Facing the challenges of

Programming weather, climate, and earth-system models on heterogeneous multi-core platforms

or

New Approaches To Debugging Complex Codes

September 7-8, 2011 at the National Center for Atmospheric Research in Boulder, Colorado

Ed Hinkel, Sales Engineer
Rogue Wave Software

ROGUE WAVE SOFTWARE

Agenda

- Introduction – Rogue Wave
- TotalView
- Approaching the Debugging Challenge
  1. TVScript – Automate Your Debugging
  2. MemoryScape – Don't Forget the Memory!
  3. Replay Engine – Shift into Reverse if You Want!
  4. CUDA – Accelerate your Development Schedule
  5. Performance – Memory Cache Usage Efficiency

ROGUE WAVE SOFTWARE

Rogue Wave Today

The largest independent provider of cross-platform software development tools and embedded components for the next generation of HPC applications

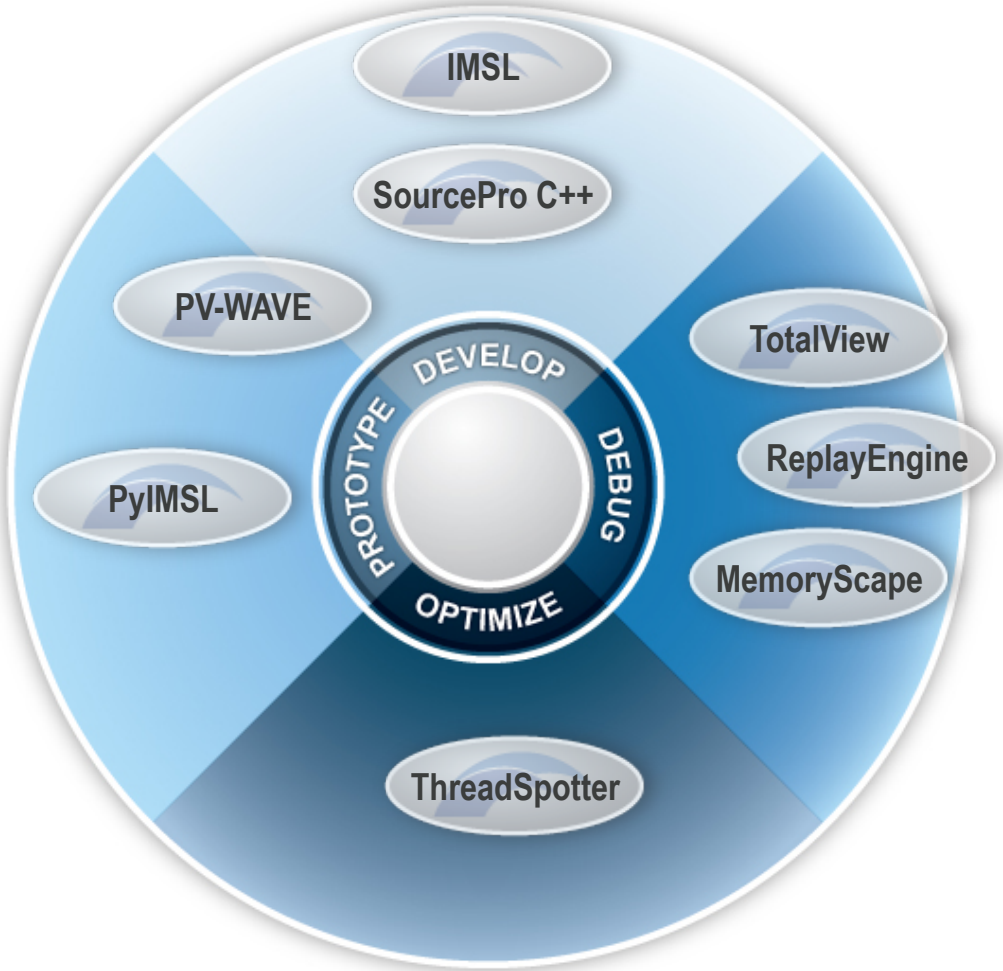
Visual Numerics®

Leader in embeddable math and statistics algorithms and visualization software for data-intensive applications.

Leading provider of intelligent software technology which analyzes and optimizes computing performance in single and multi-core environments.

Industry-leading interactive analysis and debugging tools for the world's most sophisticated software applications.

Rogue Wave Product Offerings



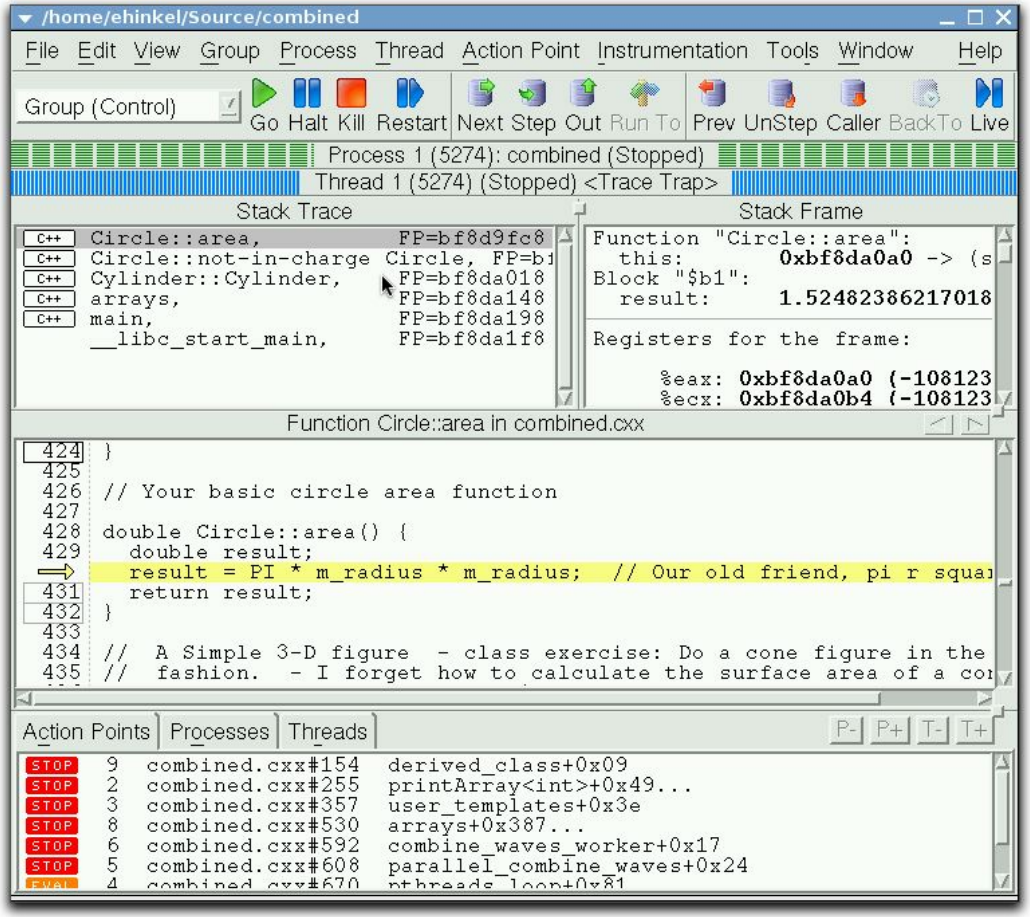
 | Copyright © 2011 Rogue Wave Software | All Rights Reserved

ROGUE WAVE SOFTWARE

What is TotalView?

A comprehensive debugging solution for demanding applications on parallel, multi-core and hybrid systems

- **Wide compiler & platform support**
  - C, C++, Fortran 77 & 90, UPC
  - Unix, Linux, OS X
- **Handles Concurrency**
  - Multi-threaded Debugging
  - Multi-process Debugging
- **Integrated Memory Debugging**
- **Reverse Debugging available**
- **Supports Multiple Usage Models**
  - Powerful and Easy GUI – Highly Graphical
  - CLI for Scripting
  - Long Distance Remote Debugging
  - Unattended Batch Debugging



Facing the Debugging Challenge

#1

Unattended Debugging

tvscript

ROGUE WAVE SOFTWARE

Unattended Debugging

- **tvscript provides for unattended, straightforward TotalView batch debugging**
  - As an adjunct to interactive debugging
  - Usable whenever jobs need to be submitted or batched
  - Provides a tool more powerful and flexible than Printf-style debugging
  - Can be used to automate test/verify environments

Think of tvscript as “Printf on steroids”!

ROGUE WAVE SOFTWARE

Debugging with TVScript

- TVScript
  - Define events
    - Breakpoints, memory errors, etc..
  - Provide actions to take in response to these events
    - Print variables or create memory reports
  - Run an MPI or serial program toward completion
    - With no user interaction
- More powerful and flexible than Printf-style debugging
  - Use to prepare and guide interactive debugging
  - Use whenever jobs need to be submitted into a managed environment
  - Can be used to automate test/verify environments

Unattended Debugging

- Using tvscript, multiple debugging sessions can be run without the need for recompiling, unlike with printf

- A single compile is all that's needed, i.e.,
  - gcc -g -o server-dbg server.c

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

ROGUE WAVE SOFTWARE

tvscript

TVscript Provides Unattended, Straightforward Batch Debugging

TVscript uses a simple, Event/Action interface	
Typical Events • Action_point • Any_memory_event • Guard_corruption error	**Typical Actions** • Display_backtrace [-level *level-num*] • List_leaks • Save_memory • Print [-slice {*slice_exp*] {*variable* | *exp*}

- A single compile is all that's needed, i.e.,

  gcc -g -o server-dbg server.c

- tvscript syntax:

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

ROGUE WAVE SOFTWARE

Unattended Debugging with Tvscript

Example

The following tells tvscript to report the contents of the *foreign_addr* structure each time the program gets to line 85

-create_actionpoint "#85=>print foreign_addr"

Typical output sample with tvscript:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! 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 = ""
!    }
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

ROGUE WAVE SOFTWARE

tvscript

Summary

- Provides process, thread and timestamp information
- Provides a single output file, even with multiple processes
- Provides many event/action descriptions, including memory debugging events
- Supports external script files, utilizing TCL within a CLI file
- Allows the generation of even more complex actions

ROGUE WAVE SOFTWARE

Facing the Debugging Challenge

#2

Memory Debugging

Don’t Forget the Memory!

