# Debugging CUDA Accelerated MPI Codes

Chris Gottbrath
Principal Product Manager, Rogue Wave Software

Aug 16th, 2011

# Agenda

- Rogue Wave Software
  - TotalView
  - MemoryScape
  - ReplayEngine
  - ThreadSpotter
- CUDA Debugging
  - Intro and Demo
- Memory Debugging
- Automated Debugging
- Technology Update
  - New Features and Capabilities
  - Scalability
- Conclusion

# Rogue Wave Today

ROGUE WAVE® SOFTWARE

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.

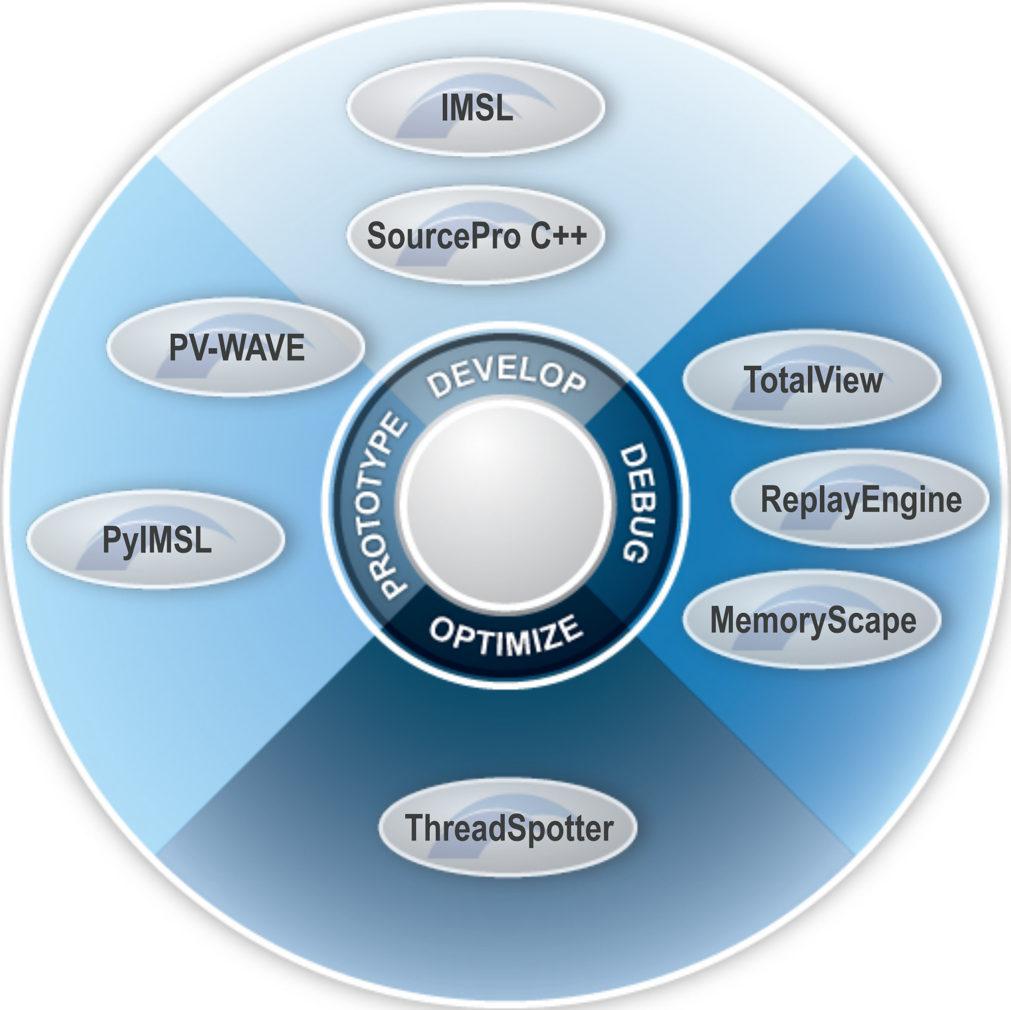
acumem multicore performance

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

TOTALVIEW

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

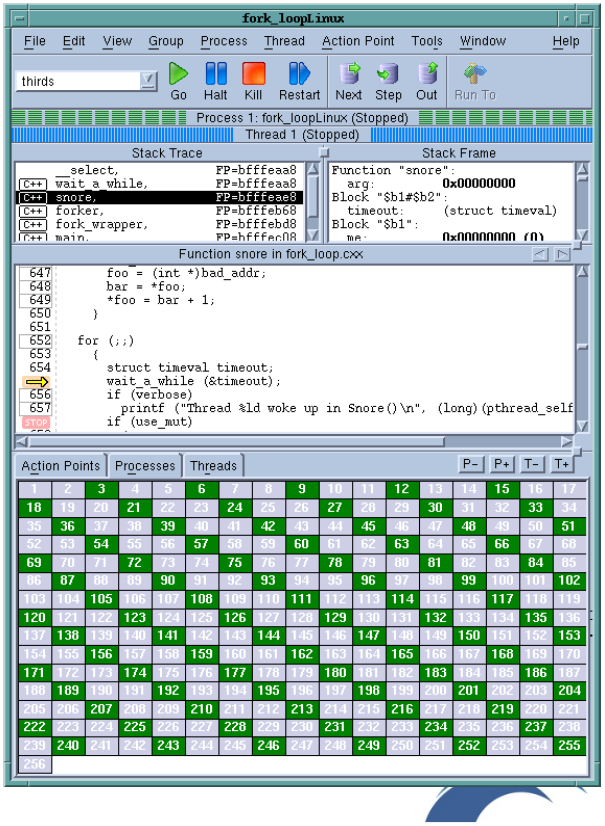
# Rogue Wave Product Offerings



# What is TotalView?

- **Application Analysis and Debugging Tool: Code Confidently**
  - Debug and Analyze C/C++ and Fortran on Linux, Unix or Mac OS X
  - Laptops to supercomputers (BG, Cray)
  - 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, GA, UPC
    - CUDA Support available
  - Includes a Remote Display Client freeing users to work from anywhere
  - Includes Memory Debugging with MemoryScape
  - Reverse Debugging available with ReplayEngine
  - Includes Batch Debugging with TVScript and the CLI



Process 1: fork_loopLinux (Stopped)

Thread 1 (Stopped)

```
      __select,        FP=bfffeaa8
C++   wait_a_while,    FP=bfffeaa8
C++   snore,           FP=bfffeae8
C++   forker,          FP=bfffeb68
C++   fork_wrapper,    FP=bfffebd8
C++   main,            FP=bfffec08
```

```
Function "snore":
  arg:            0x00000000
Block "$b1#$b2":
  timeout:        (struct timeval)
Block "$b1":
  me:             0x00000000 (0)
```

Function snore in fork_loop.cxx

```
      foo = (int *)bad_addr;
      bar = *foo;
      *foo = bar + 1;
    }

  for (;;)
    {
      struct timeval timeout;
      wait_a_while (&timeout);
      if (verbose)
        printf ("Thread %ld woke up in Snore()\n", (long)(pthread_self
      if (use_mut)
```

