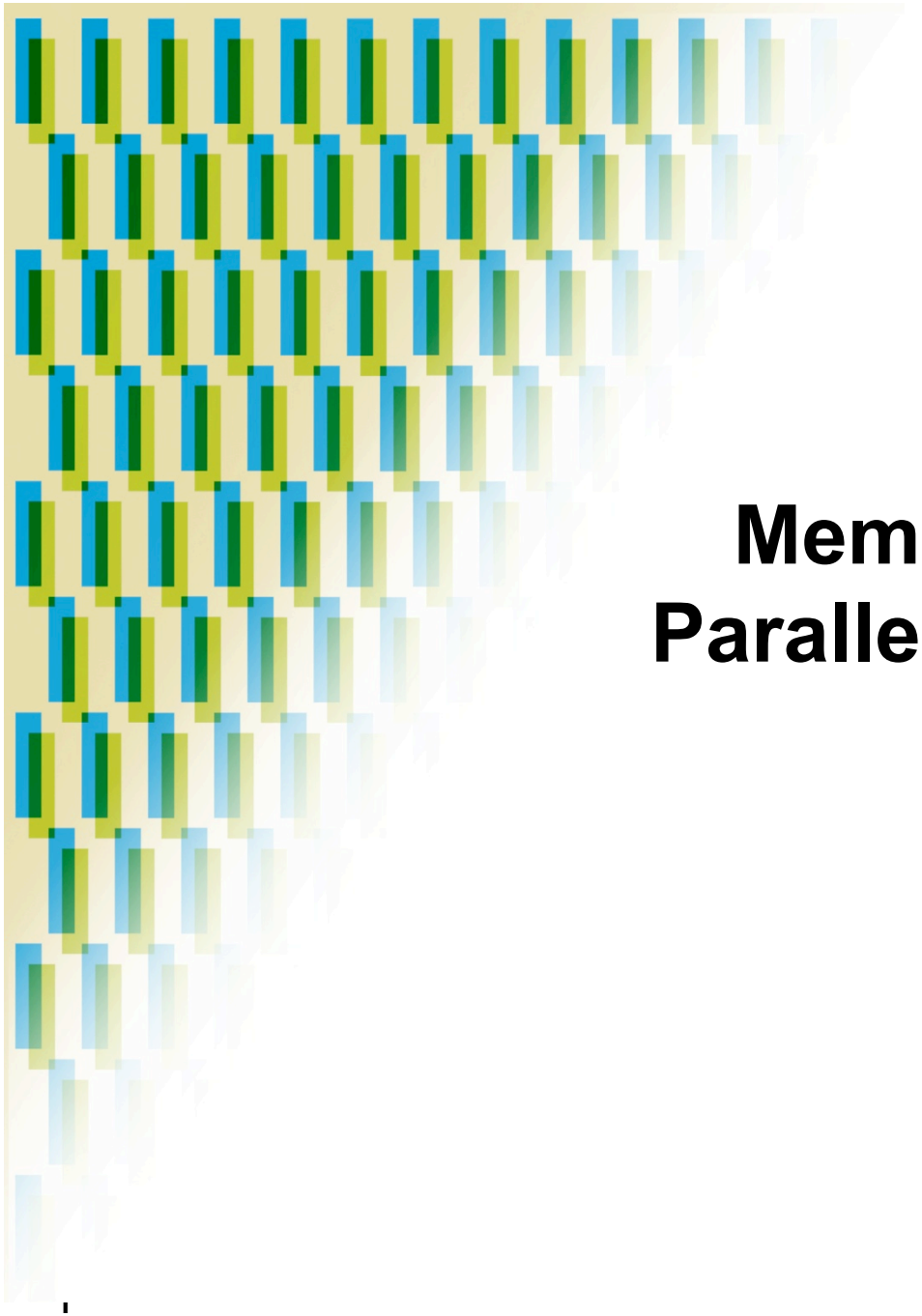
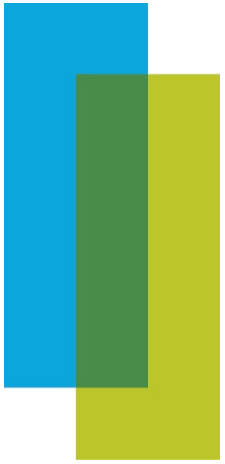


TOTALVIEW TECHNOLOGIES

Memory Debugging Parallel Applications on BlueGene

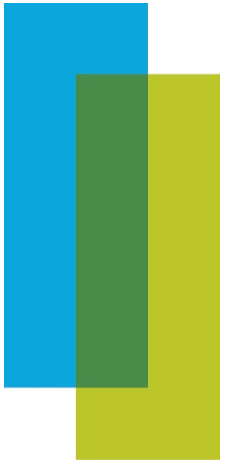
SciComp
May 21, 2009
Ed Hinkel



TOTALVIEW TECHNOLOGIES

Agenda

- Introduction
- Memory (mis) Management
- Memory Debug Options & Issues
- Memory Debugging Techniques
- What's New

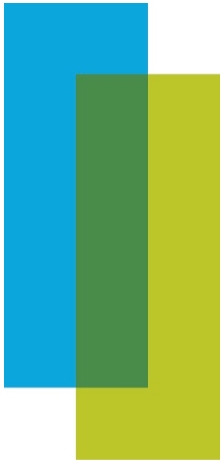


TOTALVIEW TECHNOLOGIES

Memory Bugs

Can be very elusive!

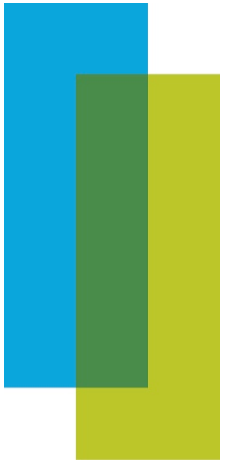
- **Memory bugs are often not immediately fatal**
- **Memory bugs can lurk in a code base for long periods of time**
- **Memory bugs can suddenly emerge when**
  - A program is ported to a new architecture
  - Programs are scaled up to a larger size
  - When code is adapted and reused from one program to another.
- **Most memory bugs are not detected by compilers**



TOTALVIEW TECHNOLOGIES

Memory bugs are often manifested in unusual ways

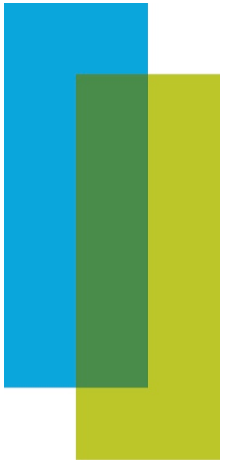
- Memory bugs often go undetected until the worst possible time
- Symptoms often surface long after the actual damage is done
- Some only surface after hours or even days of operation
- In many cases, the programs affected are “innocent bystanders”



TOTALVIEW TECHNOLOGIES

Memory Issues are on the Rise

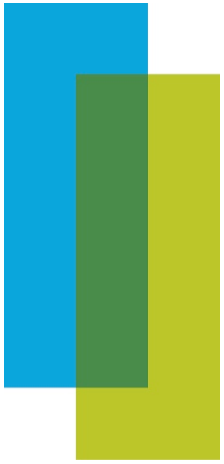
- Core counts are growing at an amazing rate
- But the Memory "per CPU" is trending downward
- Proper memory management is becoming more critical
- More than ever, you need to really know how memory is being used



TOTALVIEW TECHNOLOGIES

What is a Typical Memory Bug?

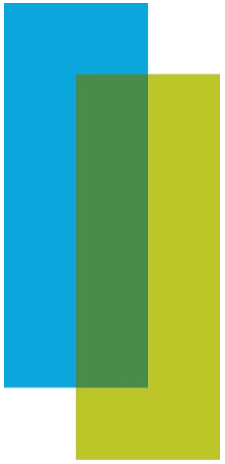
- **A Memory Bug is a mistake in the management of heap memory**
  - **Failure to check for error conditions**
  - **Leaking: Failure to free memory**
  - **Dangling references: Failure to clear pointers**
  - **Memory Corruption**
    - **Writing to memory not allocated**
    - **Over running array bounds**



TOTALVIEW TECHNOLOGIES

Memory Debugging Options

- **Developers have a range of options (many free) for memory debugging… But:**
  - **Many programs are often singular in function, requiring an array of "solutions".**
  - **Most often, there is significant "overhead" issues to consider:**
    - **Performance hits can often be huge, with unacceptable slowdowns**
    - **Additional memory usage can make bad things worse**
    - **Special instrumentation requirements can often produce an unwelcome exercise of the Heisenberg uncertainty principle**
    - **Scalability can be a big problem**