ROGUE WAVE®
SOFTWARE

What is a 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

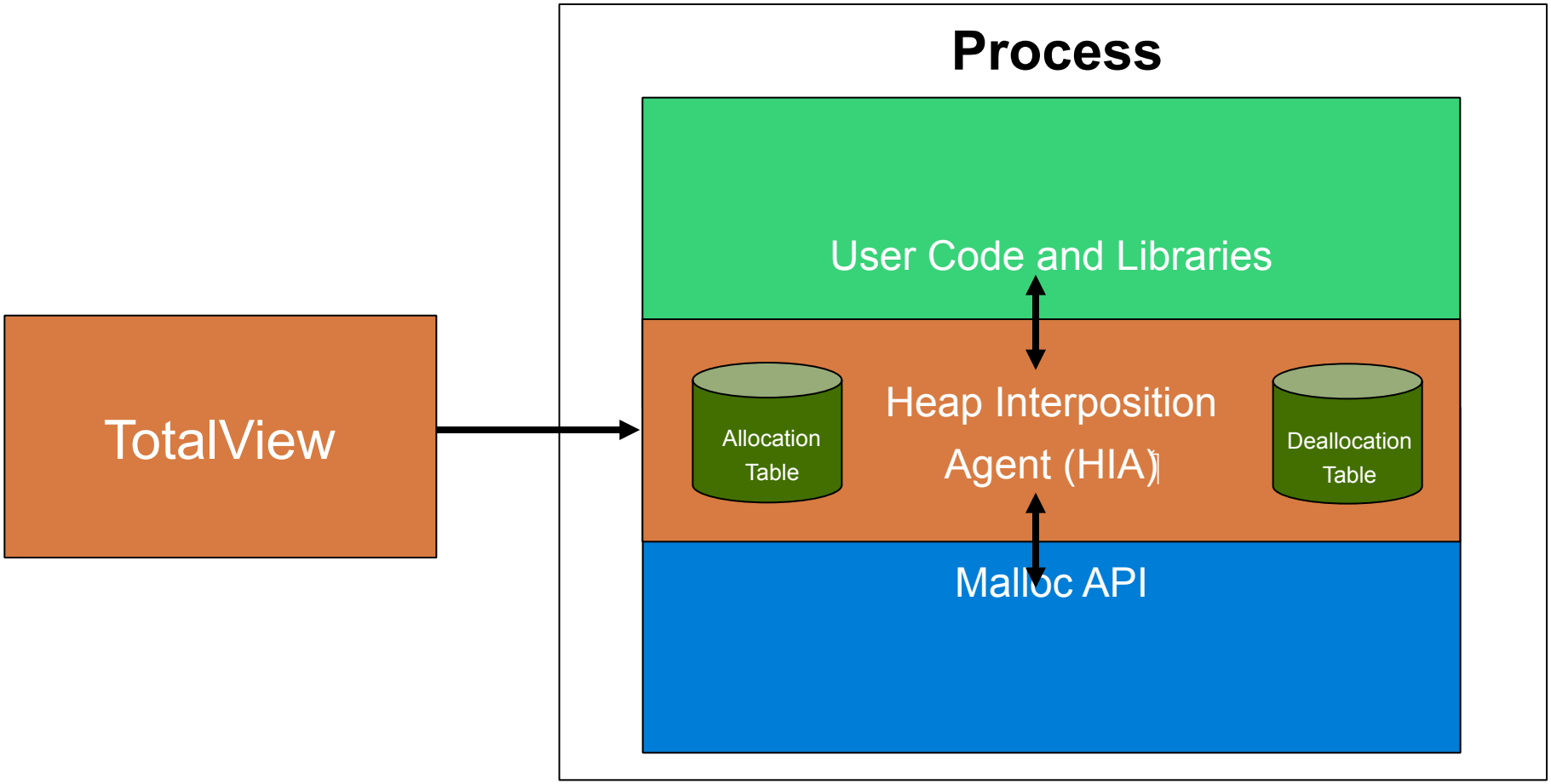
ROGUE WAVE SOFTWARE

Why Are Memory Bugs Different?

- **Memory problems can lurk**
  - For a given scale or platform or problem, they may not be fatal
  - Libraries could be source of problem
  - The fallout can occur at any subsequent memory access through a pointer
  - The mistake is rarely fatal in and of itself
  - The mistake and fallout can be widely separated
- **Potentially 'racy'**
  - Memory allocation patterns change
  - Even the fallout is not always fatal. It can result in data corruption which may or may not result in a subsequent crash

ROGUE WAVE SOFTWARE

The Agent and Interposition



ROGUE WAVE SOFTWARE

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 wide range of platforms and compilers**

ROGUE WAVE SOFTWARE

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 wide range of platforms and compilers**

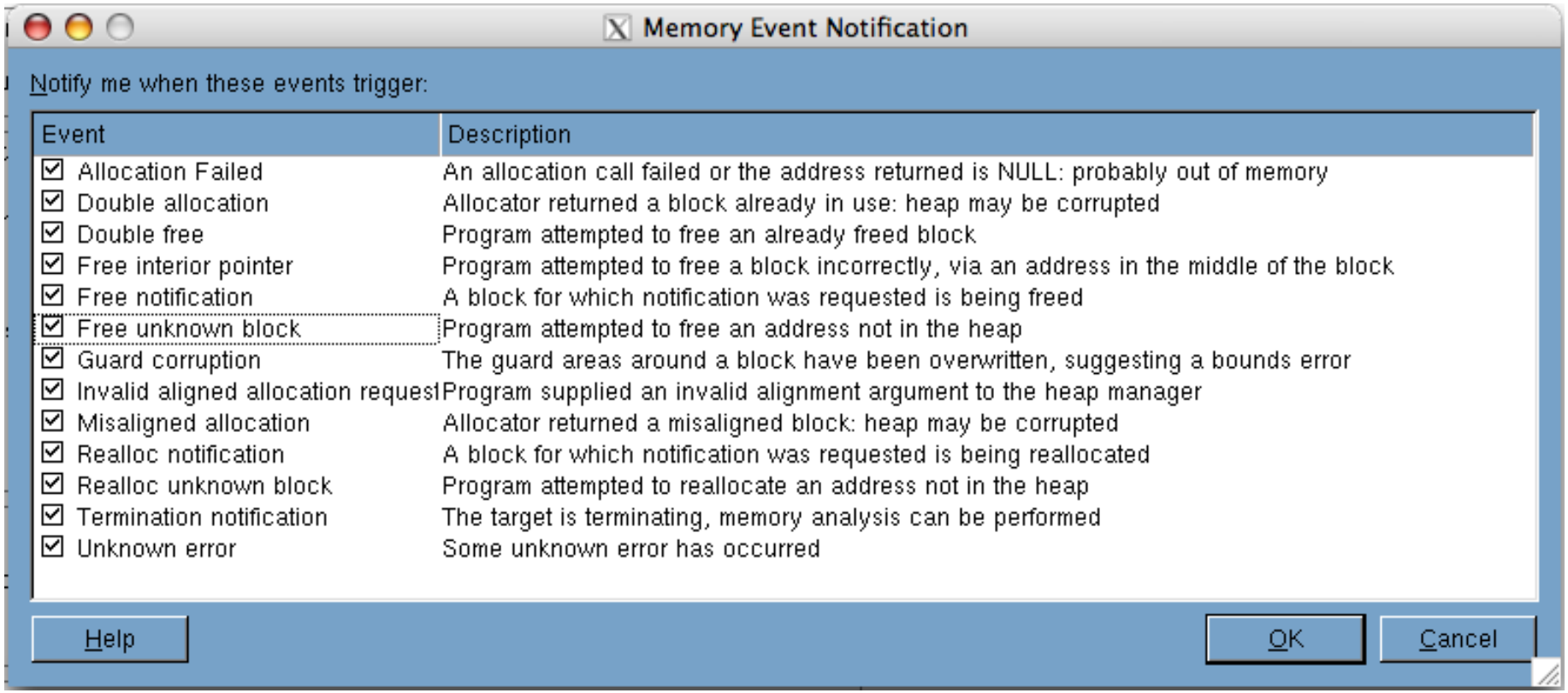
ROGUE WAVE SOFTWARE

Memory Debugger Features

- Automatically detect allocation problems
- View the heap
- Leak detection
- Block painting
- Memory Hoarding
- Dangling pointer detection
- Deallocation/reallocation notification
- Memory Corruption Detection - Guard Blocks
- Memory Comparisons between processes
- Collaboration features

ROGUE WAVE®
SOFTWARE

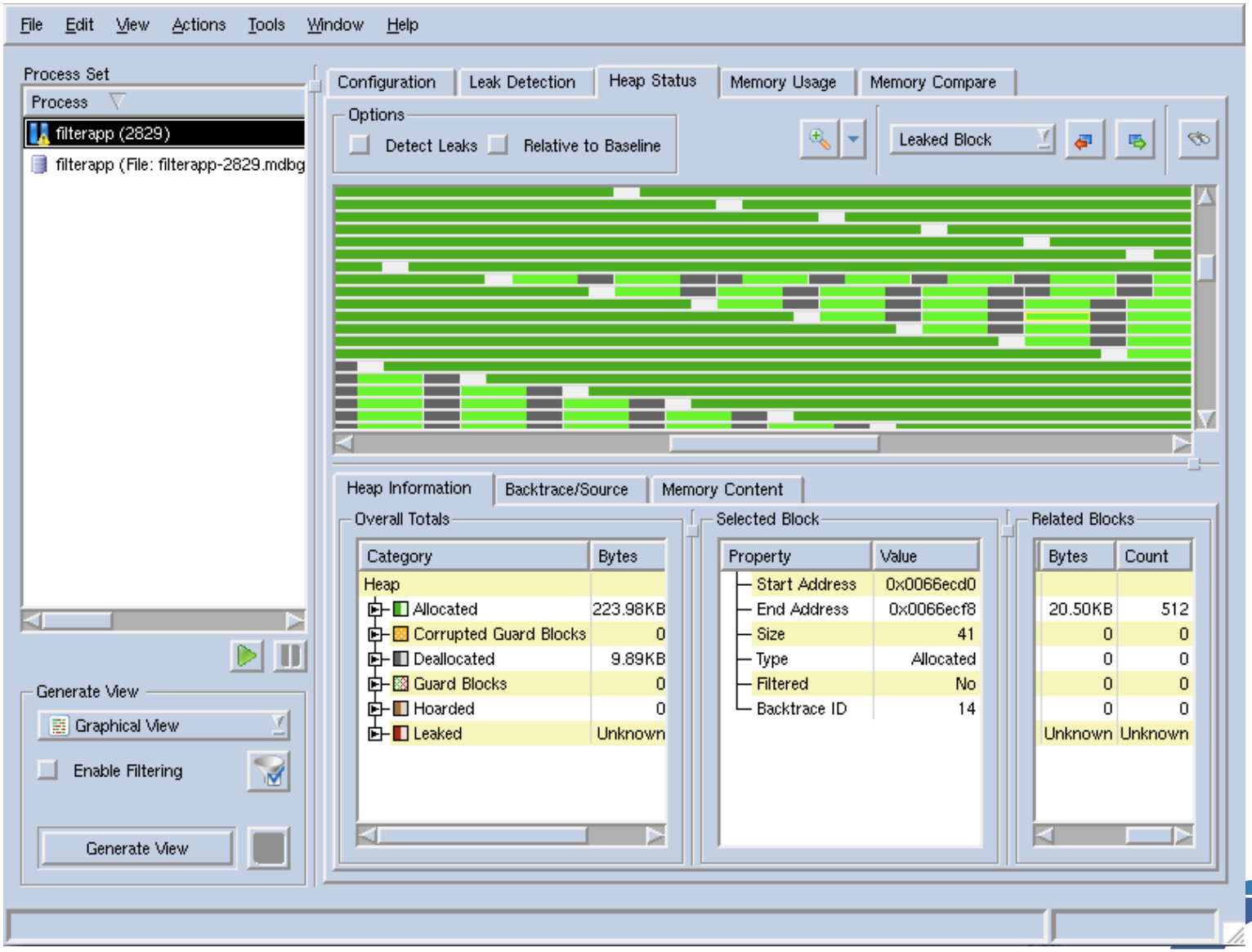
Enabling Memory Debugging
CONFIGURABILITY
Memory Event Notification