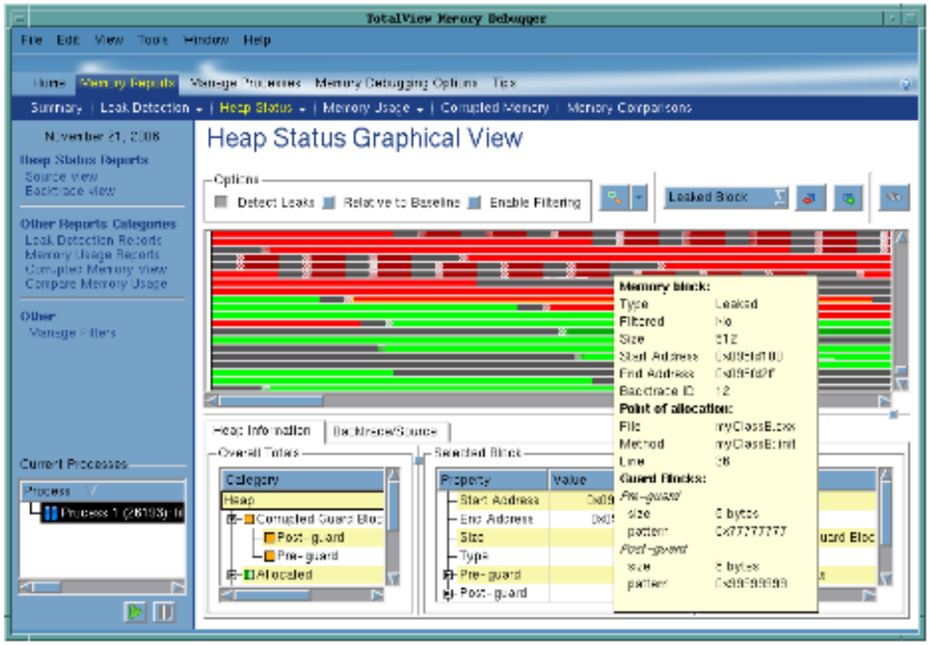
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | **3** | 4 | 5 | **6** | 7 | 8 | **9** | 10 | 11 | **12** | 13 | 14 | **15** | 16 | 17 |
| **18** | 19 | 20 | **21** | 22 | 23 | **24** | 25 | 26 | **27** | 28 | 29 | **30** | 31 | 32 | **33** | 34 |
| 35 | **36** | 37 | 38 | **39** | 40 | 41 | **42** | 43 | 44 | **45** | 46 | 47 | **48** | 49 | 50 | **51** |
| 52 | 53 | **54** | 55 | 56 | **57** | 58 | 59 | **60** | 61 | 62 | **63** | 64 | 65 | **66** | 67 | 68 |
| **69** | 70 | 71 | **72** | 73 | 74 | **75** | 76 | 77 | **78** | 79 | 80 | **81** | 82 | 83 | **84** | 85 |
| 86 | **87** | 88 | 89 | **90** | 91 | 92 | **93** | 94 | 95 | **96** | 97 | 98 | **99** | 100 | 101 | **102** |
| 103 | 104 | **105** | 106 | 107 | **108** | 109 | 110 | **111** | 112 | 113 | **114** | 115 | 116 | **117** | 118 | 119 |
| **120** | 121 | 122 | **123** | 124 | 125 | **126** | 127 | 128 | **129** | 130 | 131 | **132** | 133 | 134 | **135** | 136 |
| 137 | **138** | 139 | 140 | **141** | 142 | 143 | **144** | 145 | 146 | **147** | 148 | 149 | **150** | 151 | 152 | **153** |
| 154 | 155 | **156** | 157 | 158 | **159** | 160 | 161 | **162** | 163 | 164 | **165** | 166 | 167 | **168** | 169 | 170 |
| **171** | 172 | 173 | **174** | 175 | 176 | **177** | 178 | 179 | **180** | 181 | 182 | **183** | 184 | 185 | **186** | 187 |
| 188 | **189** | 190 | 191 | **192** | 193 | 194 | **195** | 196 | 197 | **198** | 199 | 200 | **201** | 202 | 203 | **204** |
| 205 | 206 | **207** | 208 | 209 | **210** | 211 | 212 | **213** | 214 | 215 | **216** | 217 | 218 | **219** | 220 | 221 |
| **222** | 223 | 224 | **225** | 226 | 227 | **228** | 229 | 230 | **231** | 232 | 233 | **234** | 235 | 236 | **237** | 238 |
| 239 | **240** | 241 | 242 | **243** | 244 | 245 | **246** | 247 | 248 | **249** | 250 | 251 | **252** | 253 | 254 | **255** |
| 256 | | | | | | | | | | | | | | | | |

ROGUE WAVE SOFTWARE

# What Is MemoryScape?

- **Runtime Memory Analysis : Eliminate Memory Errors**
  - Detects memory leaks *before* they are a problem
  - Explore heap memory usage with powerful analytical tools
  - Use for validation as part of a quality software development process

- **Major Features**
  - Detects
    - Malloc API misuse
    - Memory leaks
    - Buffer overflows
  - Supports
    - C, C++, Fortran
    - Linux, Unix, and Mac OS X
    - MPI, pthreads, OMP, and remote apps
  - Low runtime overhead
  - Easy to use
    - Works with vendor libraries
    - No recompilation or instrumentation
  - Enables Collaboration



**Memory block:**

| | |
|---|---|
| Type | Leaked |
| Filtered | No |
| Size | 512 |
| Start Address | 0x09e1d100 |
| End Address | 0x09f0d2f |
| Backtrace ID | 12 |

**Point of allocation:**

| | |
|---|---|
| File | myClassB.cxx |
| Method | myClassB::init |
| Line | 36 |

**Guard Blocks:**

*Pre-guard*

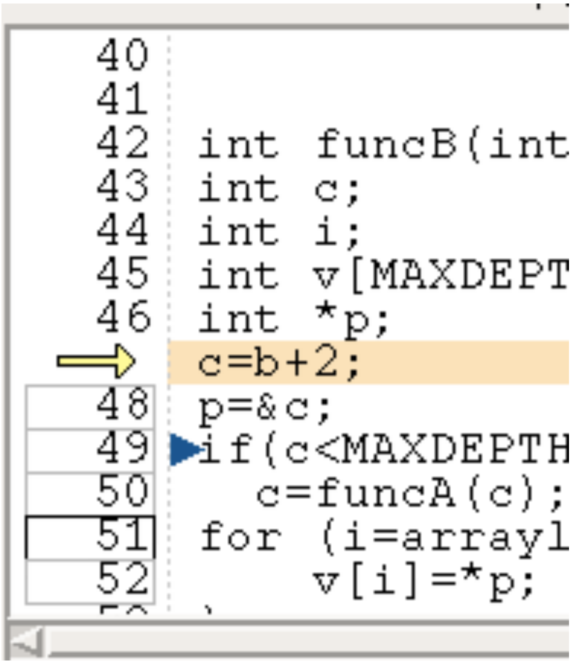
| | |
|---|---|
| size | 0 bytes |
| pattern | 0x77777777 |

*Post-guard*

| | |
|---|---|
| size | 0 bytes |
| pattern | 0x99999999 |

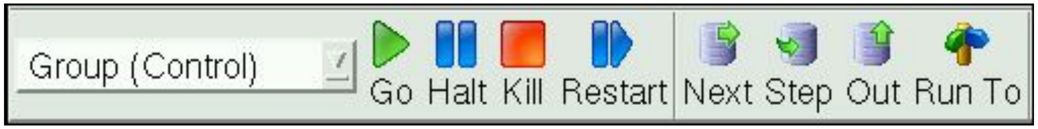
# What Is ReplayEngine?