TOTALVIEW TECHNOLOGIES

Memory Debugging Options

So,

How Does One

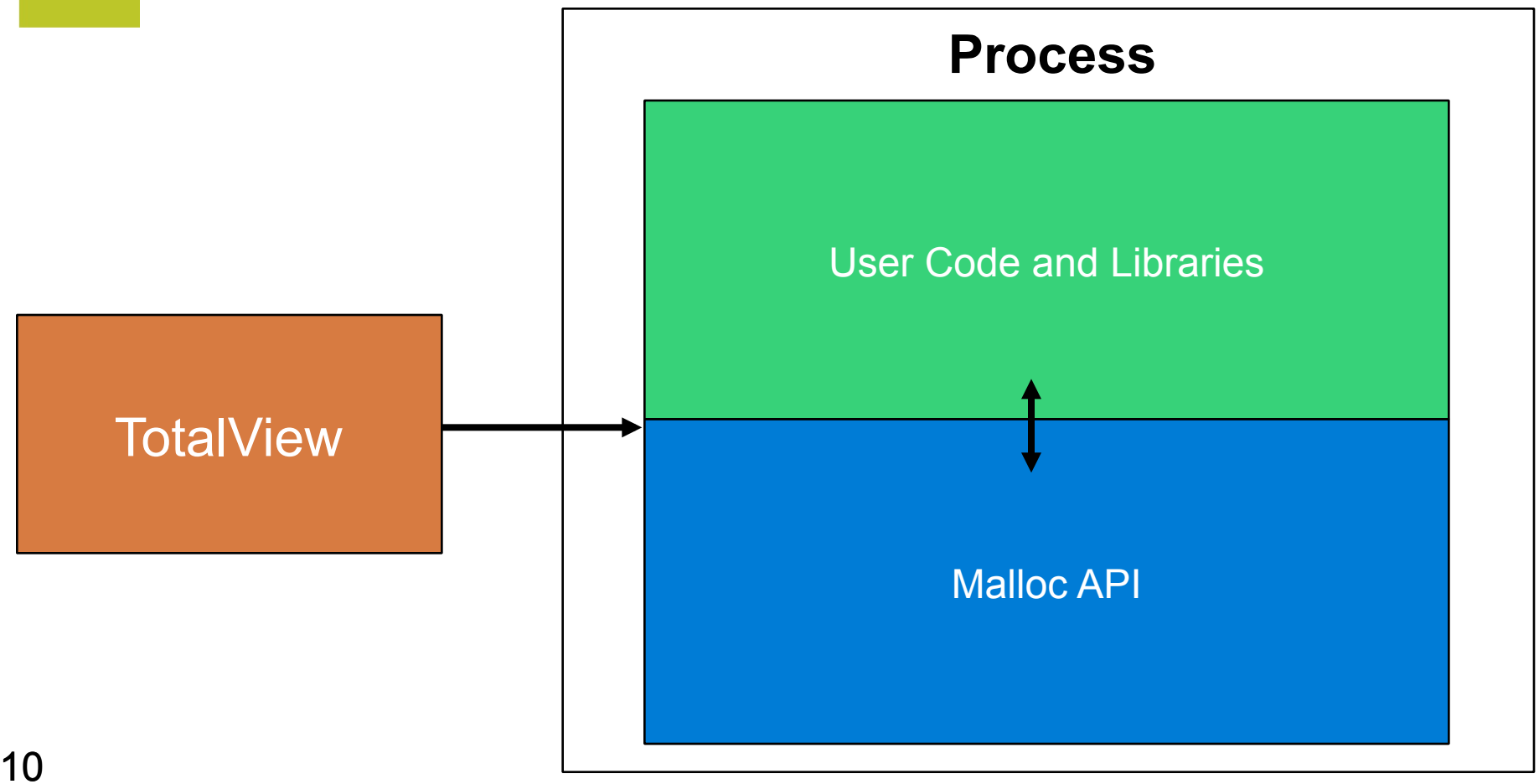
Memory Debug Effectively?

TotalView Memory Debugging Products

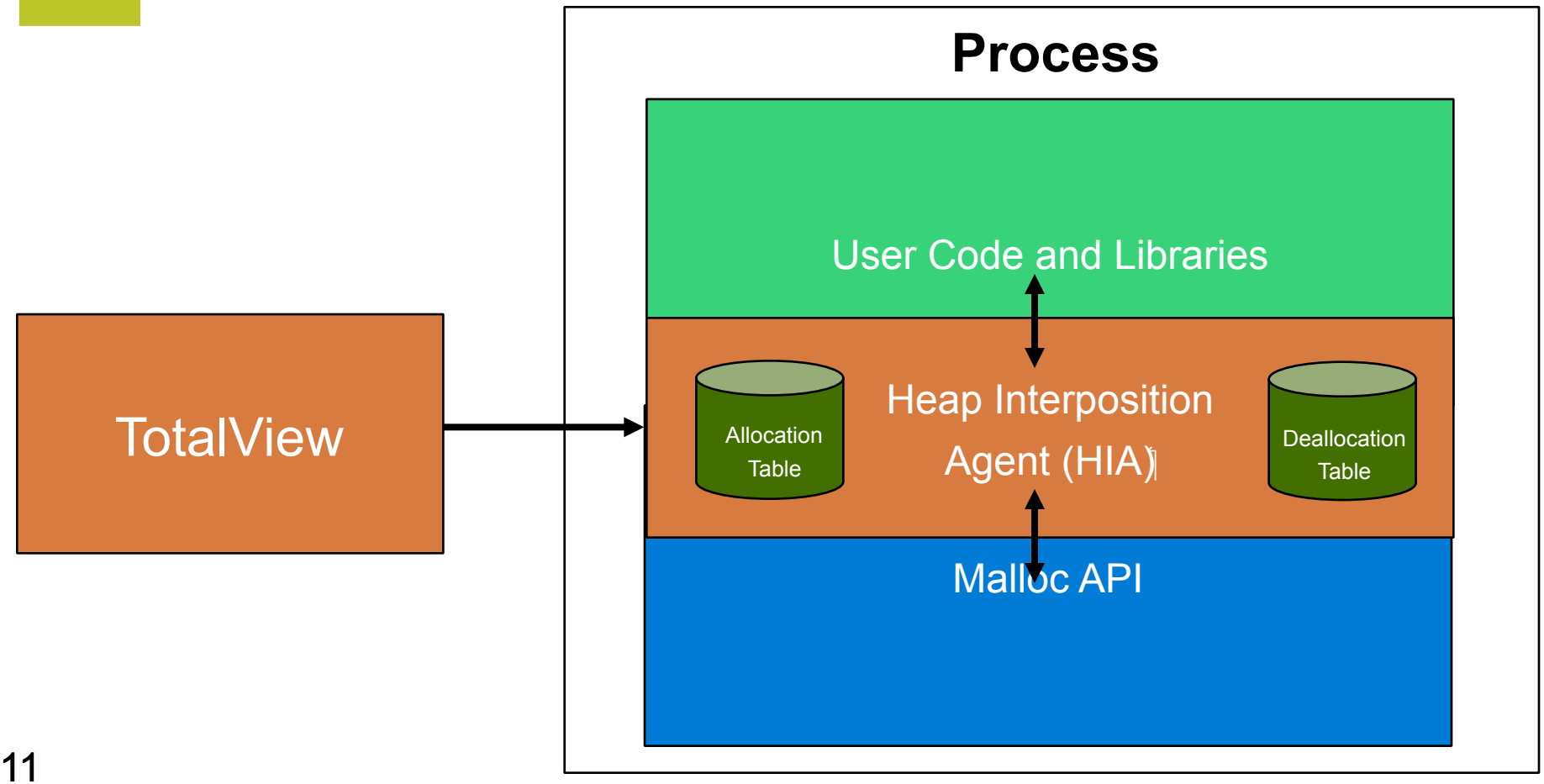
- **TotalView Source Code Debugger**
  - Fully integrated Memory Debugging Capabilities

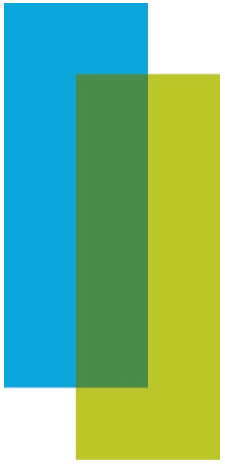
- **MemoryScape Memory Debugger**
  - Standalone Memory Debugging
    - Non-developer environments
    - Quality Assurance
    - Test groups
    - Customers

TotalView's Interposition Agent



TotalView's Interposition Agent



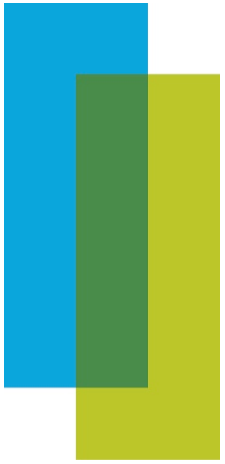


TOTALVIEW TECHNOLOGIES

TotalView HIA Technology

- **Advantages of TotalView HIA Technology**

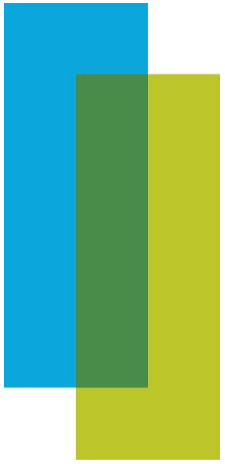
  - **Use it with your existing builds**
    No Source Code or Binary Instrumentation
  - **Programs run nearly full speed**
    - **Low performance overhead**
  - **Low memory overhead**
    - **Efficient memory usage**
  - **Support for a wide range of platforms – including Cell**