TotalView Technologies Confidential

ROGUE WAVE SOFTWARE

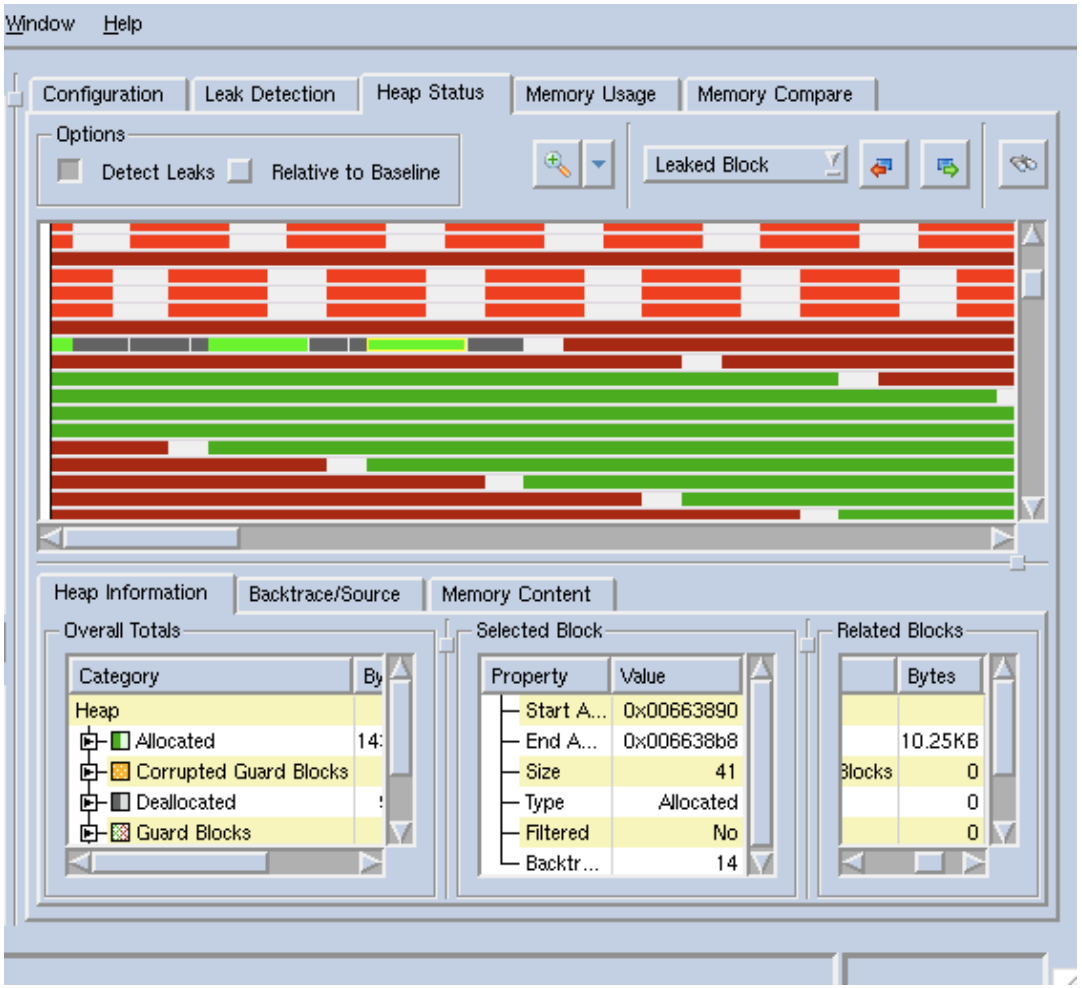
Heap Graphical View



TotalView Technologies Confidential

ROGUE WAVE SOFTWARE

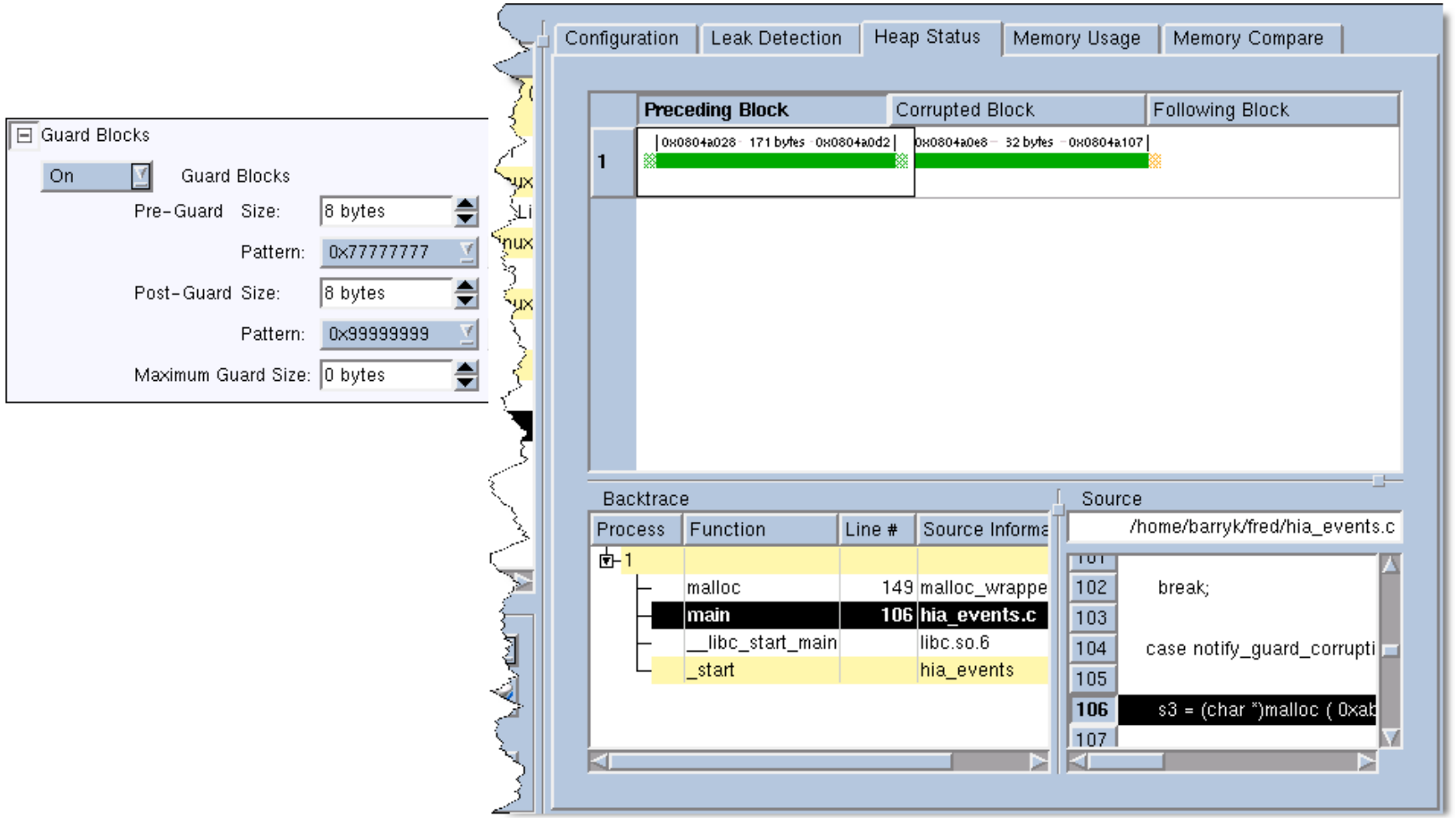
Leak Detection



- **Leak Detection**
- **Based on Conservative Garbage Collection**
- **Can be performed at any point in runtime**
  - **Helps localize leaks in time**
- **Multiple Reports**
  - **Backtrace Report**
  - **Source Code Structure**
  - **Graphically Memory Location**

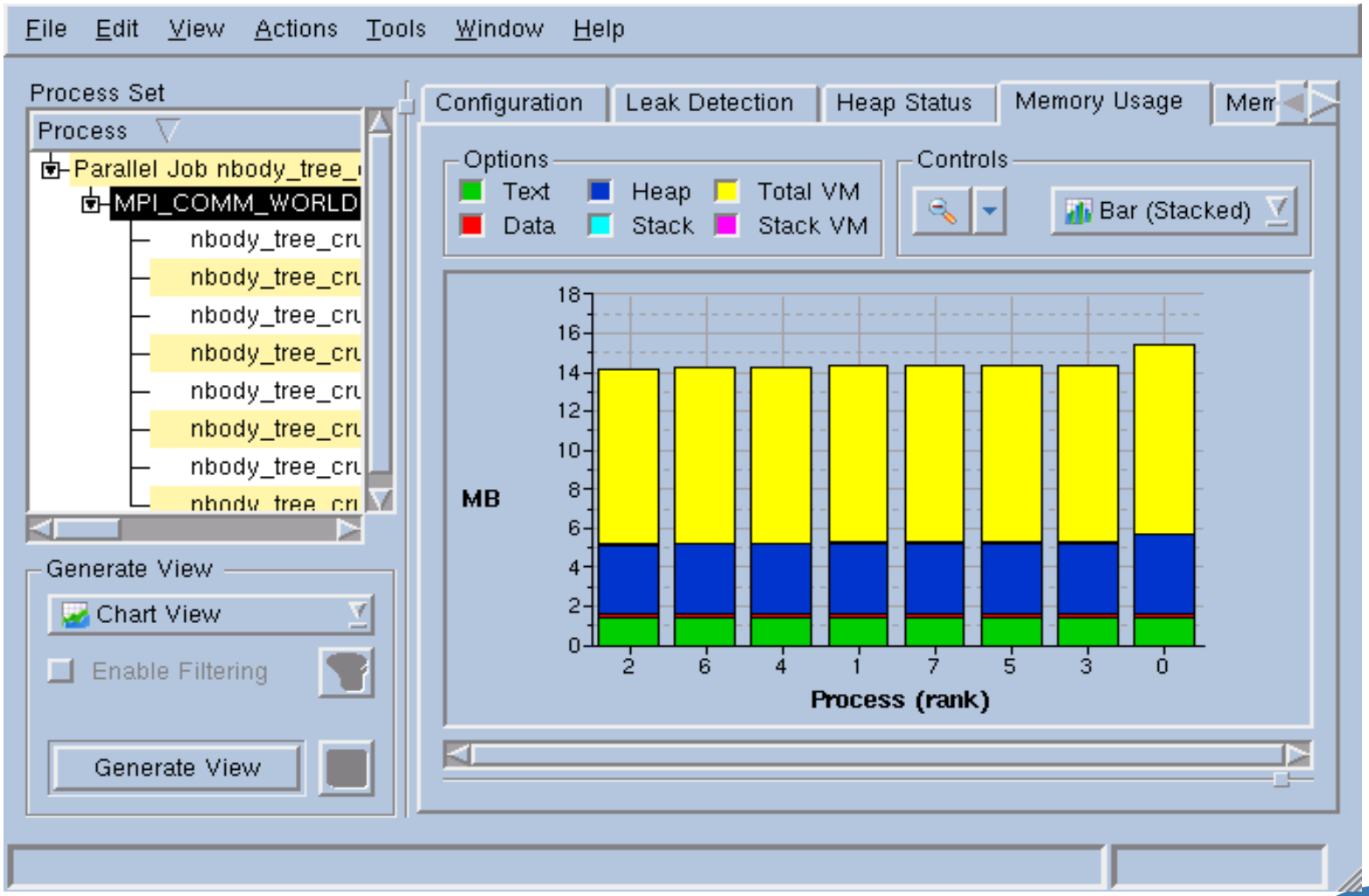
ROGUE WAVE SOFTWARE

Memory Corruption Detection (Guard Blocks)