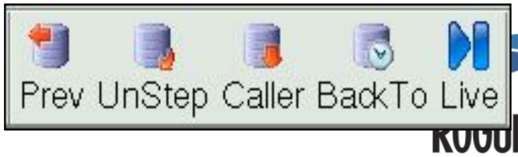
- **Reverse Debugging Tool: Radically simplify your debugging**
  - Captures and Deterministically Replays Execution
  - Eliminate the Restart Cycle and Hard-to-Reproduce Bugs
  - Step Back and Forward by Function, Line, or Instruction



```
int funcB(int
int c;
int i;
int v[MAXDEPT
int *p;
c=b+2;
p=&c;
if(c<MAXDEPTH
  c=funcA(c);
for (i=arrayl
    v[i]=*p;
```

- **Major Features**
  - Simple extension to TotalView
    - No recompilation or instrumentation
    - Explore data and state in the past just like in a live process
  - Supported on Linux x86 and x86-64
  - Supports MPI, Pthreads, and OpenMP

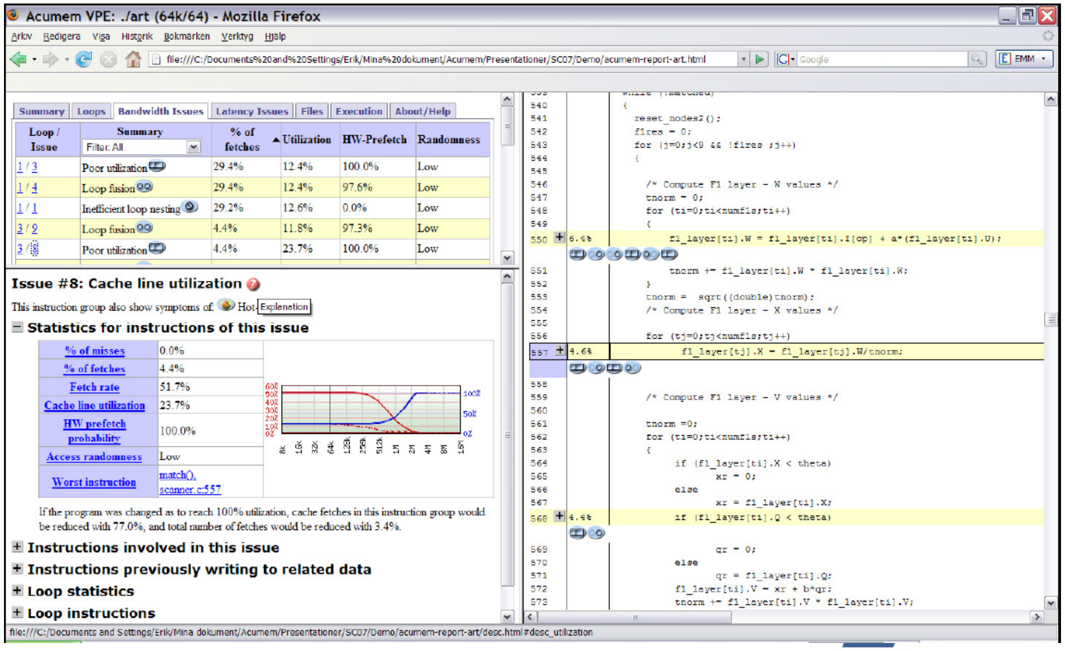




ROGUE WAVE SOFTWARE

# What is ThreadSpotter?

- Runtime Cache Performance Optimization Tool: Tune into the Multi-Core Era
  - Realize More of the Performance Offered by Multi/Many-Core Chips
  - Quickly Detects and Prioritizes Issues -- and then Provides Usable Advice!
    - Brings Cache Performance Into Reach for Every Developer
    - Makes Experienced Cache Optimizers Hyper-Efficient
- Features
  - Supports Linux x86/x86-64
  - Any compiled code
  - Runtime Analysis
    - Low overhead
  - Cache Modeling
    - Prioritizes Issues
    - Identifies Problem Lines of Code
  - Provides Advice
    - Explanations
    - Examples
    - Detailed statistics (if desired)


Acumem VPE: ./art (64k/64) - Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
file:///C:/Documents%20and%20Settings/Erik/Mina%20dokument/Acumem/Presentationer/SC07/Demo/acumem-report-art.html
Google
EMM
Summary
Loops
Bandwidth Issues
Latency Issues
Files
Execution
About/Help
60%
50%
40%
30%
20%
10%
0%
100%
50%
0%
8k
16k
32k
64k
128k
256k
512k
1M
2M
4M
8M
16M

| Loop / Issue | Summary<br>Filter: All | % of fetches | ▲Utilization | HW-Prefetch | Randomness |
|---|---|---|---|---|---|
| 1 / 3 | Poor utilization | 29.4% | 12.4% | 100.0% | Low |
| 1 / 4 | Loop fusion | 29.4% | 12.4% | 97.6% | Low |
| 1 / 1 | Inefficient loop nesting | 29.2% | 12.6% | 0.0% | Low |
| 3 / 9 | Loop fusion | 4.4% | 11.8% | 97.3% | Low |
| 3 / 8 | Poor utilization | 4.4% | 23.7% | 100.0% | Low |

**Issue #8: Cache line utilization**

This instruction group also show symptoms of: Hot- Explanation

**− Statistics for instructions of this issue**

| | |
|---|---|
| % of misses | 0.0% |
| % of fetches | 4.4% |
| Fetch rate | 51.7% |
| Cache line utilization | 23.7% |
| HW prefetch probability | 100.0% |
| Access randomness | Low |
| Worst instruction | match(),<br>scanner.c:557 |

If the program was changed as to reach 100% utilization, cache fetches in this instruction group would be reduced with 77.0%, and total number of fetches would be reduced with 3.4%.

**+ Instructions involved in this issue**

**+ Instructions previously writing to related data**

**+ Loop statistics**

**+ Loop instructions**

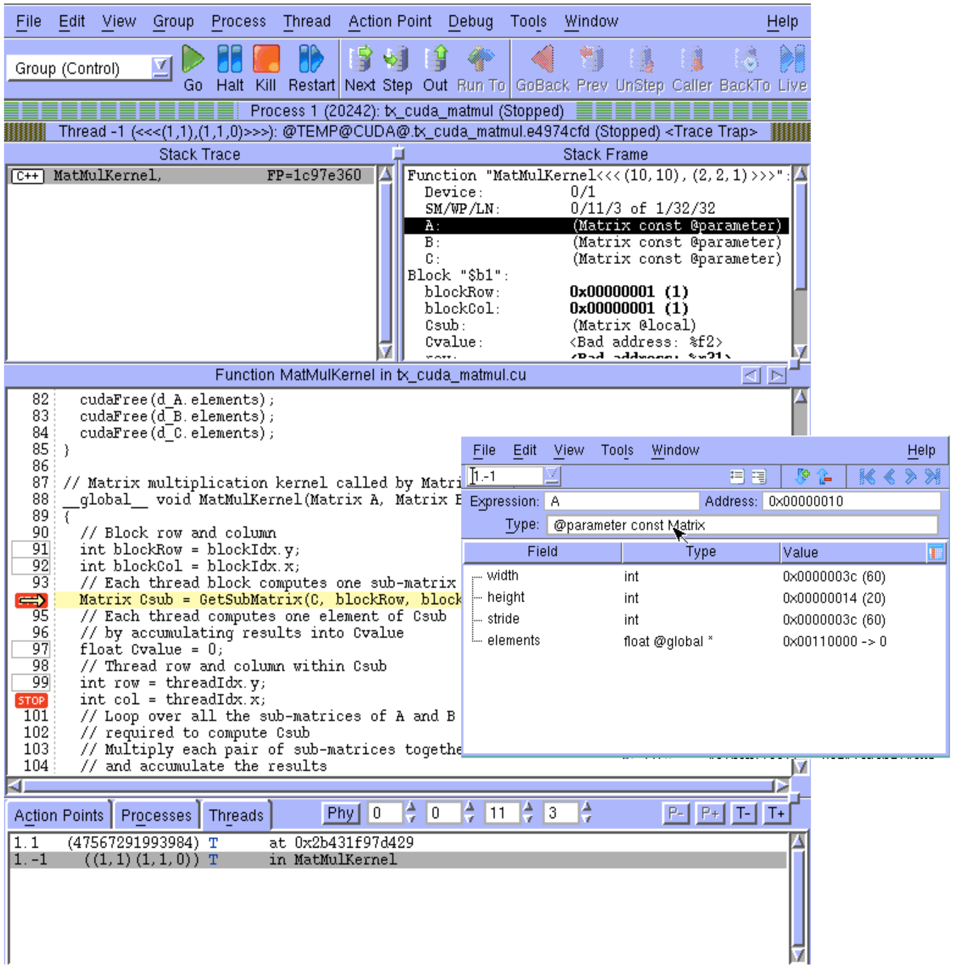
```
540           {
541             reset_nodes2();
542             f1res = 0;
543             for (j=0;j<9 && !f1res ;j++)
544             {
545
546               /* Compute F1 layer - W values */
547               tnorm = 0;
548               for (ti=0;ti<numf1s;ti++)
549               {
550 + 6.4%            f1_layer[ti].W = f1_layer[ti].I[cp] + a*(f1_layer[ti].U);
551                   tnorm += f1_layer[ti].W * f1_layer[ti].W;
552               }
553               tnorm =  sqrt((double)tnorm);
554               /* Compute F1 layer - X values */
555
556               for (tj=0;tj<numf1s;tj++)
557 + 4.6%              f1_layer[tj].X = f1_layer[tj].W/tnorm;
558
559               /* Compute F1 layer - V values */
560
561               tnorm =0;
562               for (ti=0;ti<numf1s;ti++)
563               {
564                   if (f1_layer[ti].X < theta)
565                           xr = 0;
566                   else
567                           xr = f1_layer[ti].X;
568 + 4.4%            if (f1_layer[ti].Q < theta)
569                           qr = 0;
570                   else
571                           qr = f1_layer[ti].Q;
572                   f1_layer[ti].V = xr + b*qr;
573                   tnorm += f1_layer[ti].V * f1_layer[ti].V;
```