Memory Debugger Features

TOTALVIEW TECHNOLOGIES

- Automatic allocation problem detection
- Heap Graphical View
- Leak detection
- Block painting
- Dangling pointer detection
- Deallocation/reallocation notification
- Memory Corruption Detection - Guard Blocks
- Memory Hoarding
- Memory Comparisons between processes
- Collaboration features



Enabling Memory Debugging

TOTALVIEW TECHNOLOGIES

Setting up a memory debug session…

Fexibility is Key

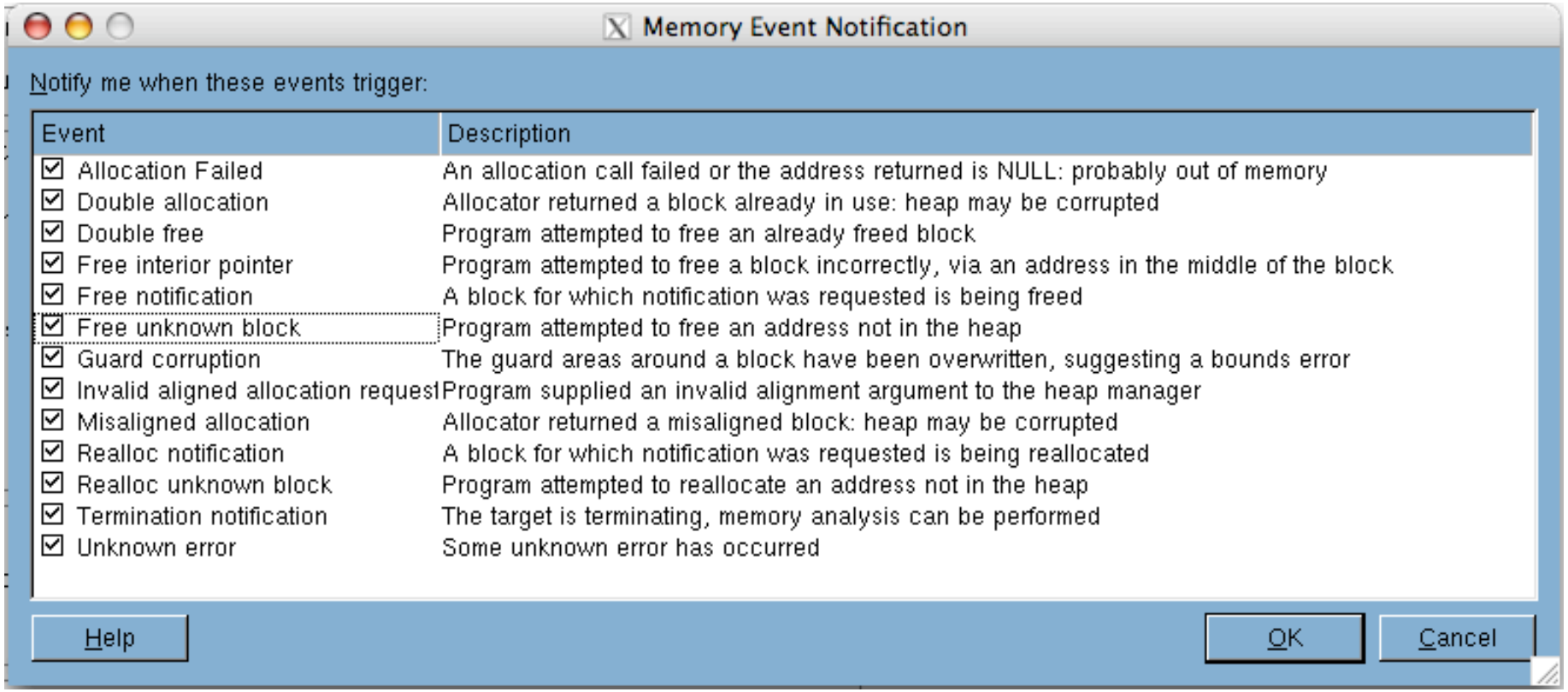
Enabling Memory Debugging
Memory Event Notification

TOTALVIEW TECHNOLOGIES

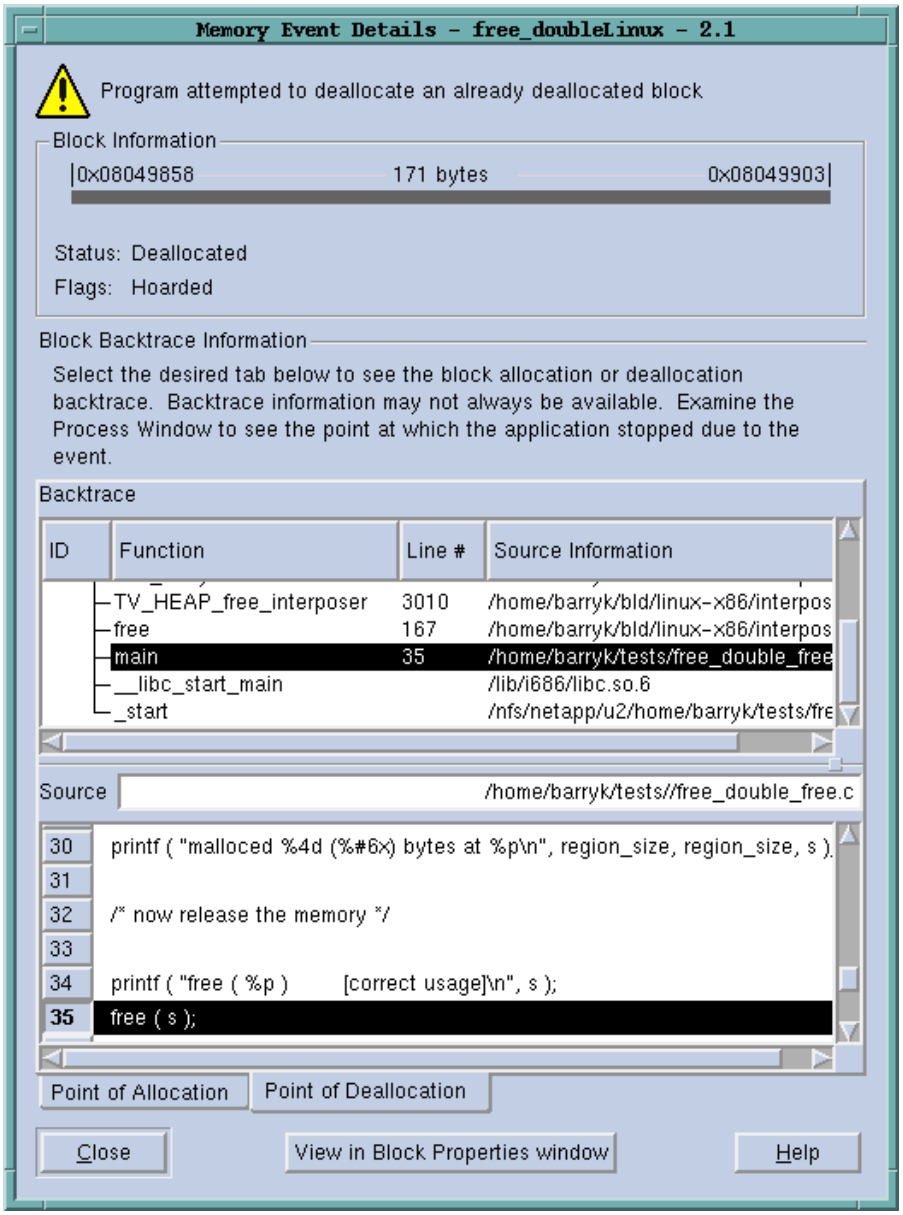
Halt execution on memory event or error

Use the **Advanced** button to control actions for individual events.