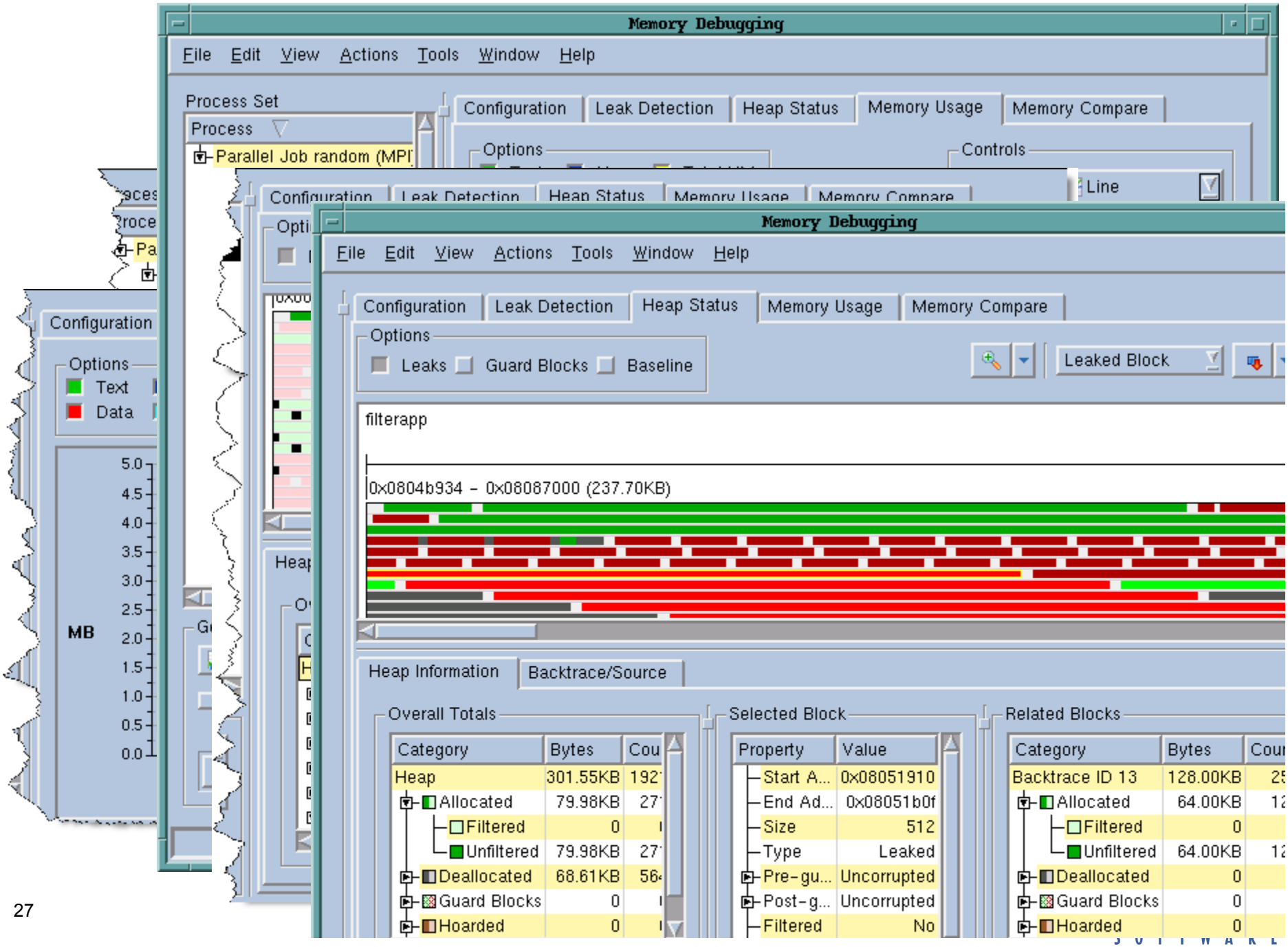
ROGUE WAVE SOFTWARE

Memory Usage Statistics



26
TotalView Technologies Confidential

ROGUE WAVE SOFTWARE


Memory Debugging
File Edit View Actions Tools Window Help
Process Set
Process
Parallel Job random (MPI
Configuration Leak Detection Heap Status Memory Usage Memory Compare
Options
Controls
Line
Memory Debugging
File Edit View Actions Tools Window Help
Configuration Leak Detection Heap Status Memory Usage Memory Compare
Options
Leaks Guard Blocks Baseline
Leaked Block
filterapp
0x0804b934 - 0x08087000 (237.70KB)
Heap Information Backtrace/Source
Overall Totals
Category Bytes Cou
Heap 301.55KB 192
Allocated 79.98KB 27
Filtered 0
Unfiltered 79.98KB 27
Deallocated 68.61KB 56
Guard Blocks 0
Hoarded 0
Selected Block
Property Value
Start A... 0x08051910
End Ad... 0x08051b0f
Size 512
Type Leaked
Pre-gu... Uncorrupted
Post-g... Uncorrupted
Filtered No
Related Blocks
Category Bytes
Backtrace ID 13 128.00KB
Allocated 64.00KB
Filtered 0
Unfiltered 64.00KB
Deallocated 0
Guard Blocks 0
Hoarded 0
Configuration
Options
Text
Data
MB
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Facing the Debugging Challenge

#3

ROGUE WAVE SOFTWARE

ReplayEngine

Reverse Debugging

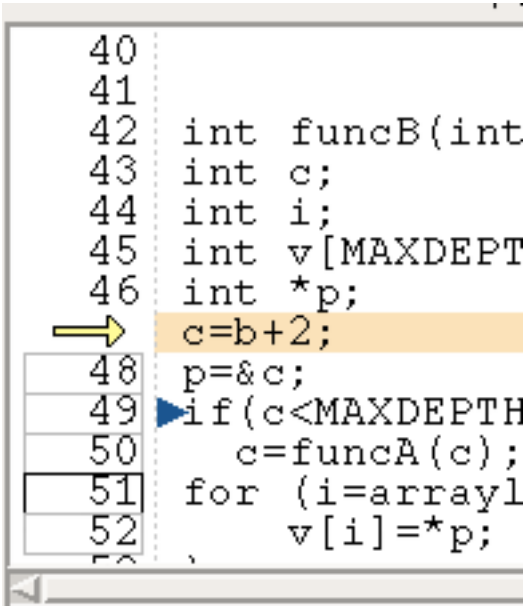
29
TotalView Technologies Confidential

ROGUE WAVE SOFTWARE

What is ReplayEngine?

A separately licensed add-on product that providing reverse debugging

- **Enhances debugging experience**
  - Add-on to TotalView
- **Captures execution history**
  - Record all external input to program
  - Records internal sources of non-determinism
- **Replays execution history**
  - Examine any part of the execution history
  - Step as easily back through code as you do forwards
  - Jump to points of interest
- **Simple extension to TotalView**
  - No recompilation or instrumentation
  - The user just says where they want to go
  - Explore data and state in the past just like a live process
- **Supported on Linux x86 and x86-64**
- **Supports MPI, Pthreads, and OpenMP**



ROGUE WAVE SOFTWARE

Reverse Debugging - ReplayEngine

ReplayEngine: Debug in Forward and Reverse!

ReplayEngine's deterministic replay capability records the execution history of your program and allows bi-directional diagnosis and debugging with TotalView.

- **Step freely forward *and* backwards through a program's execution**
- **Use Breakpoints and Watchpoints from either direction**
- **Examine historical data and execution paths *deterministically***
- **Works with integrated memory debugging**
- **Allows for "rolling history capture" of long-running programs**
  - Stop at out of memory
  - Set maximum history buffer size

ReplayEngine is a separately licensed product add-on that extends the capabilities of TotalView on Linux-86 and Linux-86-64 machines.

ROGUE WAVE SOFTWARE

ReplayEngine

An Intuitive User Interface

Step forward over functions

Step *backward* over functions

Step forward into functions

Step *backward* into functions

Advance forward out of current Function, after the call

Advance backward out of current Function, to before the call

Advance forward to selected line

Advance backward to selected line

Advance forward to “live” session

ROGUE WAVE SOFTWARE

ReplayEngine

No Replay? (or not running)

No Active Buttons!

ROGUE WAVE SOFTWARE

Facing the Debugging Challenge

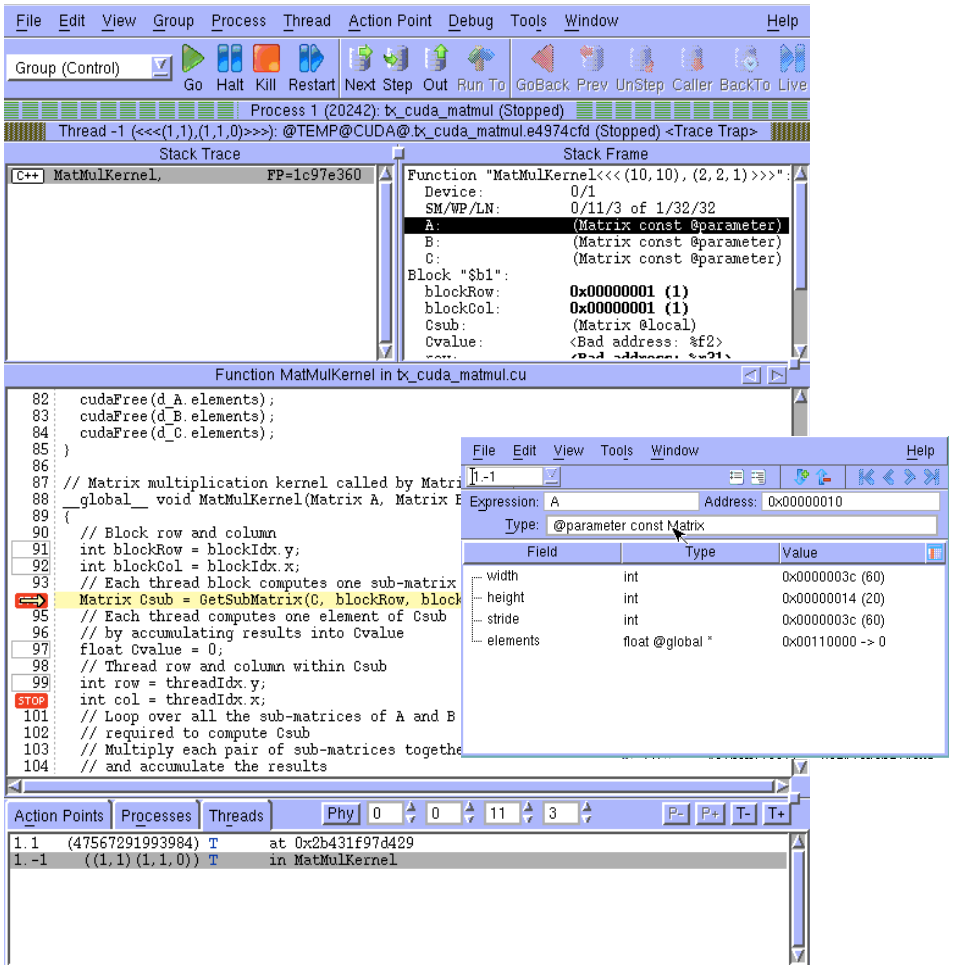
#4

ROGUE WAVE SOFTWARE

CUDA Debugging

ROGUE WAVE SOFTWARE

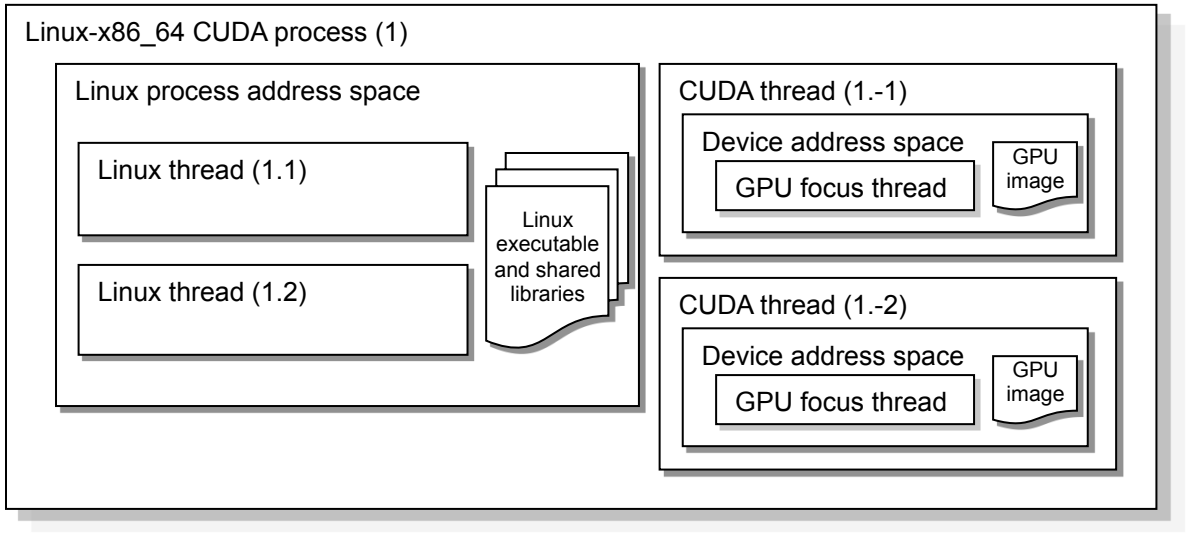
CUDA Debugging TotalView



- **Characteristics**
  - Debugging of application running on the GPU device (not in an emulator)
  - Full visibility of both Linux threads and GPU device threads
    - Device threads shown as part of the parent Unix process
    - Correctly handle all the differences between the CPU and GPU
  - Fully represent the hierarchical memory
    - Display data at any level (registers, local, block, global or host memory)
    - Making it clear where data resides with type qualification
  - Thread and Block Coordinates
    - Built in runtime variables display threads in a warp, block and thread dimensions and indexes
    - Displayed on the interface in the status bar, thread tab and stack frame
  - Device thread control
    - Warps advance Synchronously
  - Handles CUDA function inlining
    - Step in to or over inlined functions
  - Reports memory access errors
    - CUDA memcheck
  - Can be used with MPI