file:///C:/Documents and Settings/Erik/Mina dokument/Acumem/Presentationer/SC07/Demo/acumem-report-art/desc.html#desc_utilization

# Programming for the GP-GPU

- CUDA
  - Function-like kernels are written for the calculations to be performed on the GPU
    - Data parallel style, one kernel per unit of work
  - Presents a hierarchical organization for thread contexts
    - 2D grid of blocks
    - 3D block of thread
  - Exposes memory hierarchy explicitly to the user
  - Includes routines for managing device memory and data movement to and from device memory using streams
- Programming challenges
  - Coordinating CPU code + device code
  - Understanding what is going on in each kernel
    - Exceptions
  - Understanding memory usage
  - Understanding performance characteristics

# TotalView for CUDA



Process 1 (20242): tx_cuda_matmul (Stopped)

Thread -1 (<<<(1,1),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.e4974cfd (Stopped) <Trace Trap>

```
C++  MatMulKernel,                FP=1c97e360
```

```
Function "MatMulKernel<<<(10,10),(2,2,1)>>>":
  Device:          0/1
  SM/WP/LN:        0/11/3 of 1/32/32
  A:               (Matrix const @parameter)
  B:               (Matrix const @parameter)
  C:               (Matrix const @parameter)
Block "$b1":
  blockRow:        0x00000001 (1)
  blockCol:        0x00000001 (1)
  Csub:            (Matrix @local)
  Cvalue:          <Bad address: %f2>
```

Function MatMulKernel in tx_cuda_matmul.cu

```
  cudaFree(d_A.elements);
  cudaFree(d_B.elements);
  cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by Matri
__global__ void MatMulKernel(Matrix A, Matrix B
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matrix
  Matrix Csub = GetSubMatrix(C, blockRow, block
  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
  // Loop over all the sub-matrices of A and B
  // required to compute Csub
  // Multiply each pair of sub-matrices togethe
  // and accumulate the results
```

1.-1

Expression: A    Address: 0x00000010

Type: @parameter const Matrix

| Field | Type | Value |
|---|---|---|
| width | int | 0x0000003c (60) |
| height | int | 0x00000014 (20) |
| stride | int | 0x0000003c (60) |
| elements | float @global * | 0x00110000 -> 0 |

Phy 0 0 11 3

```
1.1   (47567291993984) T      at 0x2b431f97d429
1.-1    ((1,1)(1,1,0)) T      in MatMulKernel
```

- Characteristics
  - Debugging of application running on the GPU device (not in an emulator)
  - Full visibility of both Linux threads and GPU device threads
  - Fully represent the hierarchical memory
  - Thread and Block Coordinates
  - Device thread control
  - Handles CUDA function inlining
  - Reports memory access errors
  - Multi-Device Support
  - Can be used with MPI
- Supports CUDA 4.0 (in beta)

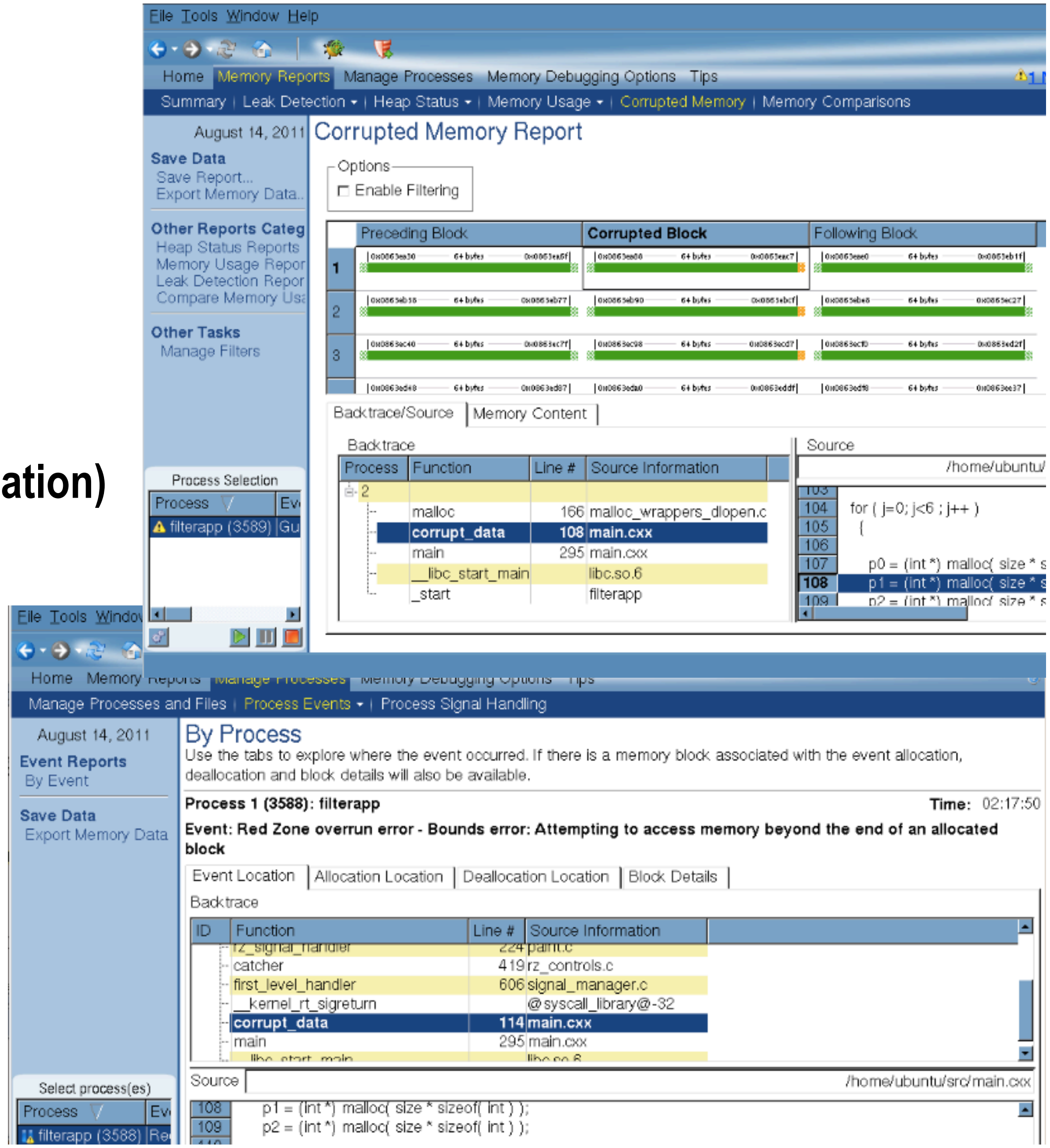
# Memory Debugging

- Heap Memory
  - User is responsible for managing
  - C: Malloc / Free
  - C++: New / Delete
  - F90: Allocate / Deallocate