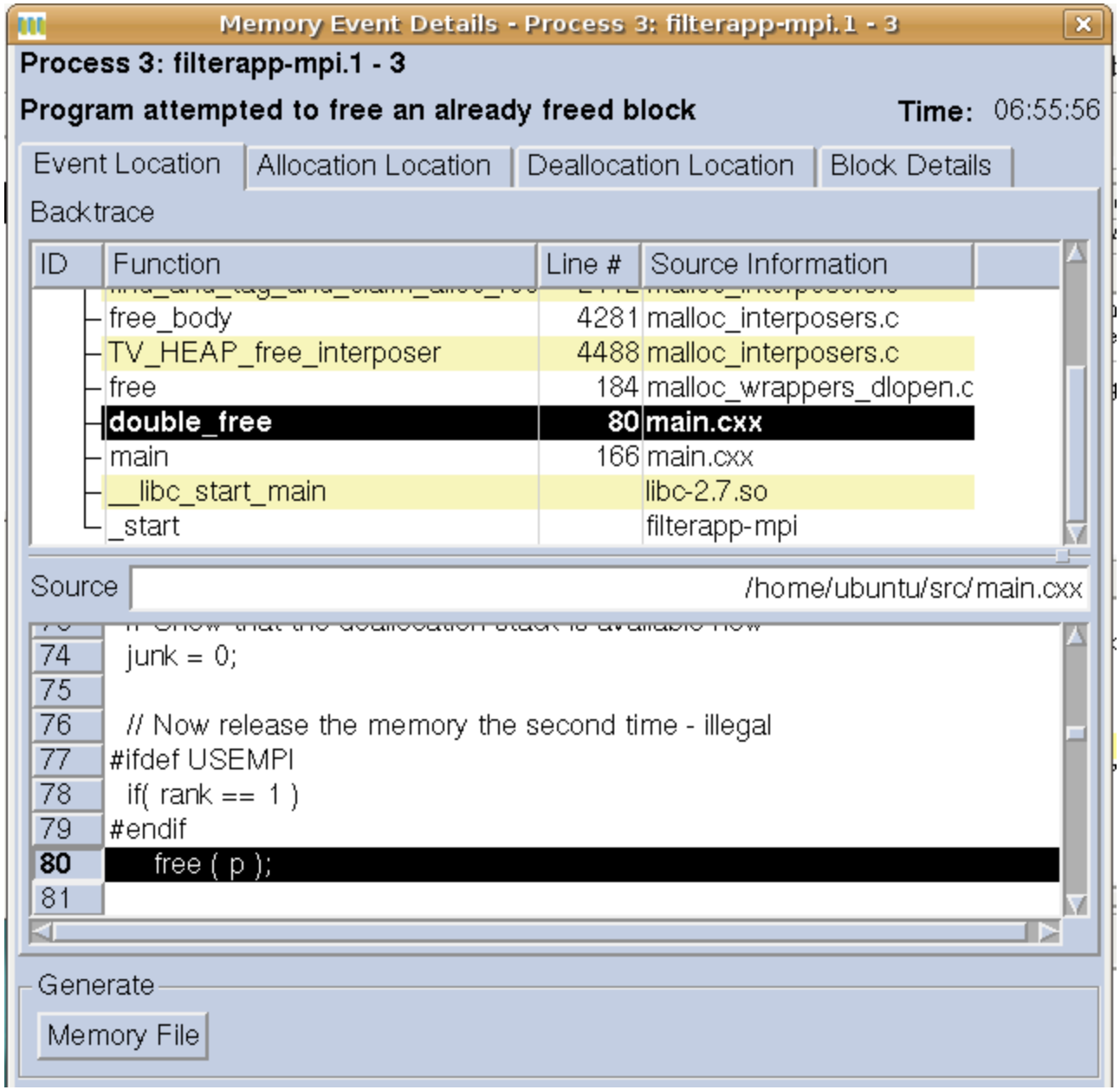
Advanced...



