Getting Started with TotalView and CUDA

Chris Gottbrath, Principal Product Manager
Ed Hinkel, Senior Sales Engineer

Agenda

- Rogue Wave Software
- CUDA Challenges
- TotalView Debugger
- TotalView for CUDA
- Follow up

ROGUE WAVE SOFTWARE

Rogue Wave Today

- **Pioneers in C++/object-oriented development**
- **Leading the way in cross-platform, parallel development**

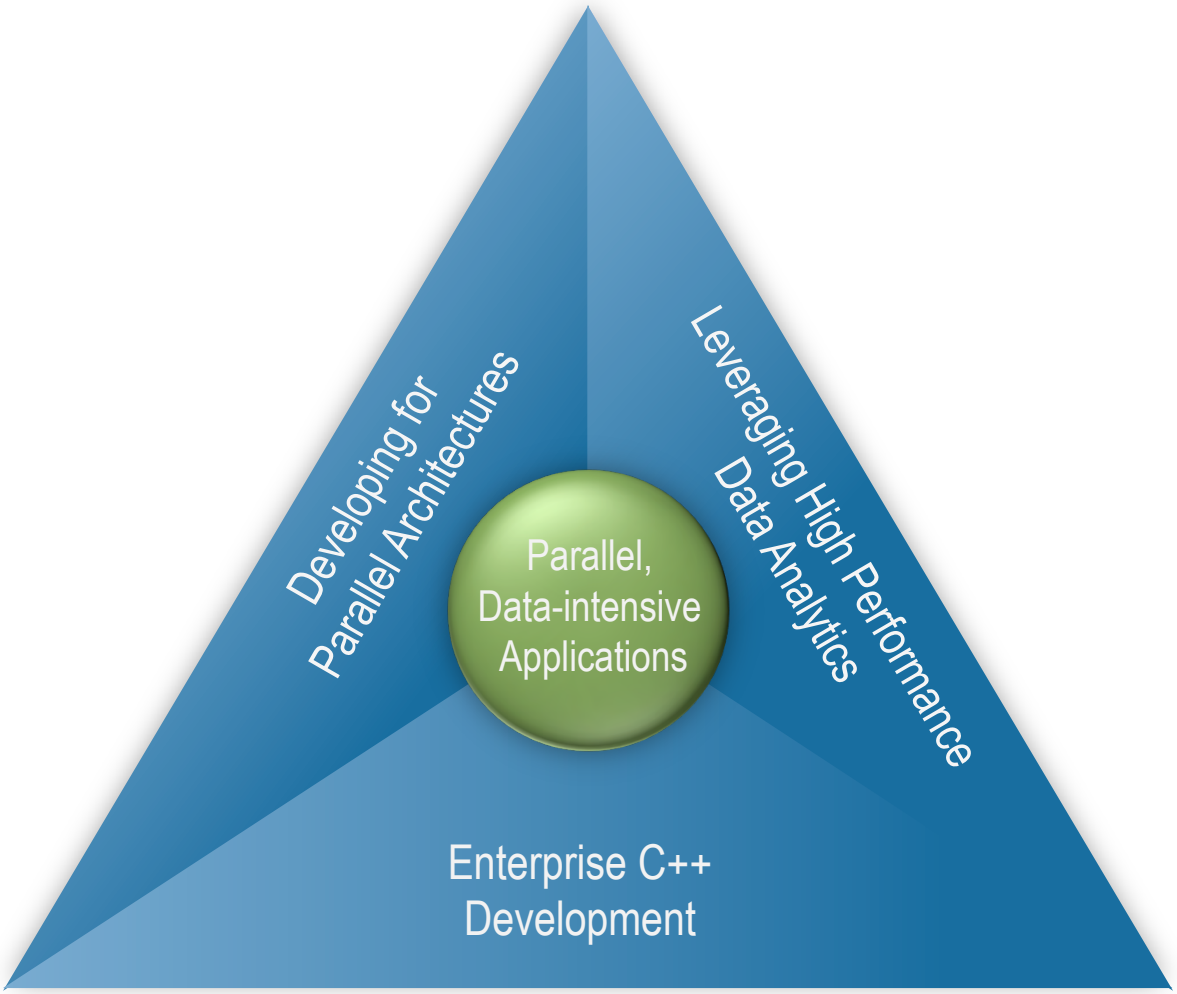
- **History**
  - Founded: 1989
  - Acquired by Battery Ventures: 2007
  - Acquired:
    - Visual Numerics: 2009
    - TotalView Technologies: 2010
    - Acumem: 2010