- Buffer Overrun / Array Bounds Violations
- Memory Leaks
- Memory Optimization

# Heap Array Bounds Violations

- Writing Outside of Allocation
  - Can result in random errors
  - Dangling pointer
  - Array index error (off by one)
- Guard Blocks
  - Lightweight (few byes per allocation)
  - Fast
  - Notification on demand
  - Notification after free
- RedZones
  - Heavier (page per allocation)
  - Fast
  - Notification at point of error



August 14, 2011

Corrupted Memory Report

Options

□ Enable Filtering

| | Preceding Block | Corrupted Block | Following Block |
|---|---|---|---|
| 1 | 0x0863ea30 — 64 bytes — 0x0863ea6f | 0x0863ea88 — 64 bytes — 0x0863eac7 | 0x0863eae0 — 64 bytes — 0x0863eb1f |
| 2 | 0x0863eb38 — 64 bytes — 0x0863eb77 | 0x0863eb90 — 64 bytes — 0x0863ebcf | 0x0863ebe8 — 64 bytes — 0x0863ec27 |
| 3 | 0x0863ec40 — 64 bytes — 0x0863ec7f | 0x0863ec98 — 64 bytes — 0x0863ecd7 | 0x0863ecf0 — 64 bytes — 0x0863ed2f |
| | 0x0863ed48 — 64 bytes — 0x0863ed87 | 0x0863eda0 — 64 bytes — 0x0863eddf | 0x0863edf8 — 64 bytes — 0x0863ee37 |

Backtrace/Source | Memory Content

Backtrace

| Process | Function | Line # | Source Information |
|---|---|---|---|
| 2 | | | |
| | malloc | 166 | malloc_wrappers_dlopen.c |
| | **corrupt_data** | **108** | **main.cxx** |
| | main | 295 | main.cxx |
| | __libc_start_main | | libc.so.6 |
| | _start | | filterapp |

Source /home/ubuntu/

```
103
104   for ( j=0; j<6 ; j++ )
105   {
106
107       p0 = (int *) malloc( size * s
108       p1 = (int *) malloc( size * s
109       p2 = (int *) malloc( size * s
```

August 14, 2011

By Process

Use the tabs to explore where the event occurred. If there is a memory block associated with the event allocation, deallocation and block details will also be available.

**Process 1 (3588): filterapp** **Time:** 02:17:50

**Event: Red Zone overrun error - Bounds error: Attempting to access memory beyond the end of an allocated block**

Event Location | Allocation Location | Deallocation Location | Block Details

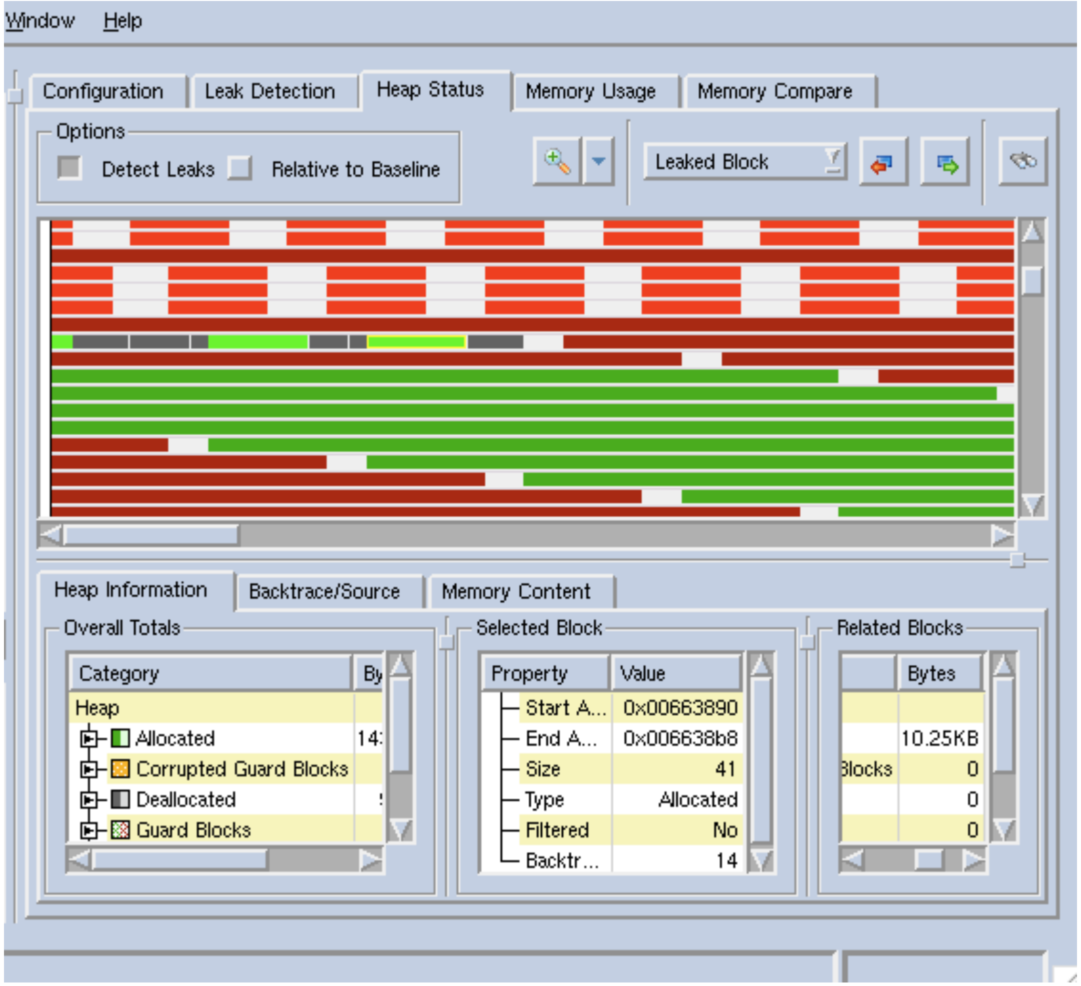
Backtrace

| ID | Function | Line # | Source Information |
|---|---|---|---|
| | rz_signal_handler | 224 | paint.c |
| | catcher | 419 | rz_controls.c |
| | first_level_handler | 606 | signal_manager.c |
| | __kernel_rt_sigreturn | | @syscall_library@-32 |
| | **corrupt_data** | **114** | **main.cxx** |
| | main | 295 | main.cxx |
| | __libc_start_main | | libc.so.6 |

Source /home/ubuntu/src/main.cxx

```
108   p1 = (int *) malloc( size * sizeof( int ) );
109   p2 = (int *) malloc( size * sizeof( int ) );
```

# Leak Detection



Options

□ Detect Leaks □ Relative to Baseline

Overall Totals

| Category | By |
|---|---|
| Heap | |
| Allocated | 14: |
| Corrupted Guard Blocks | |
| Deallocated | ! |
| Guard Blocks | |

Selected Block

| Property | Value |
|---|---|
| Start A... | 0x00663890 |
| End A... | 0x006638b8 |
| Size | 41 |
| Type | Allocated |
| Filtered | No |
| Backtr... | 14 |