ROGUE WAVE SOFTWARE

TotalView CUDA Debugging Model



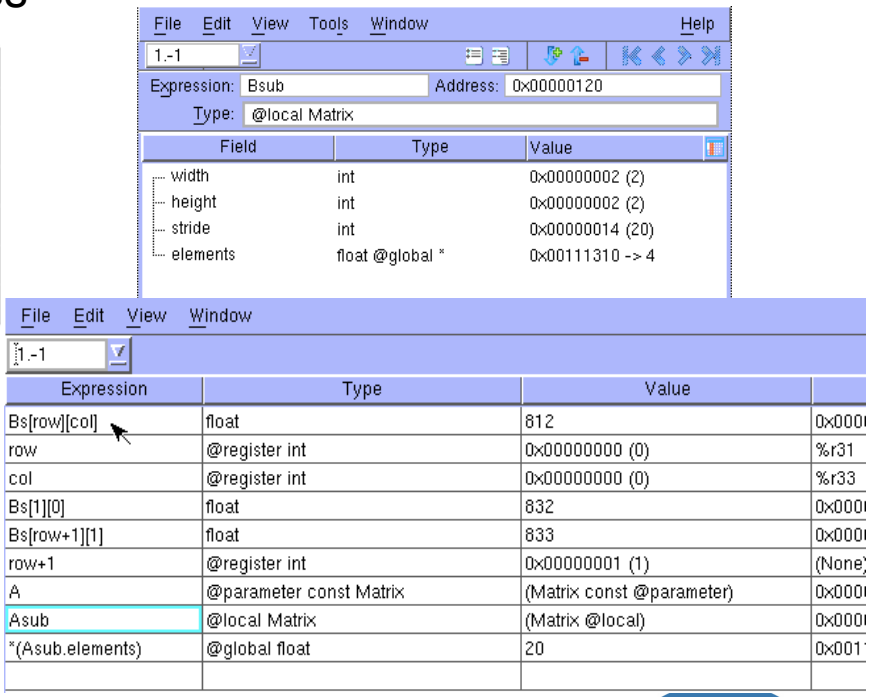
- **A Linux-x86_64 CUDA process consists of:**
  - **A Linux process address space, containing:**
    - **A Linux executable and a list of Linux shared libraries.**
  - **A collection of Linux threads, where a Linux thread:**
    - **Is assigned a positive debugger thread ID.**
  - **A collection of CUDA threads, where a CUDA thread:**
    - **Is assigned a negative debugger thread ID.**
    - **Has its own separate address space**

ROGUE WAVE SOFTWARE

Storage Qualifiers

- **Denotes location in hierarchical memory**
  - Part of the type – using "@" notation
  - Each memory space has a separate address space so 0x00001234 could refer to several places

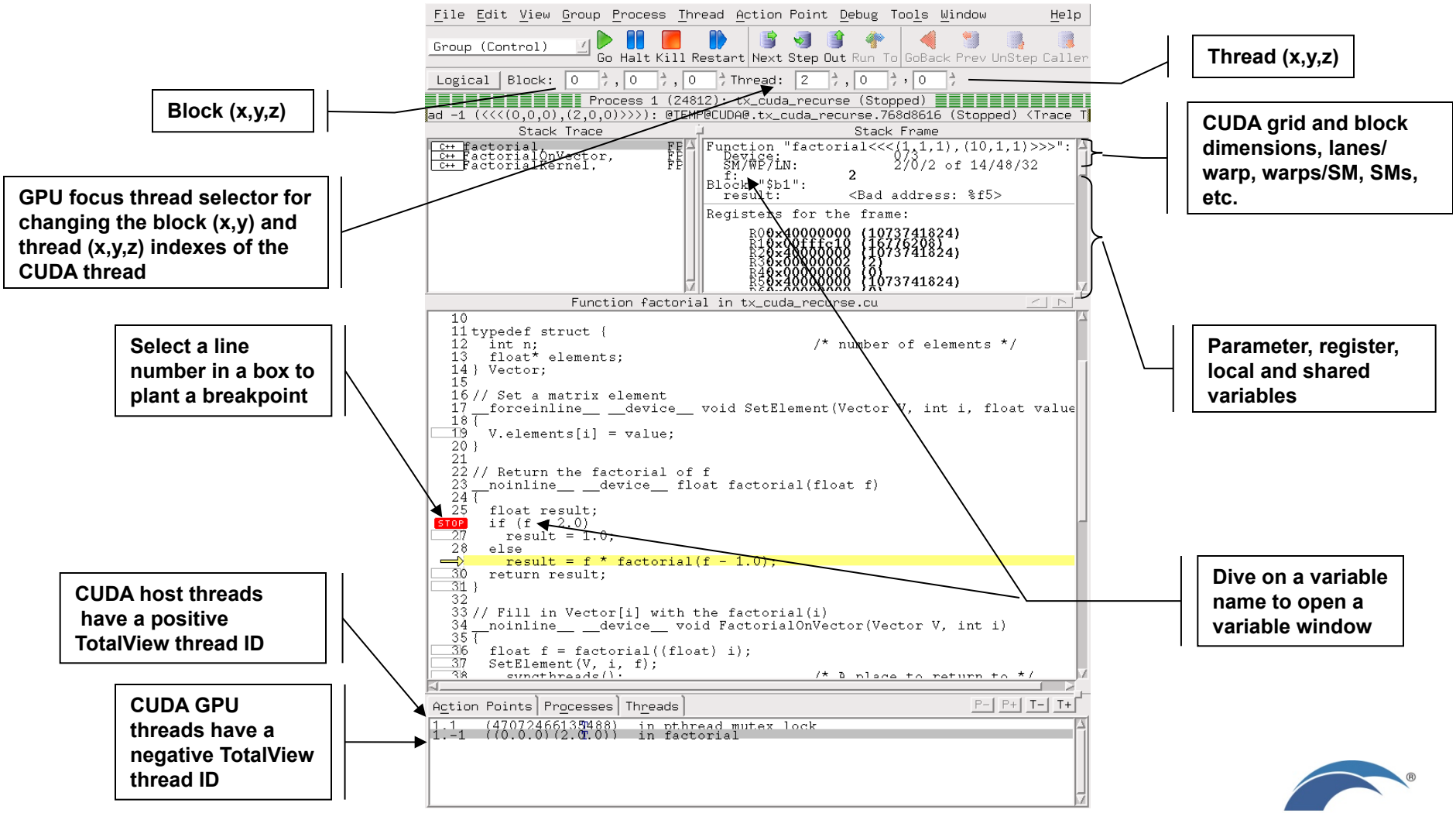
Storage Qualifier	Meaning
@parameter	Address is an offset within parameter storage.
@local	Address is an offset within local storage.
@shared	Address is an offset within shared storage.
@constant	Address is an offset within constant storage.
@global	Address is an offset within global storage.
@register	Address is a PTX register name



- **Used throughout expression system**
  - You can cast to switch between different spaces

ROGUE WAVE SOFTWARE

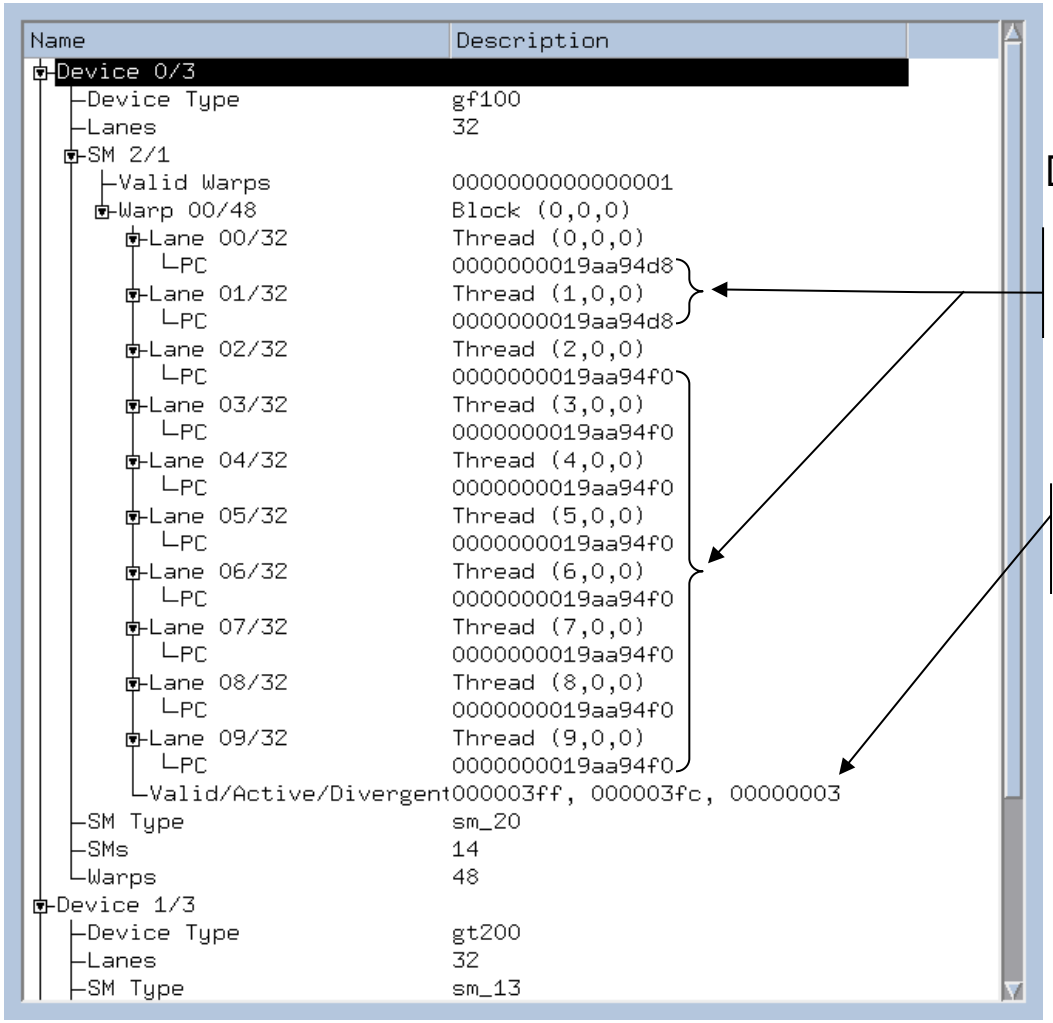
Debugging CUDA



 | Copyright © 2010 Rogue Wave Software | All Rights Reserved

GPU Device Status Display

- **Display of PCs across SMs, Warps and Lanes**
- **Updates as you step**
- **Shows what hardware is in use**
- **Helps you map between logical and hardware coordinates**