- **Customers**
  - 3,000+ in 36 countries
  - Financial services, telecoms, oil and gas, government and aerospace, research and academic

ROGUE WAVE SOFTWARE

Representative Customers

Rogue Wave Solution Portfolio



- **Development**
  - PyIMSL Studio
  - PV-Wave
  - SourcePro
  - IMSL Libraries
- **Debugging**
  - TotalView
  - ReplayEngine
  - MemoryScape
- **Optimization**
  - ThreadSpotter

ROGUE WAVE SOFTWARE

GPU architecture

Used in conjunction with conventional CPUs

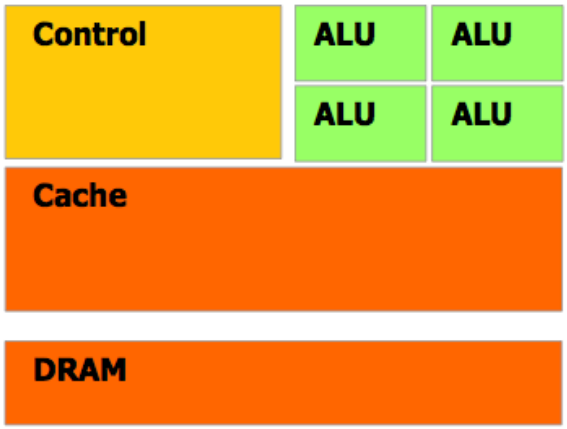
- Acts as an accelerator to a host process
- Or, perhaps the host processor acts to support the GPU

Distinct architecture

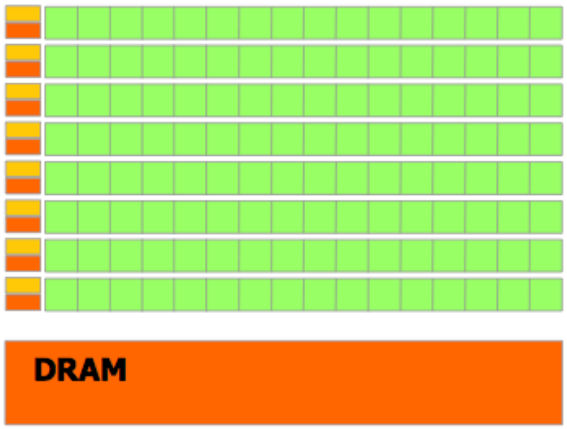
- Distinct processor architecture from the CPU

Many more cores than an SMP

- Multiple streaming multiprocessors
- Potentially 10k+ thread contexts



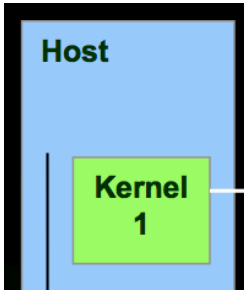
CPU



GPU

Programming for the GP-GPU

- CUDA
  - Function-like kernels are written for calculations to be performed on the GPU
    - Data parallel style, one kernel per unit of work
  - Presents a hierarchical organization for thread contexts
    - 2D or 3D 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