Related Blocks

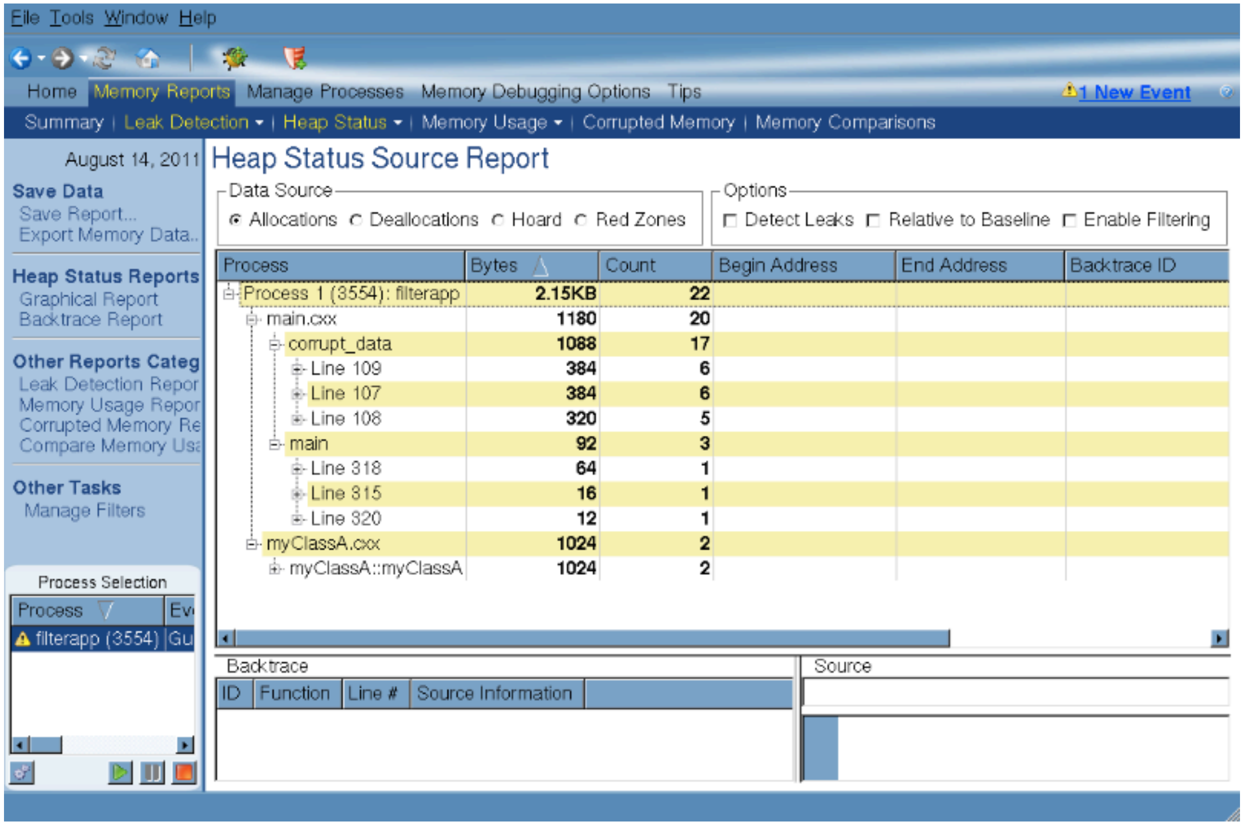
| | Bytes |
|---|---|
| | |
| | 10.25KB |
| Blocks | 0 |
| | 0 |
| | 0 |

## 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

# Memory Optimization

- Prevent OOM errors
- Mem Usage
  - Per process
  - Per library
  - Per function
- Compare
  - Between
    - Processes
    - Points in Time
    - Datasets
    - Runs
- Track
  - Automate reporting



August 14, 2011

Heap Status Source Report

Data Source: ● Allocations ○ Deallocations ○ Hoard ○ Red Zones

Options: □ Detect Leaks □ Relative to Baseline □ Enable Filtering

| Process | Bytes | Count | Begin Address | End Address | Backtrace ID |
|---|---|---|---|---|---|
| Process 1 (3554): filterapp | 2.15KB | 22 | | | |
| main.cxx | 1180 | 20 | | | |
| corrupt_data | 1088 | 17 | | | |
| Line 109 | 384 | 6 | | | |
| Line 107 | 384 | 6 | | | |
| Line 108 | 320 | 5 | | | |
| main | 92 | 3 | | | |
| Line 318 | 64 | 1 | | | |
| Line 315 | 16 | 1 | | | |
| Line 320 | 12 | 1 | | | |
| myClassA.cxx | 1024 | 2 | | | |
| myClassA::myClassA | 1024 | 2 | | | |

# Automatic Debugging

- Non-Interactive Batch Debugging
  - Work in the “main” batch queue
  - Don’ t have to baby-sit job waiting on it to run
  - Can script to perform checks that would be tedious to do by hand
  - Verification can be part of automated processes (nightly build and test)

- Automatic Transformation of Data
  - Simplify interactive (and scripted) debugging
  - Perform validation/sanity checking of large datasets
  - Comparative debugging
  - Allows you to focus on troubleshooting your program

# 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 [-level ***level-num***] [***num_levels***] [***options***]
    - List_leaks
    - Save_memory
    - Print [-slice {***slice_exp***] {***variable*** | ***exp***}

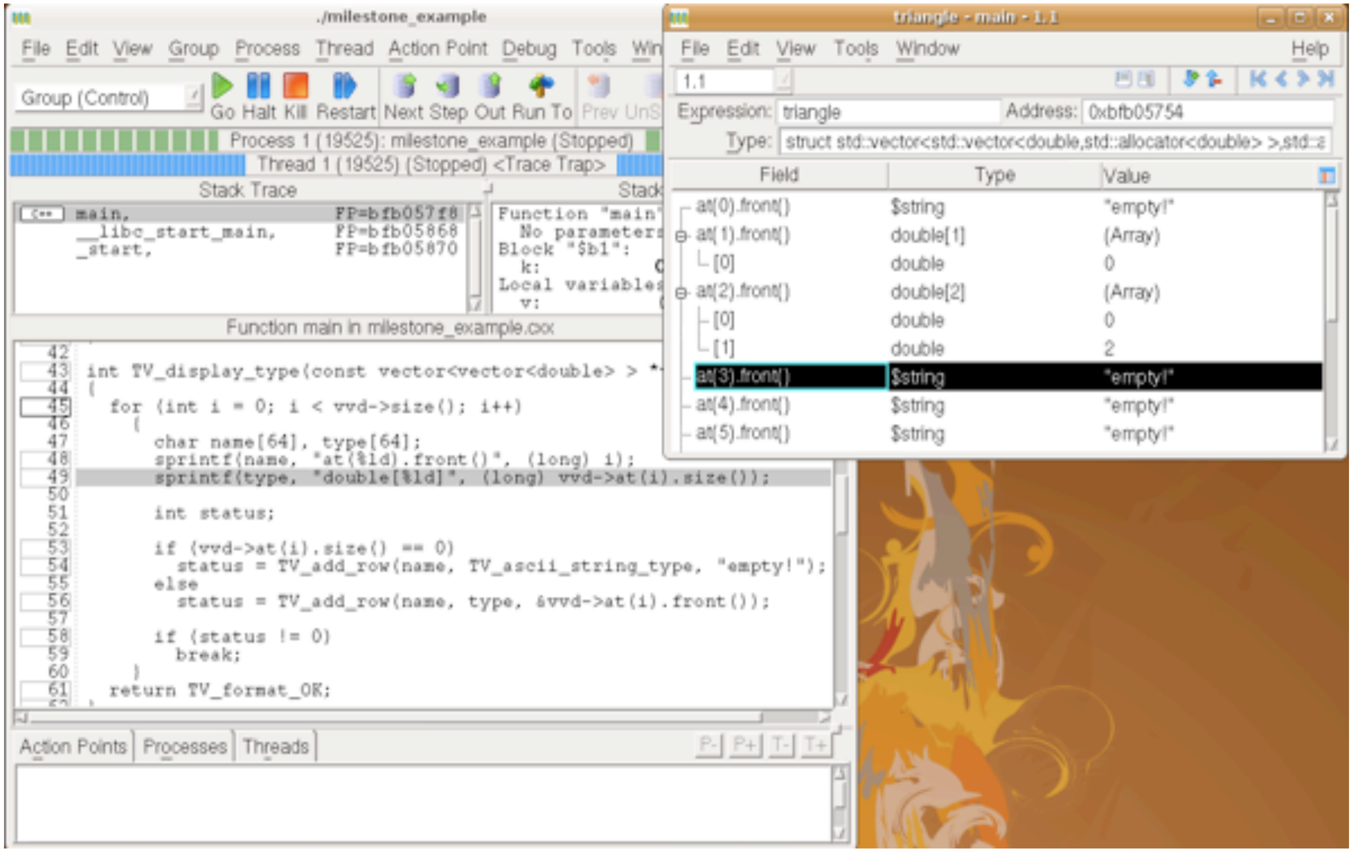
```
•!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!
•! Print
•!
•! Process:
•!   ./server (Debugger Process ID:  1, System ID:  12110)
•! Thread:
•!   Debugger ID:  1.1, System ID:  3083946656
•! Time Stamp:
•!   06-26-2008 14:04:09
•! Triggered from event:
•!   actionpoint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

- **Example**

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

# 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


./milestone_example
File Edit View Group Process Thread Action Point Debug Tools Win
Group (Control)
Go Halt Kill Restart Next Step Out Run To Prev UnS
Stack Trace
Stack
Function main in milestone_example.cxx
Action Points Processes Threads
P- P+ T- T+
triangle - main - 1.1
File Edit View Tools Window Help
1.1

Process 1 (19525): milestone_example (Stopped)

Thread 1 (19525) (Stopped) <Trace Trap>