Example of Divergent GPU threads

Different PC for two groups of Lanes

State of Lanes inside the warp

| Copyright © 2010 Rogue Wave Software | All Rights Reserved

ROGUE WAVE SOFTWARE

Facing the Debugging Challenge

#5

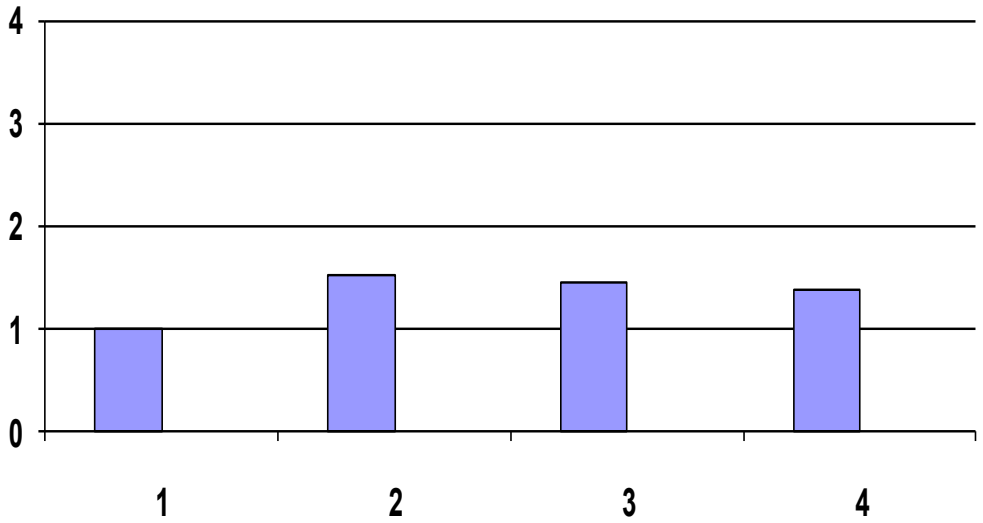
ROGUE WAVE SOFTWARE

ThreadSpotter cache memory optimization tool

- Analyzes memory bandwidth and latency, data locality and thread communications
- Identifies specific issues and pinpoints troublesome areas in source code
- Provides guidance towards a resolution
- Increases productivity for experts and non-experts

ROGUE WAVE SOFTWARE

Example: Why do more cores = less performance?



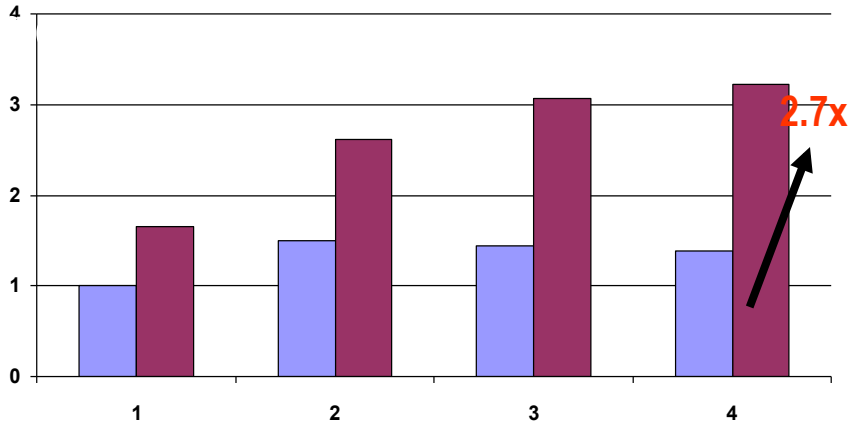
Poor parallelism?

Actually, this program is “embarrassingly parallel”

Poor memory usage ➔ super-linear slowdown

ROGUE WAVE SOFTWARE

The Same Application Optimized



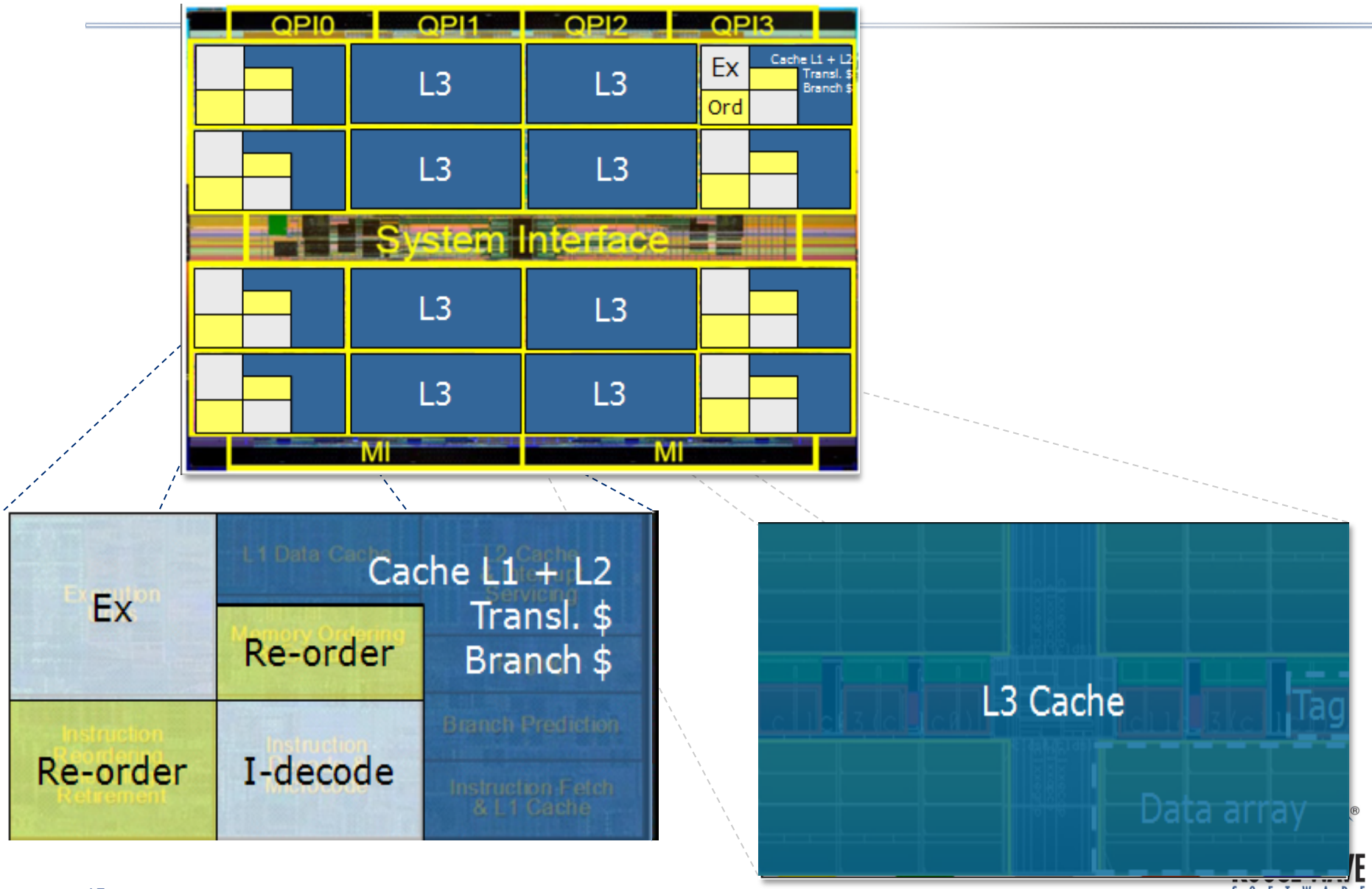
Optimization can be rewarding, but costly…

- Often requires expert knowledge
- Typically weeks of wading through performance data

➔ ThreadSpotter's one-click advice: Change one line

ROGUE WAVE SOFTWARE

How is the silicon used?



A rule of thumb

Memory system level	Relative latency
L1 cache	1x
Higher cache levels	10x
Main memory	100x

Source: AMD, *Michael Wall*

ROGUE WAVE SOFTWARE

Simple modifications can make a big difference

Program A

```
struct DATA
{
    int a;
    int b;
    int c;
    int d;
};
DATA * pMyData;

for (long i=0; i<10*1024*1024; i++)
{
    pMyData[i].a = pMyData[i].b;
}
```

Program B

```
struct DATA
{
    int a;
    int b;
};


DATA * pMyData;

for (long i=0; i<10*1024*1024; i++)
{
    pMyData[i].a = pMyData[i].b;
}
```

Partially Used Structures

 | Copyright © 2010 Rogue Wave Software | All Rights Reserved

ROGUE WAVE SOFTWARE

Partially Used Structures

Defined data structure includes a,b,c,d… but only uses a & b

50%

Redefined data structure includes a,b,c,d

100%

ROGUE WAVE SOFTWARE

Alignment Characteristics

```
#////////////////////////////////////////////////////
struct DATA
{
        char a;
        int b;
        char c;
};
DATA * pMyData;
//////////////////////////////////////////////////////
virtual void Initialize(int nbThread)
{ pMyData = new DATA[36*1024*1024]; }
//////////////////////////////////////////////////////
```

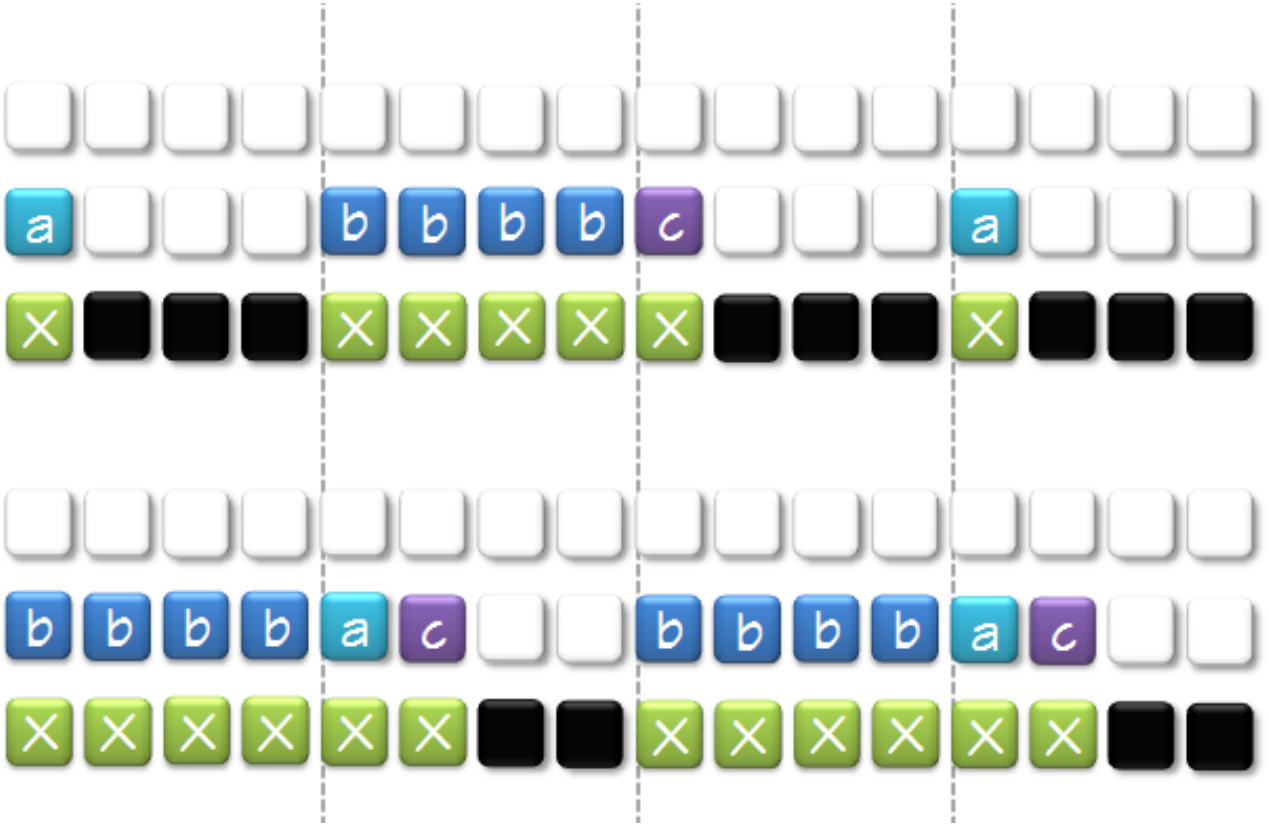
```
////////////////////////////////////////////////////
struct DATA
{
        int b;
        char a;
        char c;
};
DATA * pMyData;
////////////////////////////////////////////////////
virtual void Initialize(int nbThread)
{ pMyData = new DATA[36*1024*1024]; }
```