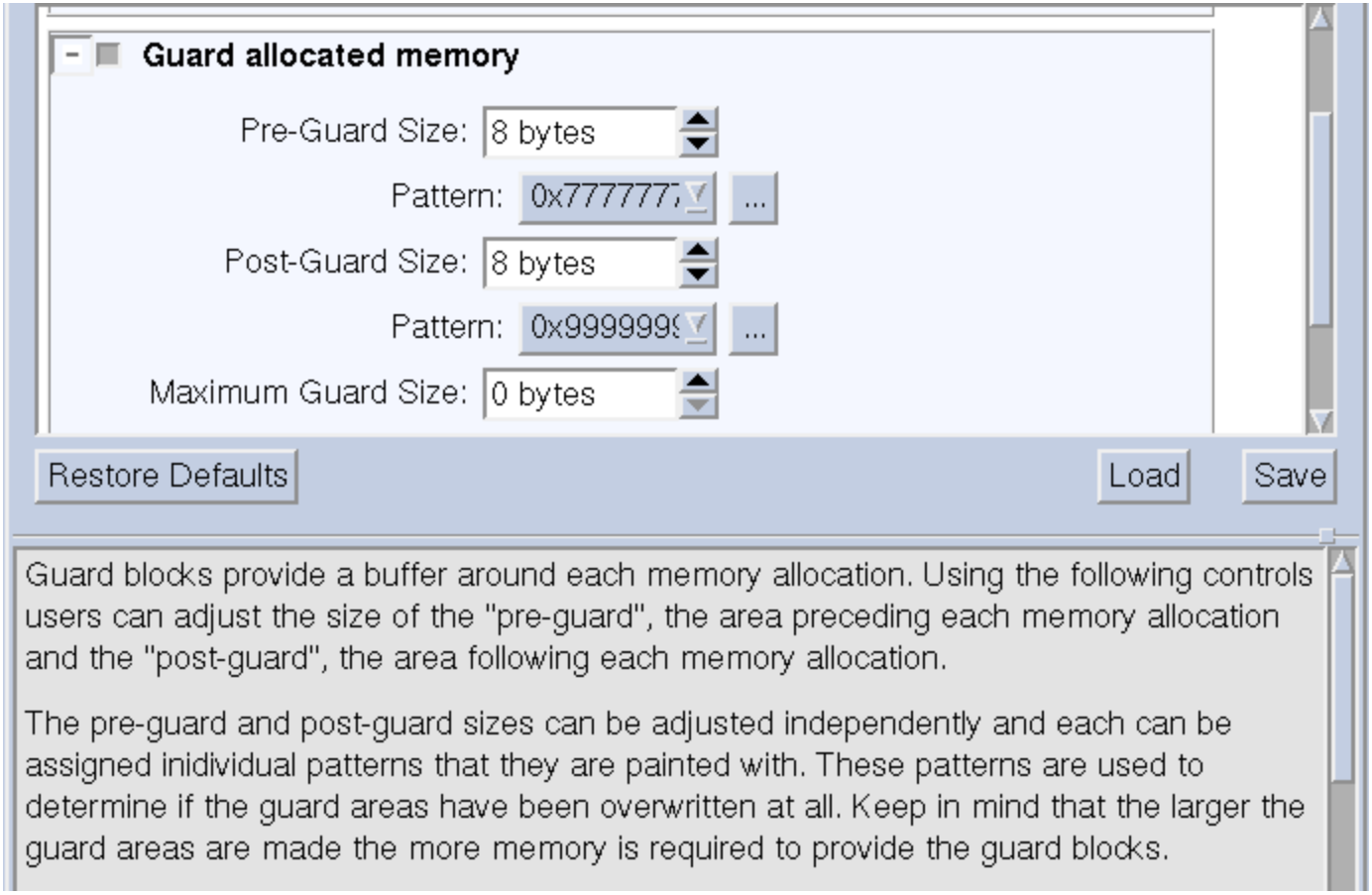
Memory Event Details Window



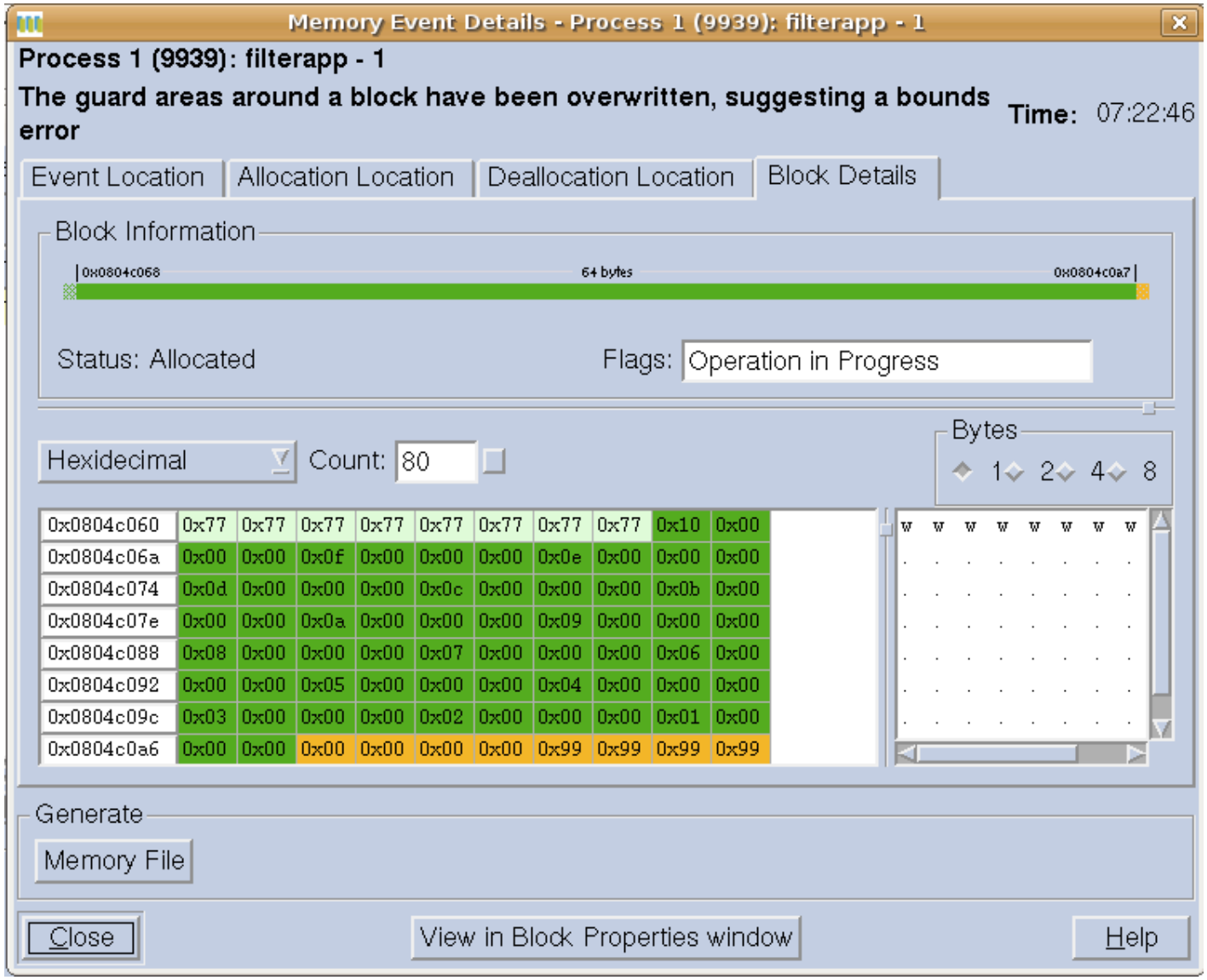
TOTALVIEW TECHNOLOGIES



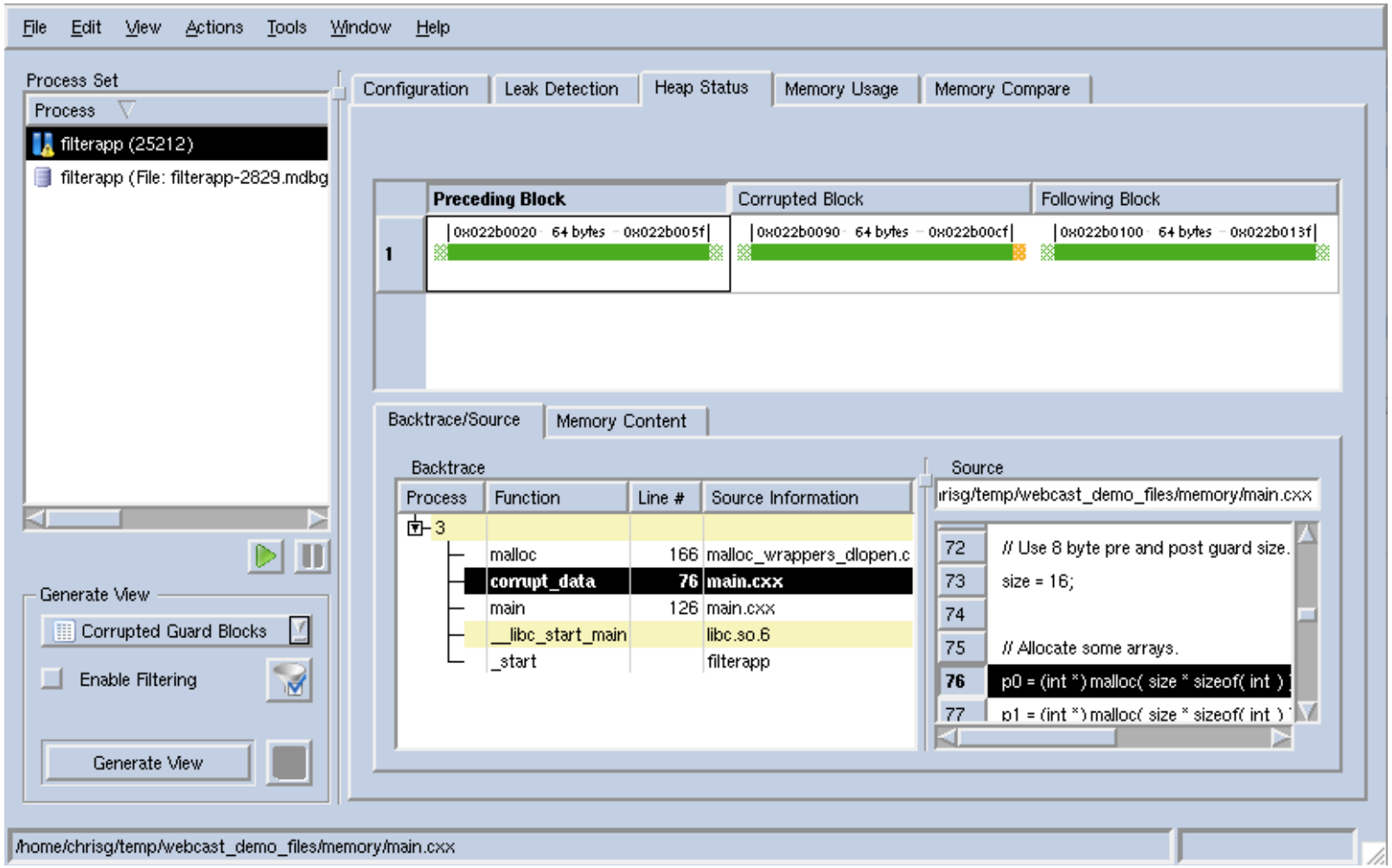
Memory Corruption Detection (Guard Blocks)



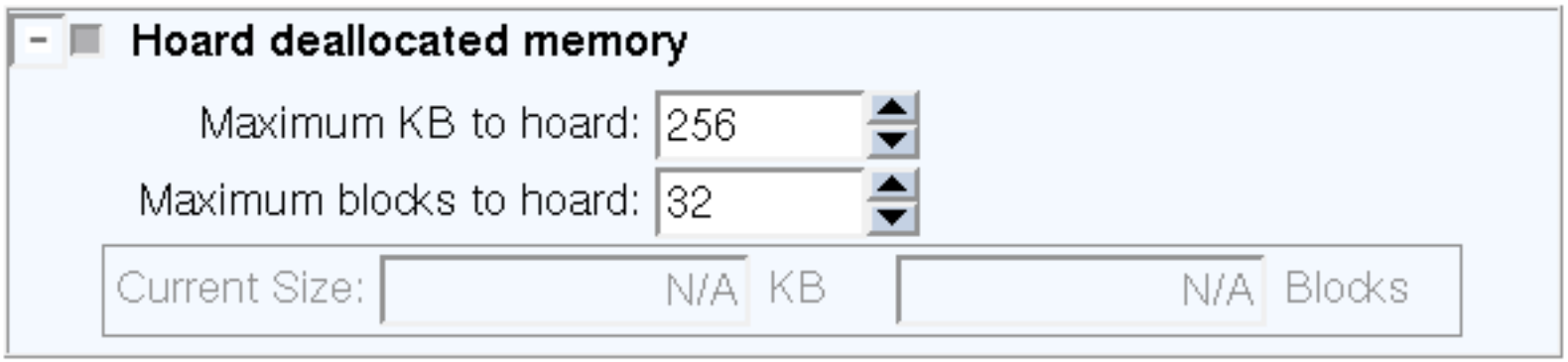
Memory Corruption Detection (Guard Blocks)