```
C++ main,               FP=bfb057f8
    __libc_start_main,  FP=bfb05868
    _start,             FP=bfb05870
```

```
Function "main"
  No parameters
Block "$b1":
  k:
Local variables
  v:
```

```
int TV_display_type(const vector<vector<double> > *
{
  for (int i = 0; i < vvd->size(); i++)
    {
      char name[64], type[64];
      sprintf(name, "at(%ld).front()", (long) i);
      sprintf(type, "double[%ld]", (long) vvd->at(i).size());

      int status;

      if (vvd->at(i).size() == 0)
        status = TV_add_row(name, TV_ascii_string_type, "empty!");
      else
        status = TV_add_row(name, type, &vvd->at(i).front());

      if (status != 0)
        break;
    }
  return TV_format_OK;
```

Expression: triangle Address: 0xbfb05754

Type: struct std::vector<std::vector<double,std::allocator<double> >,std::a

| Field | Type | Value |
|---|---|---|
| at(0).front() | $string | "empty!" |
| at(1).front() | double[1] | (Array) |
| [0] | double | 0 |
| at(2).front() | double[2] | (Array) |
| [0] | double | 0 |
| [1] | double | 2 |
| at(3).front() | $string | "empty!" |
| at(4).front() | $string | "empty!" |
| at(5).front() | $string | "empty!" |

# C++View Interface

- **Only two functions:**

```
int TV_ttf_display_type  ( const T * )