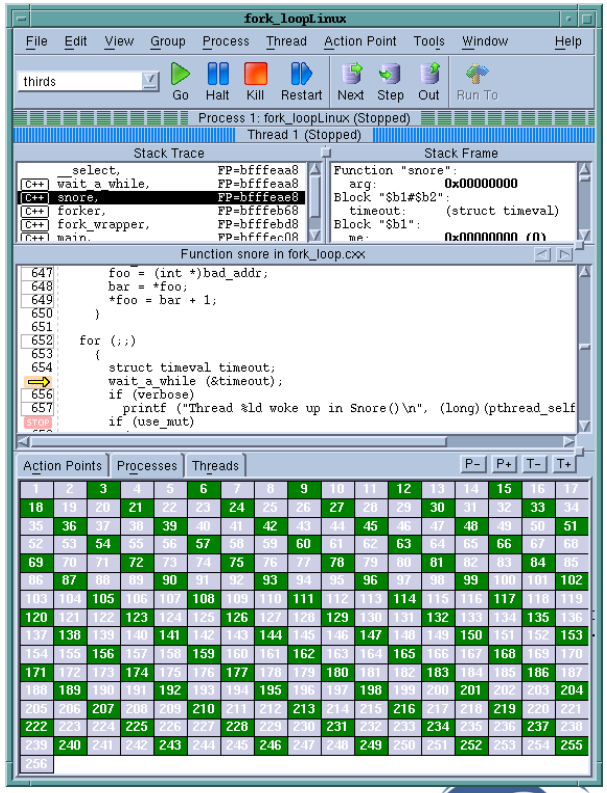
Programming challenges

- Coordinating CPU code + device code
- Understanding what is going on in each kernel
  - Exceptions
- Understanding memory usage
- Understanding performance characteristics

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**



ROGUE WAVE SOFTWARE

How can TotalView help you?

Effective Debugging requires the capability to control and examine specific instances of program execution in detail

- **Threads and/or MPI**
  - When you have
    - Deadlocks and hangs
    - Race conditions
  - It provides
    - Asynchronous thread control
    - Powerful group mechanism
- **Fortran and/or C++**
  - Complex data structures
    - Diving and recursive dive
  - STL Collection Classes
    - STLView
  - Rich class hierarchies
    - Powerful type-casting features
- **Memory Analysis**
  - Leaks and Bounds Errors
    - Automatic error detection tools
  - Out of Memory Errors
    - Analysis of heap memory usage by file function and line
- **Data Analysis**
  - Numerical errors
    - Extensible data visualization
    - Slicing and filtering of arrays
    - Powerful expression system
    - Conditional watchpoints

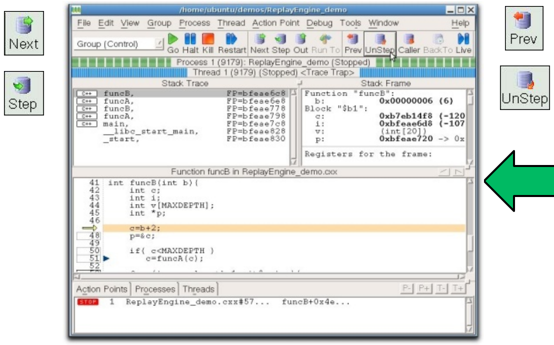
TotalView provides an answer to the question : "What is my program *really* doing?"

ROGUE WAVE SOFTWARE

TotalView Debugging Ecosystem

Debugging with TotalView

Reverse Debugging with ReplayEngine



```
funcB,               FP=bfeae6c8
funcA,               FP=bfeae6e8
funcB,               FP=bfeae778
funcA,               FP=bfeae798
main,                FP=bfeae7c8
__libc_start_main,   FP=bfeae828
_start,              FP=bfeae830
```

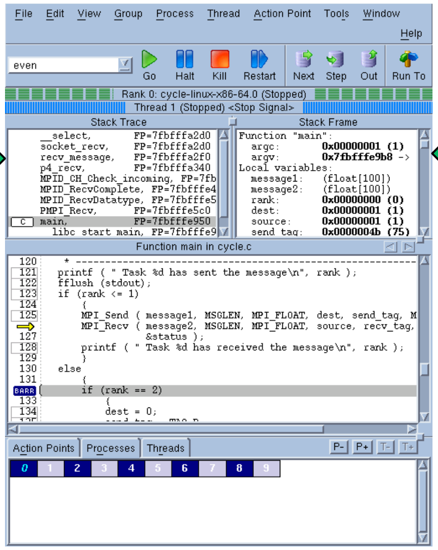
```
Function "funcB":
  b:          0x00000006 (6)
Block "$b1":
  c:          0xb7eb14f8 (-120
  i:          0xbfeae6d8 (-107
  v:          (int[20])
  p:          0xbfeae720 -> 0x
Registers for the frame:
```

```
int funcB(int b){
    int c;
    int i;
    int v[MAXDEPTH];
    int *p;

    c=b+2;
    p=&c;

    if( c<MAXDEPTH )
       c=funcA(c);
```

```
STOP  1  ReplayEngine_demo.cxx#57...  funcB+0x4e...
```