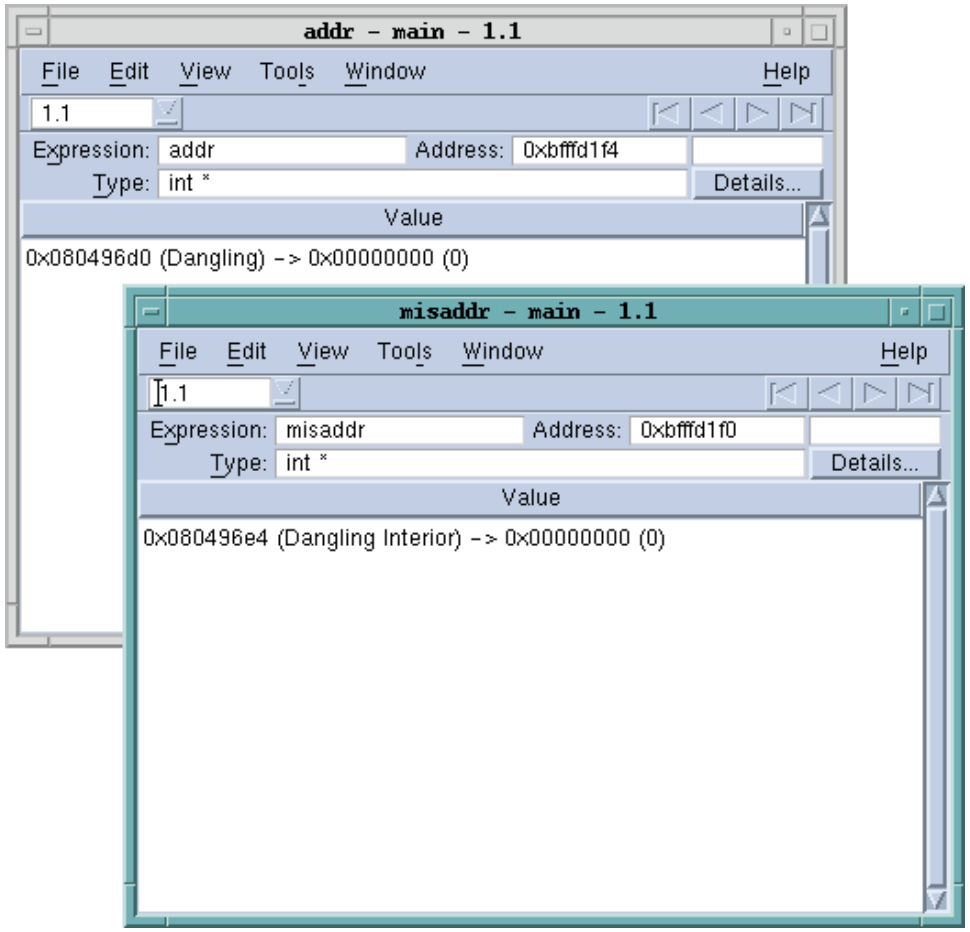
Memory Corruption Report



Enabling Memory Debugging
Painting & Hoarding

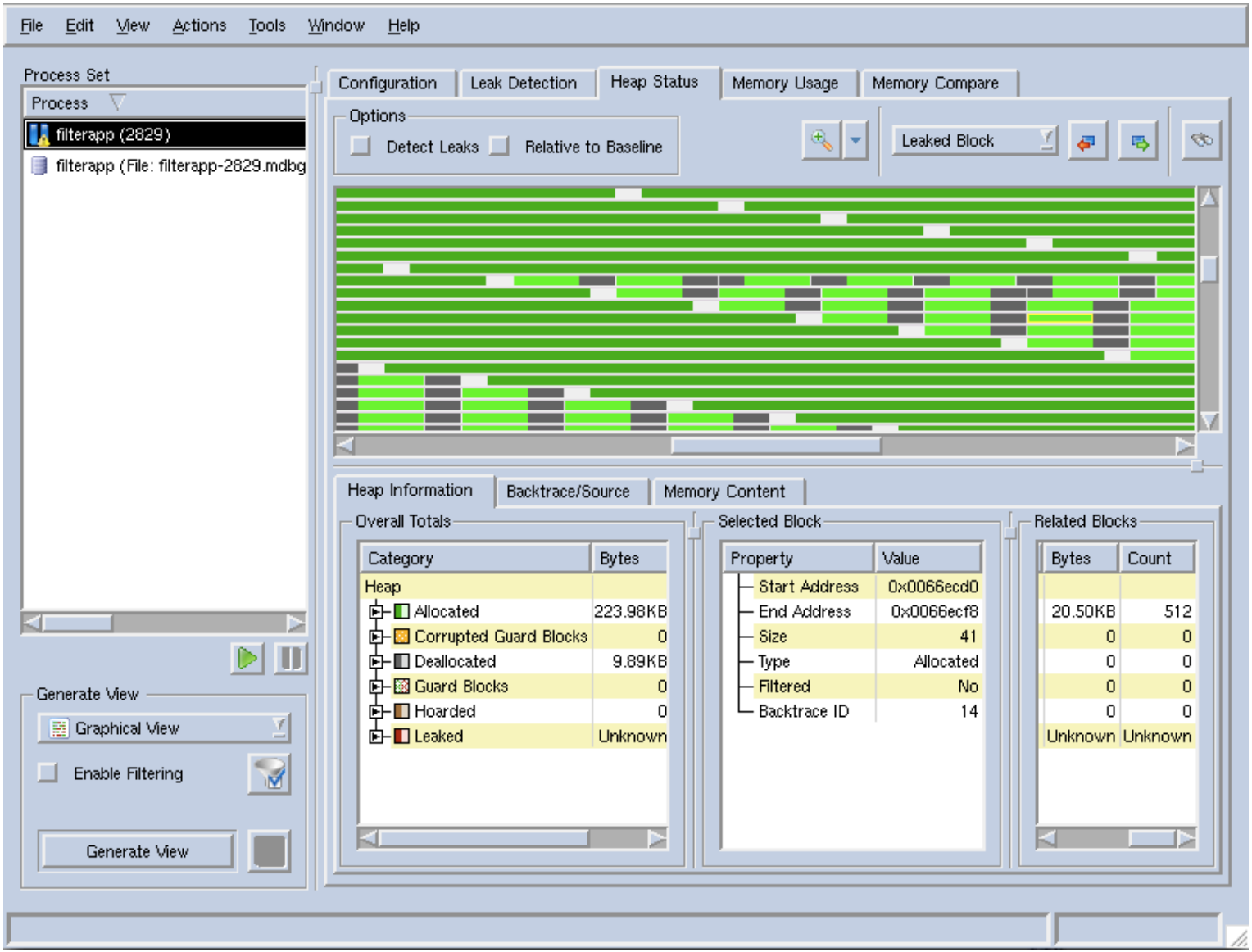


Dangling Pointer Detection

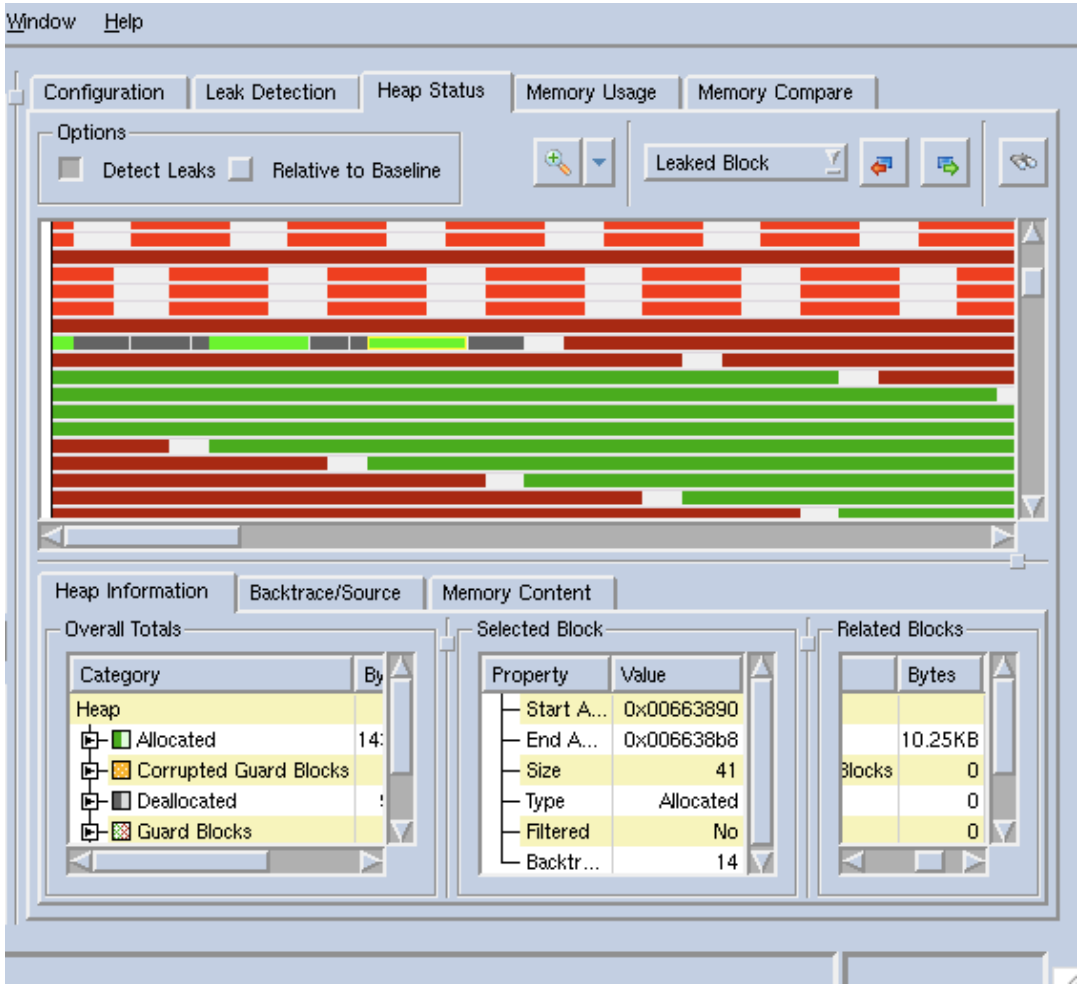


Heap Graphical View