int TV_ttf_add_row (
   const char * field_name,
   const char * field_type,
   const char * address)
```

# Scalability In TotalView Today

- A Long History of Leadership
  - Have worked with customers such as LLNL, LANL, Sandia and others on scalability improvements for many years
- TotalView Architecture
  - No Hard Limit
  - Multi-Platform (Cray, IBM BG, Linux Clusters, etc..)
  - Efficient Use of Cluster Resources
    - Extremely light weight debug agents
    - Minimal memory footprint (efficient shared data structures)
    - Each agent can control many processes and threads
  - Challenging User Applications
    - More space on the compute nodes for user application code
  - Full Control of Debugger Components
    - Changes focused on HPC needs
- Customer Experiences
  - TotalView is regularly used to debug scales of up to 10k processes
  - TotalView is also used on >10k processes

# Research and Development

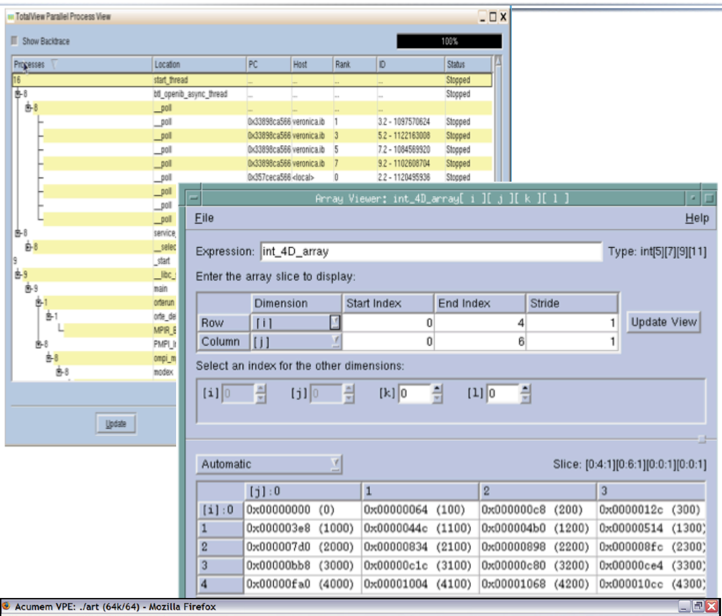
- Current Focus Areas
  - Transition TotalView from a flat 1:N communication to a tree
  - Scalable presentation of state and data
  - Usability at scale
  - Application driven tuning: Optimization focused on real-world applications and workloads
    - Across various machines
- Goals
  - Provide performance at >100,000 tasks to be debugged
  - Setting the stage for the millions of tasks we expect to see at exascale
- Several Concurrent Projects
  - FastOS project with Bart Miller and Mike Brim of University of Wisconsin
    - TBON-FS Group File Operations
    - Academic research based on MRNet & Dyninst components
  - LLNL Petascale Parallel Debugger Scalability contract
    - MRNet - product R & D
    - Multi-platform: BlueGene/Q, Cray XT/XE/XK, Linux Cluster
    - Preliminary results
      - First user observable improvements are in start up time
      - 5x improvement in at-scale start up performance on Cray
      - 20x improvement in at-scale start up performance on a “vanilla” linux cluster.
  - LLNL IDDA  Dynamic Application contract
    - Focusing on a class of tool-breaking applications
    - Thousands of DLLs and Huge Symbol Table Size

# Peta and Exascale Scalability

- R&D work is planned to roll into the product releases 2012 and 2013
  - Multi-platform Application Based Optimization
    - Cray XT/XE/XK, Blue Gene/Q, Linux clusters
    - Scientific applications including especially dynamic apps
    - GPU accelerated cluster scalability
  - Tree-Based Overlay Network
    - Broadcast of Operations
    - Aggregation of Events and Data
  - UI Layer
    - New GUI Framework
    - Co-Design of Advanced Displays for Debugging at Scale
    - Simplifed Discovery of Relevant Information Through Aggregation
- These changes set the stage for exascale debugging
  - Multi-platform
  - Highly real-world optimized
  - Tree based
  - Low resource usage
  - Support for computational accelerator technology
  - Highly flexible architecture with an exclusive focus on HPC

# Recent Changes

- **TV 8.9 series**
  - Powerful parallel debugging
  - Support for CUDA 3.0 - 4.0 (in beta)
  - New Views: Multi-dimensional Array & Parallel Backtrace
  - C++View and TVScript for Automatic Debugging
  - Easy and Secure Remote Graphical Display
  - Updated platform support
- **ReplayEngine 2.0 series**
  - Deterministic Replay Radically Transforms Debugging
  - Brings Reverse Debugging to HPC Clusters
- **MemoryScape 3.2 series**
  - Memory Leaks and Array Bounds Checking for HPC
  - Red Zones for Instant Array Bounds Checking
- **ThreadSpotter 2011**
  - Memory Cache Optimization Made Easy



| Processes | Location | PC | Host | Rank | ID | Status |
|---|---|---|---|---|---|---|
| 16 | start_thread | ... | ... | ... | ... | Stopped |
| 8 | btl_openib_async_thread | ... | ... | ... | ... | Stopped |
| 8 | __poll | ... | ... | ... | ... | |
| | __poll | 0x33898ca566 | veronica.ib | 1 | 3.2 - 1097570624 | Stopped |
| | __poll | 0x33898ca566 | veronica.ib | 3 | 5.2 - 1122163008 | Stopped |
| | __poll | 0x33898ca566 | veronica.ib | 5 | 7.2 - 1084569920 | Stopped |
| | __poll | 0x33898ca566 | veronica.ib | 7 | 9.2 - 1102608704 | Stopped |
| | __poll | 0x357ceca566 | <local> | 0 | 2.2 - 1120495936 | Stopped |
| | __poll | | | | | |
| | __poll | | | | | |
| | __poll | | | | | |
| 8 | service_ | | | | | |
| 8 | __selec | | | | | |
| 9 | _start | | | | | |
| 9 | __libc_ | | | | | |
| 9 | main | | | | | |
| 1 | orterun | | | | | |
| 1 | orte_de | | | | | |
| | MPIR_B | | | | | |
| 8 | PMPI_I | | | | | |
| 8 | ompi_m | | | | | |
| 8 | modex | | | | | |

Expression: int_4D_array

Type: int[5][7][9][11]

Enter the array slice to display:

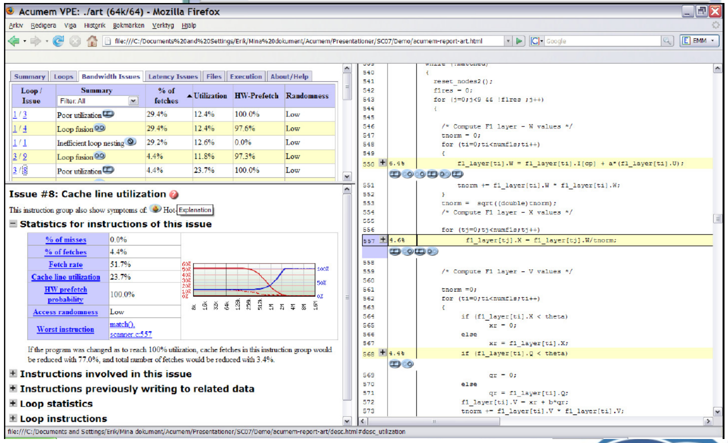
| | Dimension | Start Index | End Index | Stride |
|---|---|---|---|---|
| Row | [ i ] | 0 | 4 | 1 |
| Column | [ j ] | 0 | 6 | 1 |

Select an index for the other dimensions:

[i] 0 [j] 0 [k] 0 [l] 0

Slice: [0:4:1][0:6:1][0:0:1][0:0:1]

| | [j] : 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| [i] : 0 | 0x00000000 (0) | 0x00000064 (100) | 0x000000c8 (200) | 0x0000012c (300) |
| 1 | 0x000003e8 (1000) | 0x0000044c (1100) | 0x000004b0 (1200) | 0x00000514 (1300) |
| 2 | 0x000007d0 (2000) | 0x00000834 (2100) | 0x00000898 (2200) | 0x000008fc (2300) |
| 3 | 0x00000bb8 (3000) | 0x00000c1c (3100) | 0x00000c80 (3200) | 0x00000ce4 (3300) |
| 4 | 0x00000fa0 (4000) | 0x00001004 (4100) | 0x00001068 (4200) | 0x000010cc (4300) |



Acumem VPE: ./art (64k/64) - Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
file:///C:/Documents%20and%20Settings/Erik/Mina%20dokument/Acumem/Presentationer/SC07/Demo/acumem-report-art.html
Google
EMM
Summary Loops Bandwidth Issues Latency Issues Files Execution About/Help
60% 50% 40% 30% 20% 10% 0%
100% 50% 0%
8k 16k 32k 64k 128k 256k 512k 1M 2M 4M 8M 16M

| Loop / Issue | Summary<br>Filter: All | % of fetches | Utilization | HW-Prefetch | Randomness |
|---|---|---|---|---|---|
| 1 / 3 | Poor utilization | 29.4% | 12.4% | 100.0% | Low |
| 1 / 4 | Loop fusion | 29.4% | 12.4% | 97.6% | Low |
| 1 / 1 | Inefficient loop nesting | 29.2% | 12.6% | 0.0% | Low |
| 3 / 9 | Loop fusion | 4.4% | 11.8% | 97.3% | Low |
| 3 / 8 | Poor utilization | 4.4% | 23.7% | 100.0% | Low |

**Issue #8: Cache line utilization**

This instruction group also show symptoms of: Hot- Explanation

⊟ **Statistics for instructions of this issue**

| | |
|---|---|
| % of misses | 0.0% |
| % of fetches | 4.4% |
| Fetch rate | 51.7% |
| Cache line utilization | 23.7% |
| HW prefetch probability | 100.0% |
| Access randomness | Low |
| Worst instruction | match(), scanner.c:557 |

If the program was changed as to reach 100% utilization, cache fetches in this instruction group would be reduced with 77.0%, and total number of fetches would be reduced with 3.4%.

⊞ **Instructions involved in this issue**

⊞ **Instructions previously writing to related data**

⊞ **Loop statistics**

⊞ **Loop instructions**

```
      {
        reset_nodes2();
        f1res = 0;
        for (j=0;j<9 && !f1res ;j++)
        {

          /* Compute F1 layer - W values */
          tnorm = 0;
          for (ti=0;ti<numf1s;ti++)
          {
6.4%          f1_layer[ti].W = f1_layer[ti].I[cp] + a*(f1_layer[ti].U);
              tnorm += f1_layer[ti].W * f1_layer[ti].W;
          }
          tnorm =  sqrt((double)tnorm);
          /* Compute F1 layer - X values */

          for (tj=0;tj<numf1s;tj++)
4.6%            f1_layer[tj].X = f1_layer[tj].W/tnorm;


          /* Compute F1 layer - V values */

          tnorm =0;
          for (ti=0;ti<numf1s;ti++)
          {
               if (f1_layer[ti].X < theta)
                      xr = 0;
               else
                      xr = f1_layer[ti].X;
4.4%           if (f1_layer[ti].Q < theta)
                      qr = 0;
               else
                      qr = f1_layer[ti].Q;
               f1_layer[ti].V = xr + b*qr;
               tnorm += f1_layer[ti].V * f1_layer[ti].V;
```

file:///C:/Documents and Settings/Erik/Mina dokument/Acumem/Presentationer/SC07/Demo/acumem-report-art/desc.html#desc_utilization

# Summary

- Rogue Wave

  HPC tools, components and libraries

  Parallel Programming is Hard, We Make it Easier

- Debugging with the TotalView Family of Products
  - Advanced, Scalable, Graphical, Easy to Use
  - MPI Debugging
  - CUDA Debugging
  - Memory Debugging
  - Automated Debugging
  - Deterministic Reverse Debugging

- Optimization with ThreadSpotter
  - Programmer Friendly Analysis of Cache and Memory Use

# Thanks!

- Contact me
  - Chris.Gottbrath@RogueWave.com
- or my colleagues
  - Ian.Dillan@RogueWave.com
  - Ed.Hinkel@RogueWave.com
- or for more information

  Check out: www.roguewave.com

  Email: TVSupport@roguewave.com