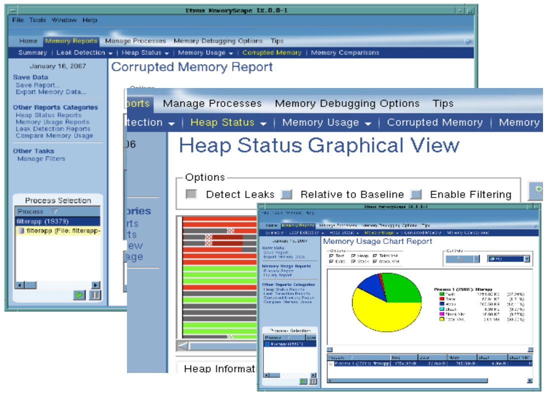


```
__select,             FP=7fbfffa2d0
socket_recv,          FP=7fbfffa2d0
recv_message,         FP=7fbfffa2f0
p4_recv,              FP=7fbfffa340
MPID_CH_Check_incoming, FP=7fb
MPID_RecvComplete,    FP=7fbfffe4
MPID_RecvDatatype,    FP=7fbfffe5
PMPI_Recv,            FP=7fbfffe5c0
main,                 FP=7fbfffe950
 libc start main,     FP=7fbfffe9
```

```
Function "main":
  argc:        0x00000001 (1)
  argv:        0x7fbfffe9b8 ->
Local variables:
  message1:    (float[100])
  message2:    (float[100])
  rank:        0x00000000 (0)
  dest:        0x00000001 (1)
  source:      0x00000001 (1)
  send tag:    0x0000004b (75)
```

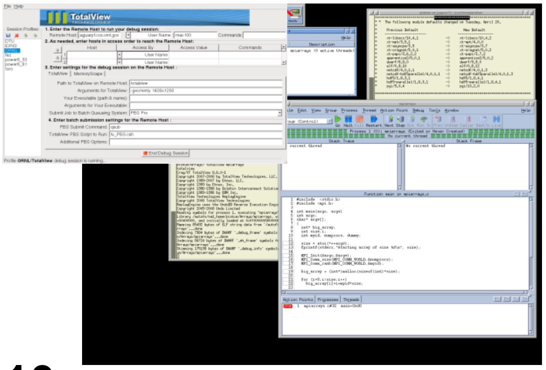
```
 * -------------------------------------------------------------
printf ( " Task %d has sent the message\n", rank );
fflush (stdout);
if (rank <= 1)
    {
    MPI_Send ( message1, MSGLEN, MPI_FLOAT, dest, send_tag, M
    MPI_Recv ( message2, MSGLEN, MPI_FLOAT, source, recv_tag,
               &status );
    printf ( " Task %d has received the message\n", rank );
    }
else
    {
    if (rank == 2)
        {
        dest = 0;
```

Memory Debugging with MemoryScape

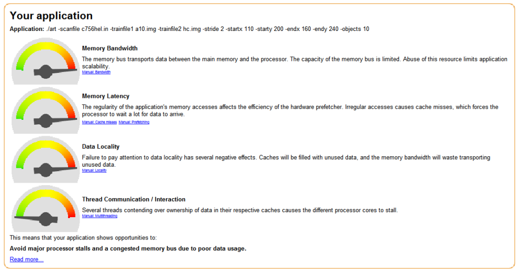


Remote Display Window

Easy Secure Fast



Optimizing with ThreadSpotter



Your application

Application: ./art -scanfile c756hel.in -trainfile1 a10.img -trainfile2 hc.img -stride 2 -startx 110 -starty 200 -endx 160 -endy 240 -objects 10

Memory Bandwidth

The memory bus transports data between the main memory and the processor. The capacity of the memory bus is limited. Abuse of this resource limits application scalability.

Manual: Bandwidth

Memory Latency

The regularity of the application's memory accesses affects the efficiency of the hardware prefetcher. Irregular accesses causes cache misses, which forces the processor to wait a lot for data to arrive.