What's the difference?

 | Copyright © 2010 Rogue Wave Software | All Rights Reserved

ROGUE WAVE SOFTWARE

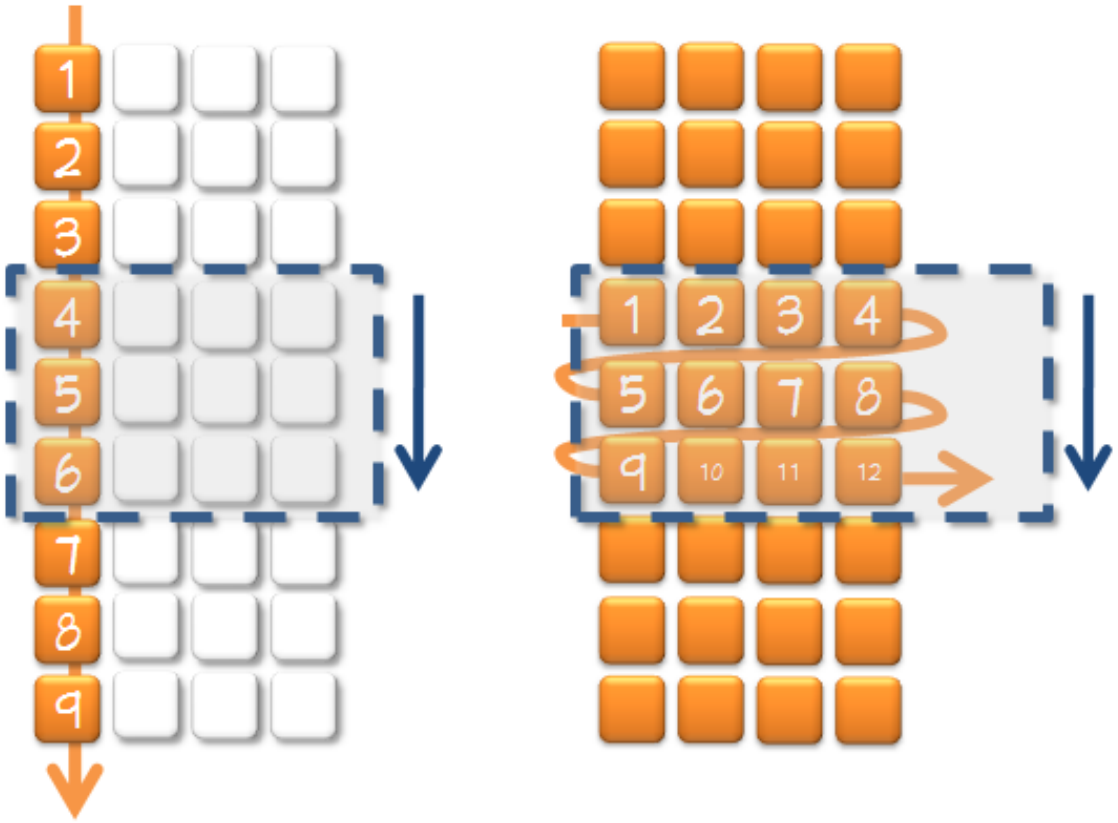
Alignment Characteristics



ROGUE WAVE SOFTWARE

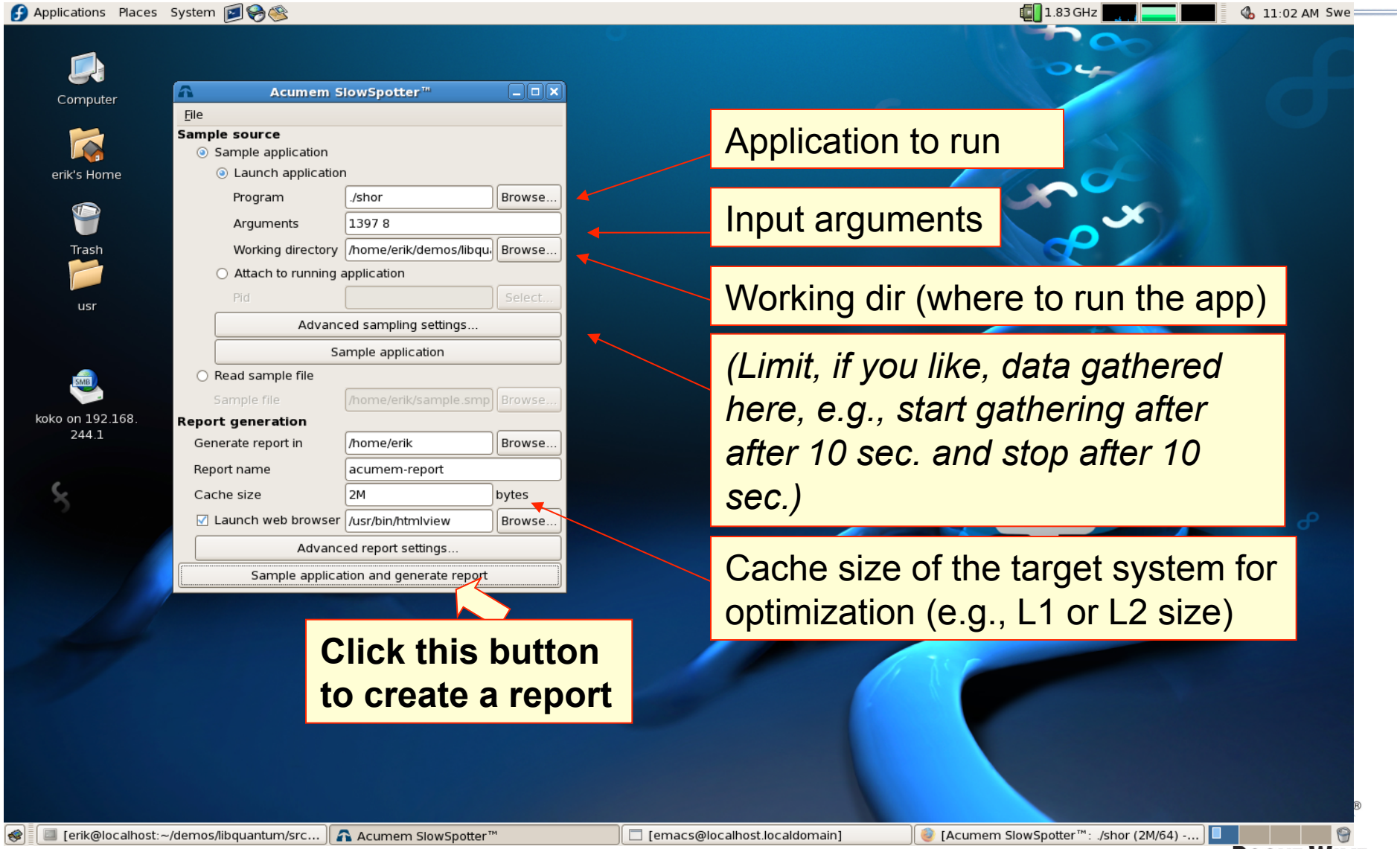
Test 4: Inefficient Loop Nesting

Explanation



ROGUE WAVE SOFTWARE

A One-Click Report Generation



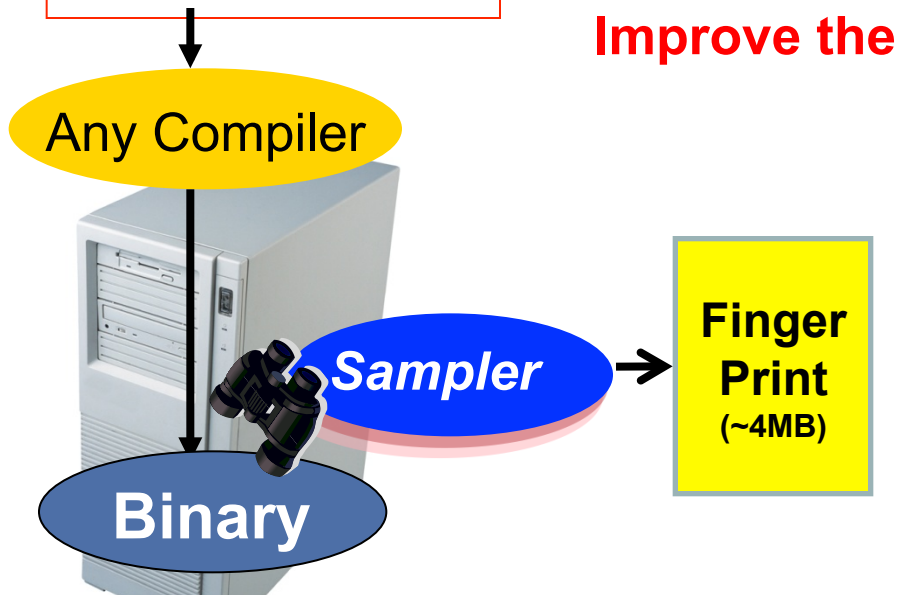
ROGUE WAVE SOFTWARE

Acumem SlowSpotter™

Source:

C, C++, Fortran, OpenMP…

```
/* Unoptimized  Array Multiplication: x = y * z    N = 1024 */
for (i = 0; i < N; i = i + 1)
   for (j = 0; j < N; j = j + 1)
      {r = 0;
      for (k = 0; k < N; k = k + 1)
         r =  r + y[i][k] * z[k][j];
      x[i][j] = r;
      }
/* Unoptimized  Array Multiplication: x = y * z    N = 1024 */
for (i = 0; i < N; i = i + 1)
   for (j = 0; j < N; j = j + 1)
      {r = 0;
      for (k = 0; k < N; k = k + 1)
         r =  r + y[i][k] * z[k][j];
      x[i][j] = r;
      }
```



Mission:

Find the SlowSpots™

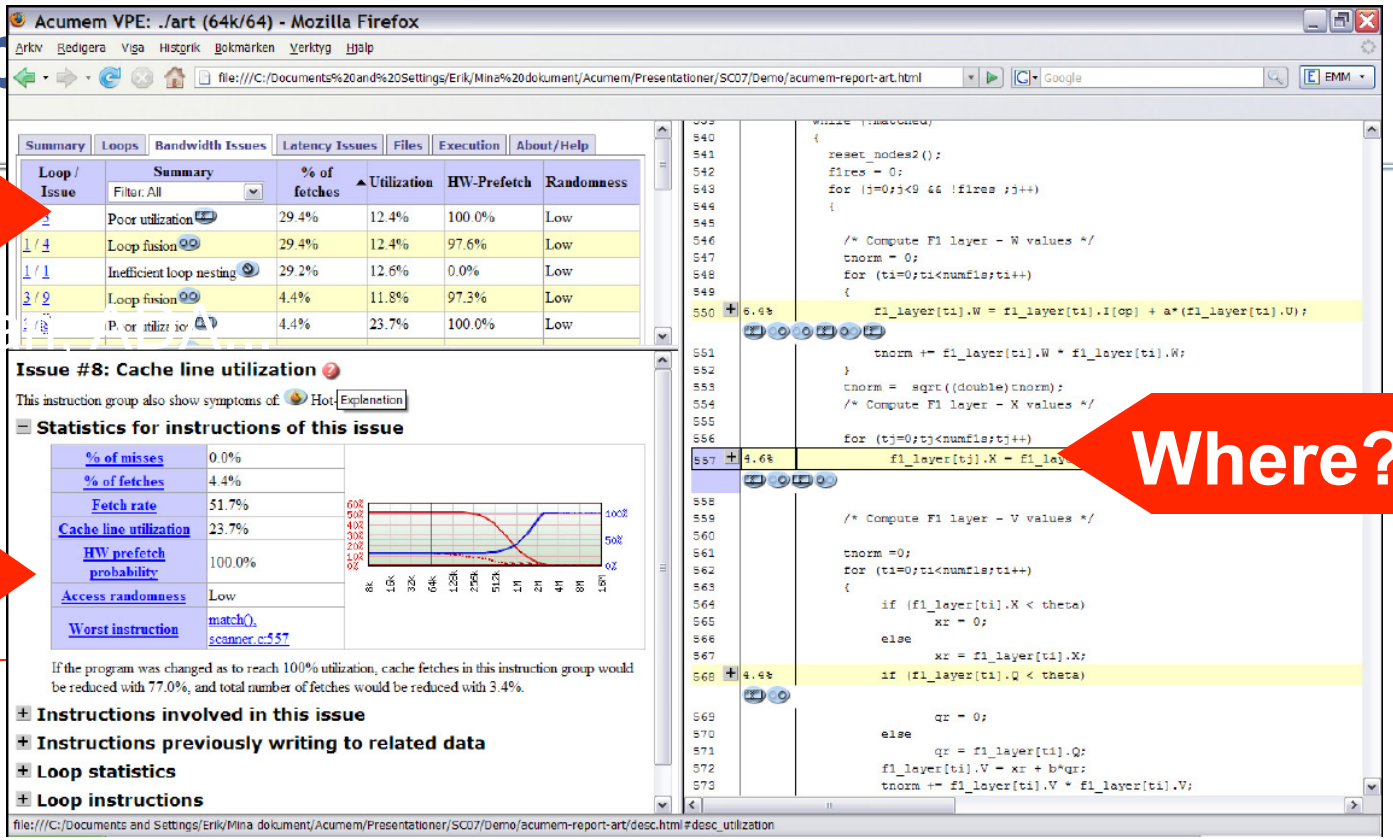
Assess their importance

Enable for non-experts to fix them

Improve the productivity of experts

ROGUE WAVE SOFTWARE

ThreadSp



What?

How?

Where?

Any Compiler

Binary

Sampler

Finger Print (~4MB)

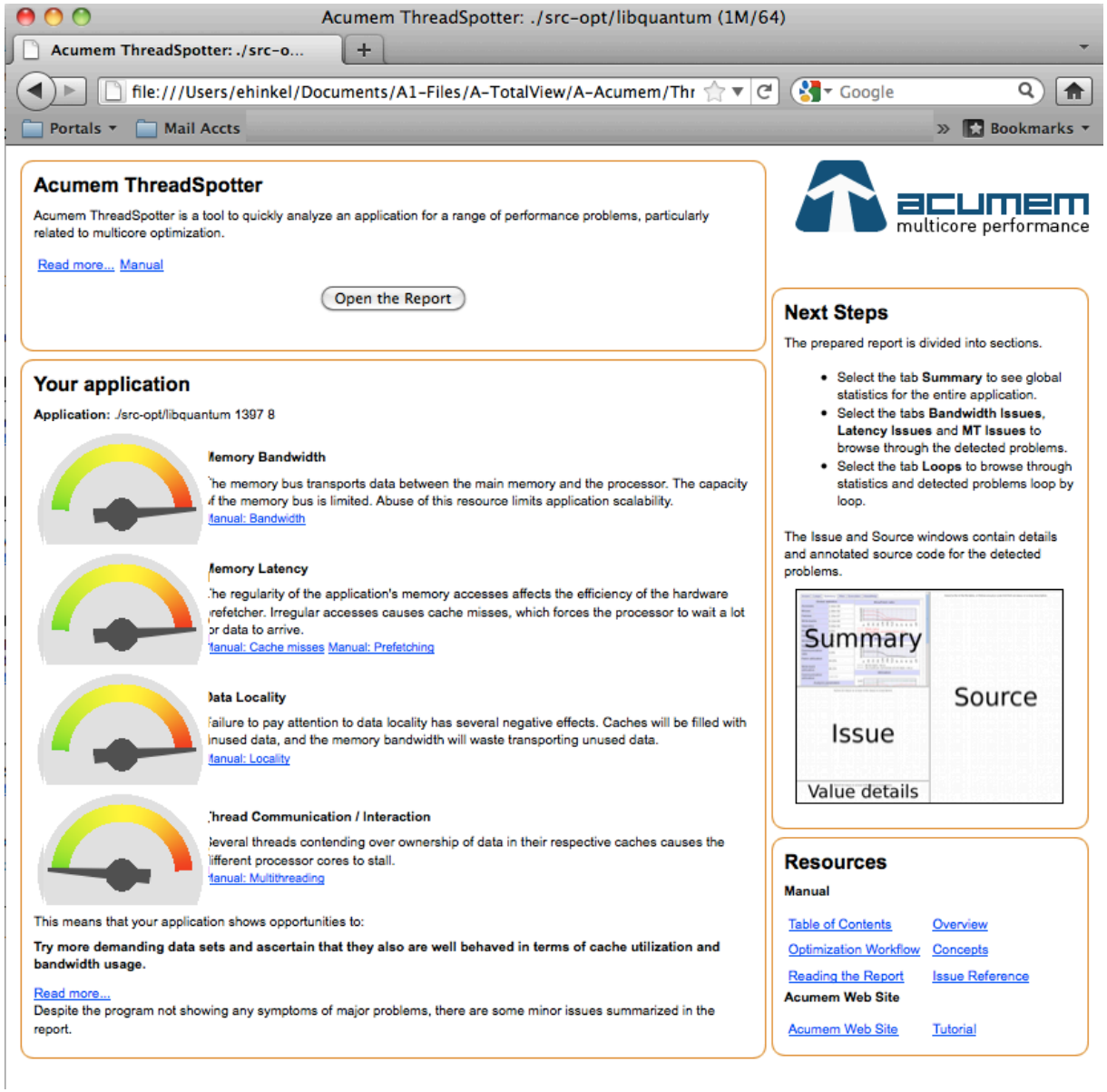
Model

Advice

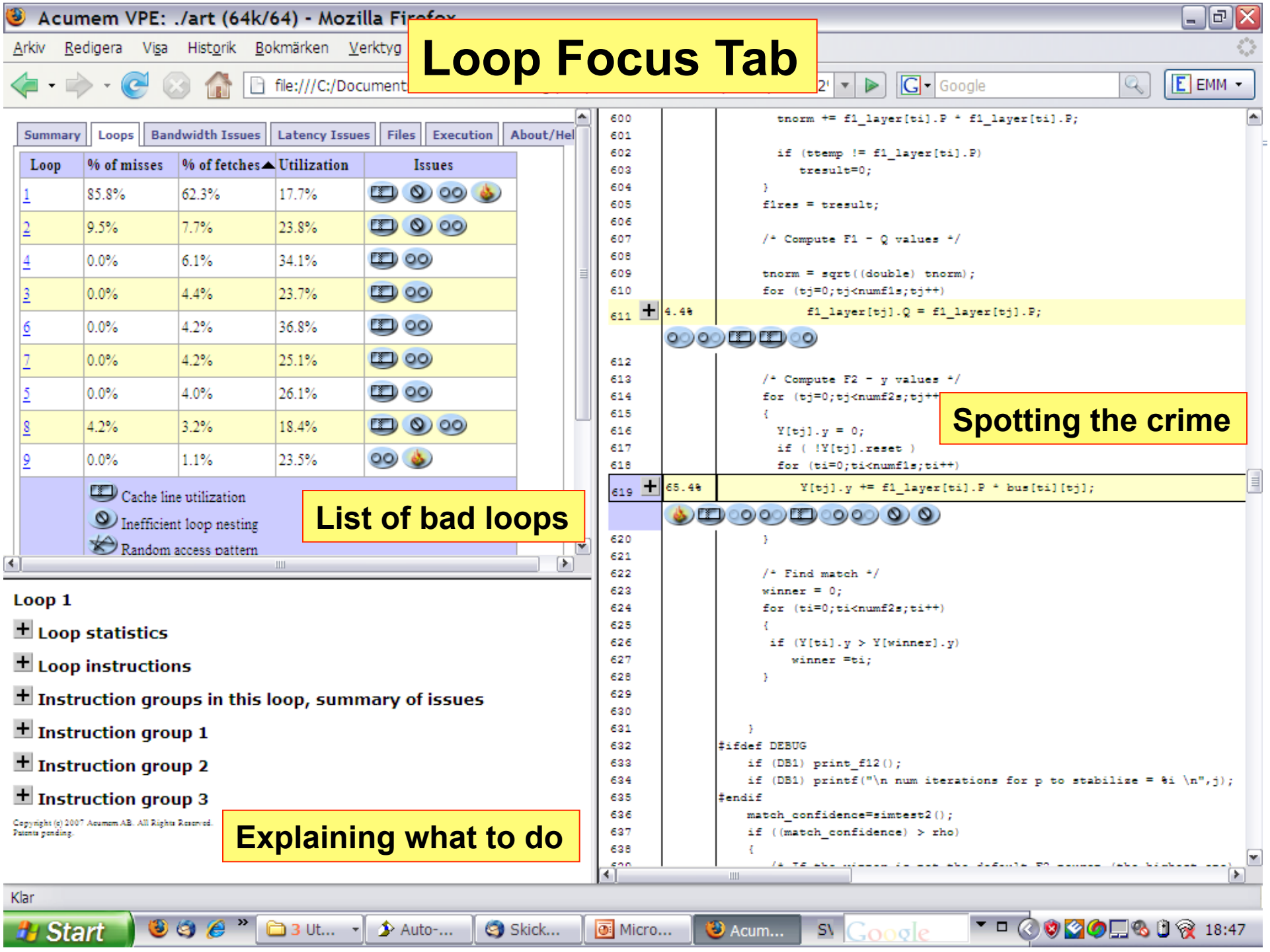
Target System Parameters

ROGUE WAVE SOFTWARE

Frontpage



ROGUE WAVE SOFTWARE


Acumem VPE: ./art (64k/64) - Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg
Loop Focus Tab
Summary
Loops
Bandwidth Issues
Latency Issues
Files
Execution
About/Hel
Loop
% of misses
% of fetches
Utilization
Issues
1 85.8% 62.3% 17.7%
2 9.5% 7.7% 23.8%
4 0.0% 6.1% 34.1%
3 0.0% 4.4% 23.7%
6 0.0% 4.2% 36.8%
7 0.0% 4.2% 25.1%
5 0.0% 4.0% 26.1%
8 4.2% 3.2% 18.4%
9 0.0% 1.1% 23.5%
Cache line utilization
Inefficient loop nesting
Random access pattern
List of bad loops
Spotting the crime
Loop 1
Loop statistics
Loop instructions
Instruction groups in this loop, summary of issues
Instruction group 1
Instruction group 2
Instruction group 3
Explaining what to do
Klar
Start
18:47

Thank You

Ed Hinkel
Rogue Wave Software
ehinkel@roguewave.com