TOTALVIEW TECHNOLOGIES



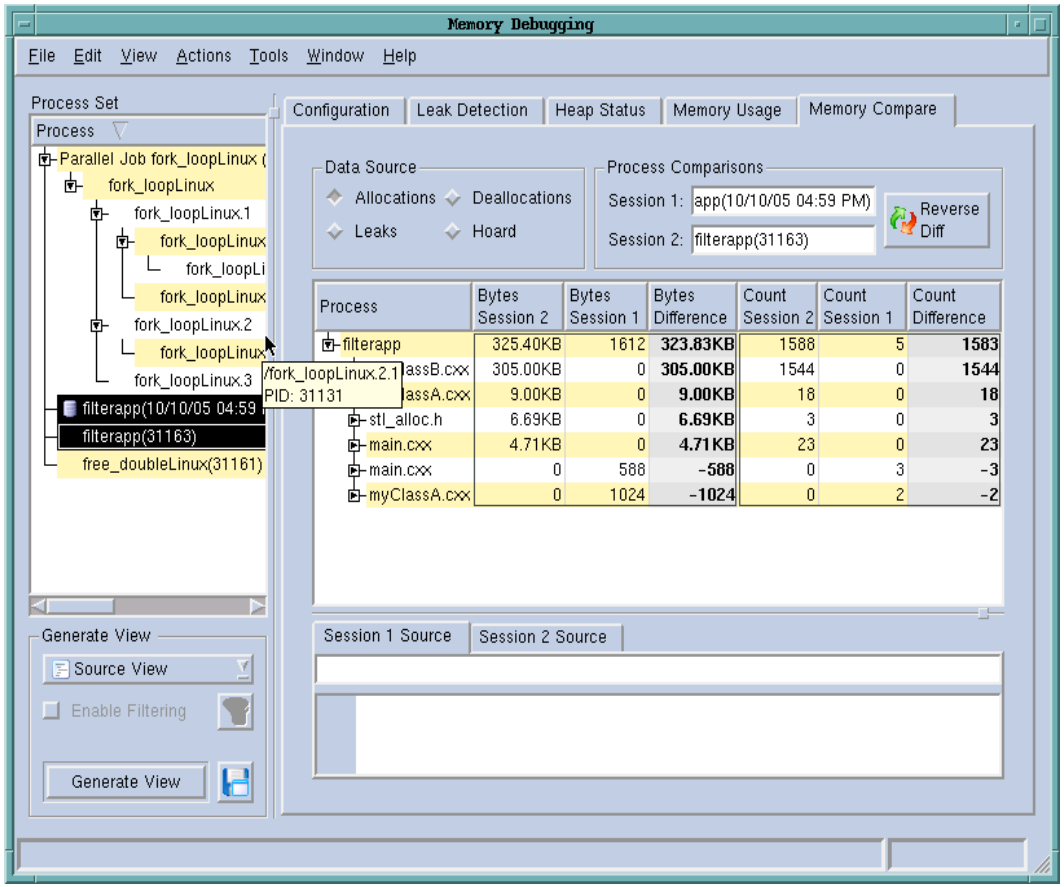
Leak Detection



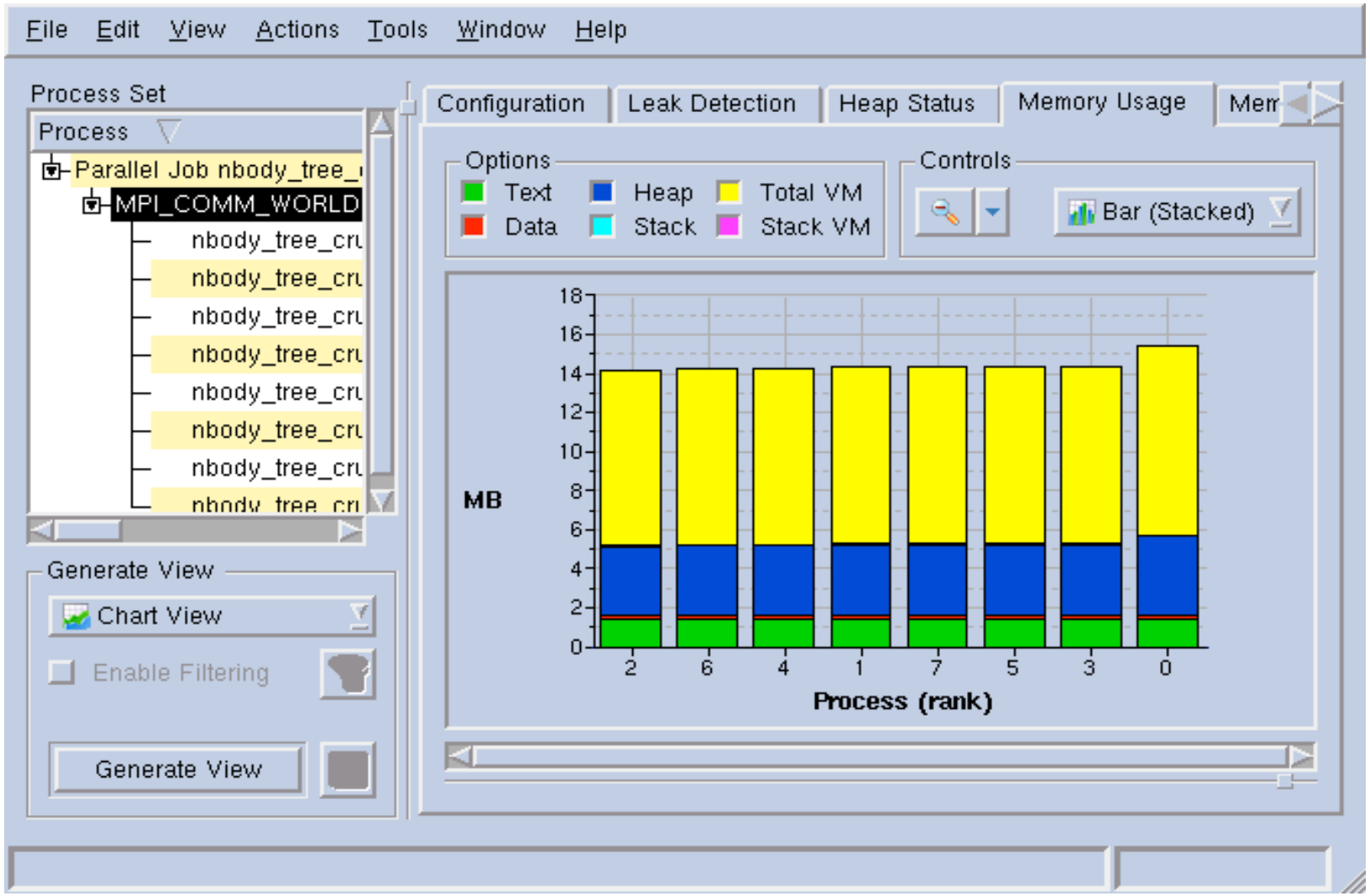
- Based on Conservative Garbage Collection
- Can be performed at any point in runtime
- Helps localize leaks in time

