sin_family = 0x0002 (2)
•!      sin_port = 0x1fb6 (8118)
•!      sin_addr = {
•!        s_addr = 0x6658a8c0 (1717086400)
•!      }
•!      sin_zero = ""
•!    }
•!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!
```

```
-create_actionpoint "#85=>print foreign_addr”
```

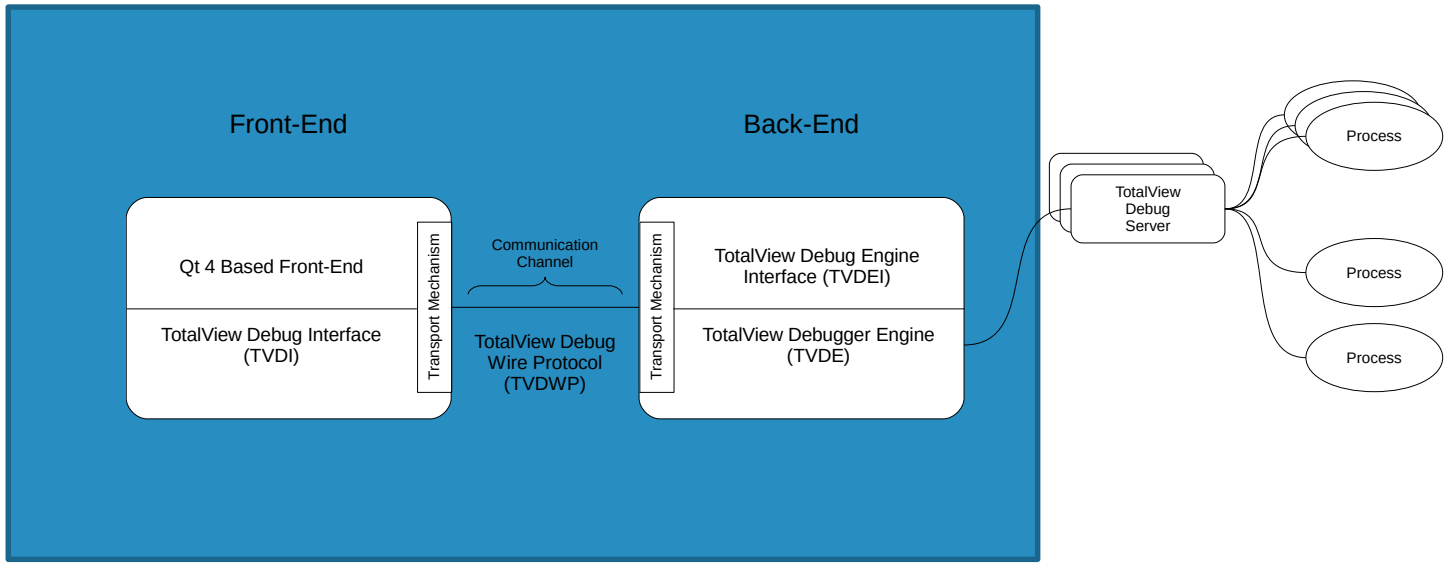
New TotalView UI

TotalView new UI



New Architecture in building UI

Used to be one large application



Why does the architecture matter

- This isn’t short term thinking
  - No need for XWindows on target platforms
  - Performance
  - Scalability
  - Platforms
- More effective debugger on all platforms
- Easier 3rd party integrations

Q&A

Thanks!

- Visit the website
  - http://www.roguewave.com/products/totalview.aspx
  - Videos
  - Documentation
  - Sign up for an evaluation
  - Contact customer support
  - Post on the user forum