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

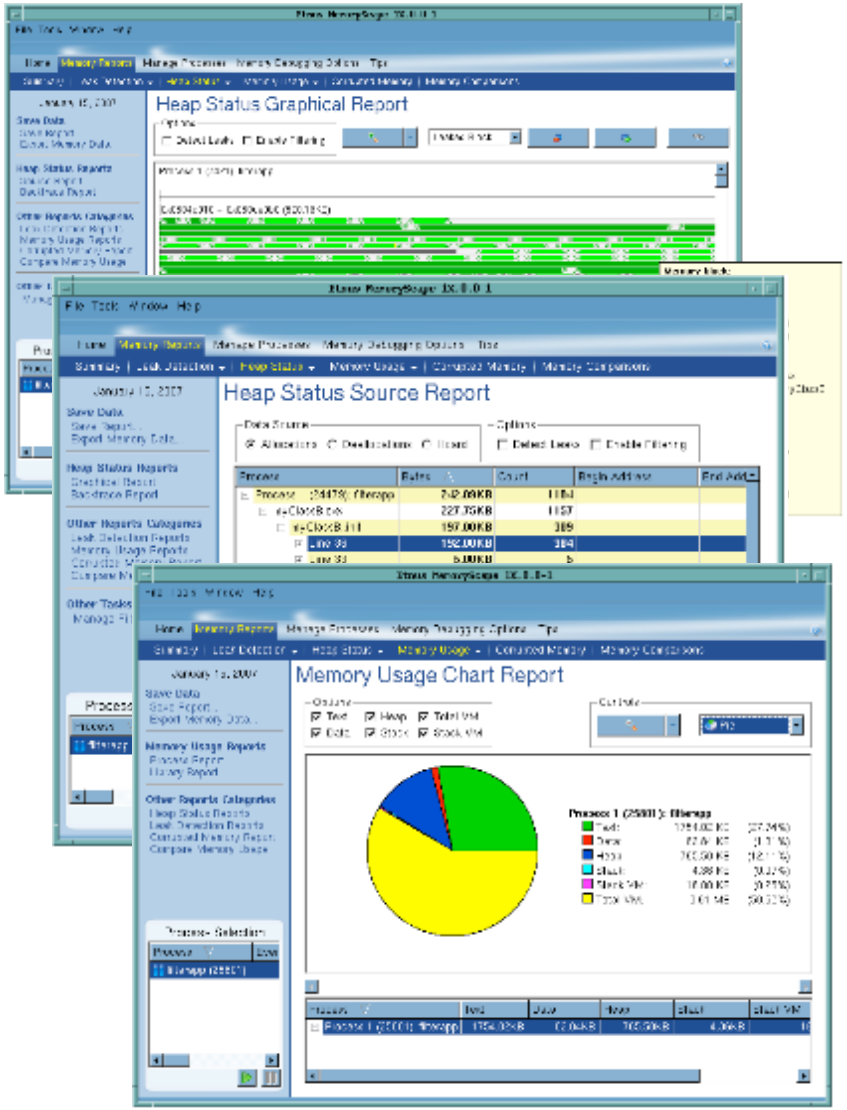


Memory Usage Statistics



Memory Reports

- **Multiple Reports**
  - Memory Statistics
  - Interactive Graphical Display
  - Source Code Display
  - Backtrace Display
  - HTML - interactive format
- **Allow the user to**
  - Monitor Program Memory Usage
  - Discover Allocation Layout
  - Look for Inefficient Allocation
  - Find Memory Leaks



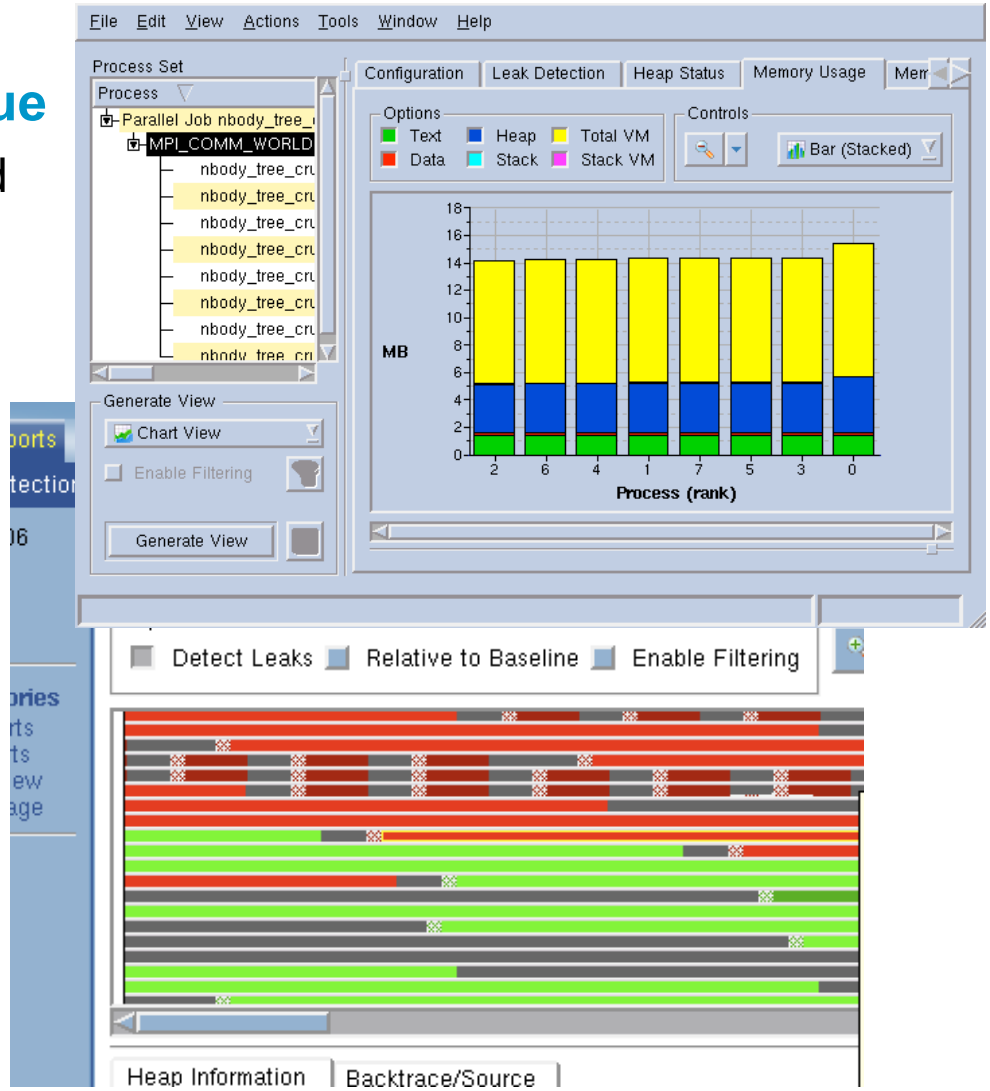
TOTALVIEW TECHNOLOGIES

Script Mode – MemScript - Tvscript

- **Automation Support**
  - Scripting lets users run tests and check programs for memory leaks without having to be in front of the program
  - Simple command line program
    - Doesn't start up the GUI
    - Can be run from within a script or test harness
  - The user defines
    - What configuration options are active
    - What thing to look for
    - Actions to be taken for each type of event that may occur

Parallel Memory Debugging

- **Memory is a growing issue**
  - Node resources are limited
  - Predicting and managing memory usage across parallel applications is complex
- **Analysis may include**
  - Comparing usage across
    - Processes of job
    - Time
    - Datasets
  - Exploring layout of allocations
  - Leak detection
  - Buffer overflow detection



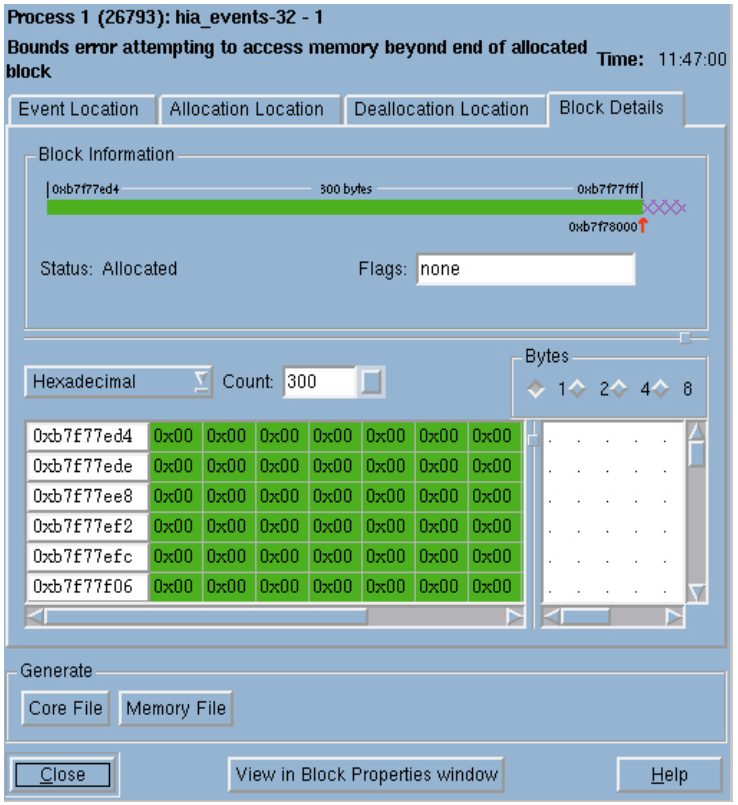
TotalView provides complimentary set of memory 'tools'

- **Guard Blocks**
  - Low runtime overhead, small size, over- and under-runs
  - Identify heap allocation bounds errors after the fact
- **HIA Events**
  - Low overhead, only catch certain types of errors
- **Memory Statistics**
  - No overhead, very high level, pick out outliers and patterns
- **Heap Graphical Display**
  - Detailed view, understand re-allocation and fragmentation behavior
- **Leak Detection**
  - Analysis of state
- **(de)allocation Hoard**
  - Helps identify dangling pointers
- **RedZones (TV 8.7, MS 2.5)**
  - Low runtime overhead, large size, over- or under-runs
  - Flags heap allocation bounds errors as they happen

Coming in TotalView 8.7 and MemoryScape 3.0

- Redzones -

- **Allocates a "protected page"**
  - adjacent to selected heap allocations
  - Before or after
- **A write into this space triggers immediate events**
  - Event occurs as the write is happening
- **Pages have a fixed size**
  - If there are many heap allocations this can potential have a large memory usage overhead
- **Ways to manage Redzones memory overhead**
  - Turn redzones on and off manually
  - Specify (by size) what allocations you want to have redzones on



Thanks!

QUESTIONS?

totalviewtech.com