Manual: Cache misses Manual: Prefetching

Data Locality

Failure to pay attention to data locality has several negative effects. Caches will be filled with unused data, and the memory bandwidth will waste transporting unused data.

Manual: Locality

Thread Communication / Interaction

Several threads contending over ownership of data in their respective caches causes the different processor cores to stall.

Manual: Multithreading

This means that your application shows opportunities to:

Avoid major processor stalls and a congested memory bus due to poor data usage.

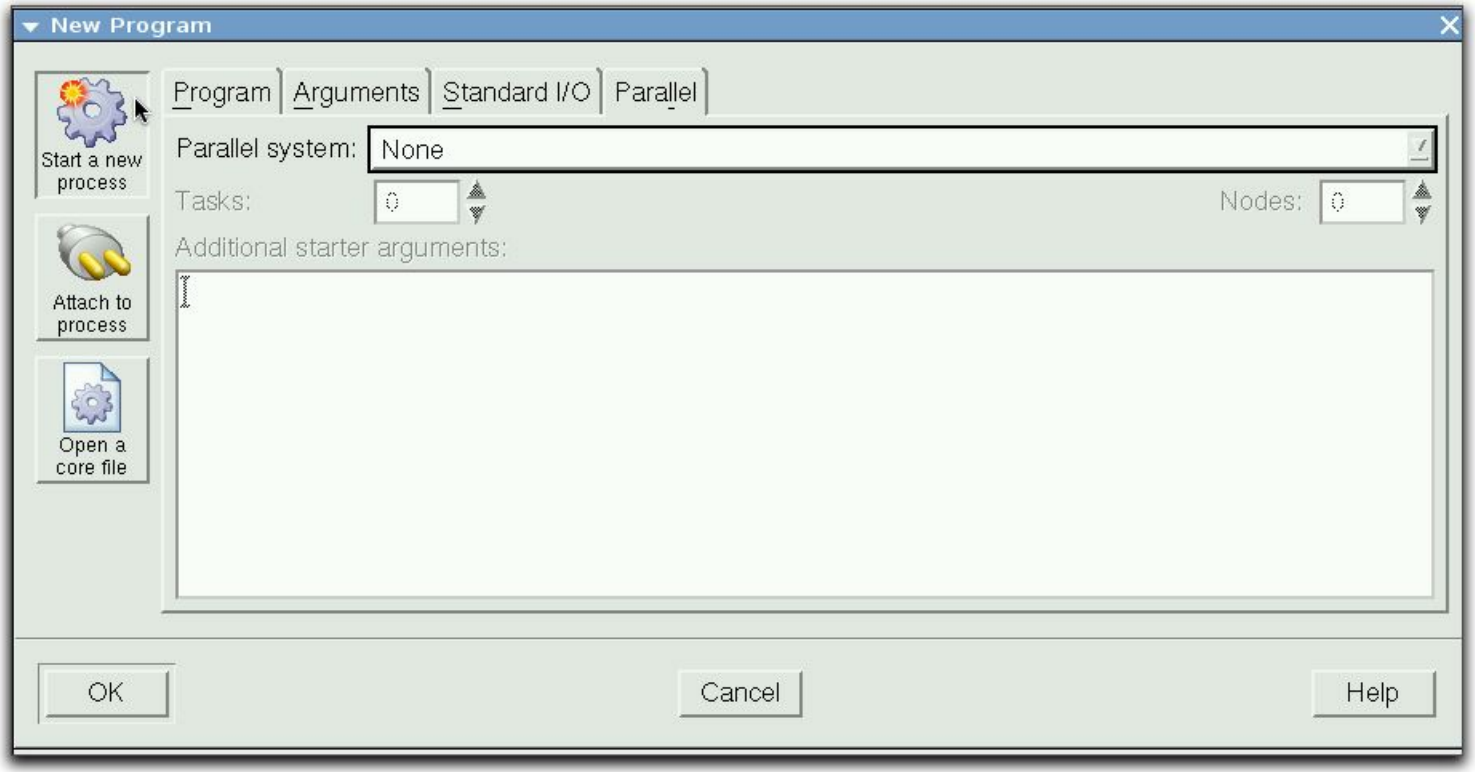
Read more...

Batch Debugging 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)
!      }
```

ROGUE WAVE SOFTWARE

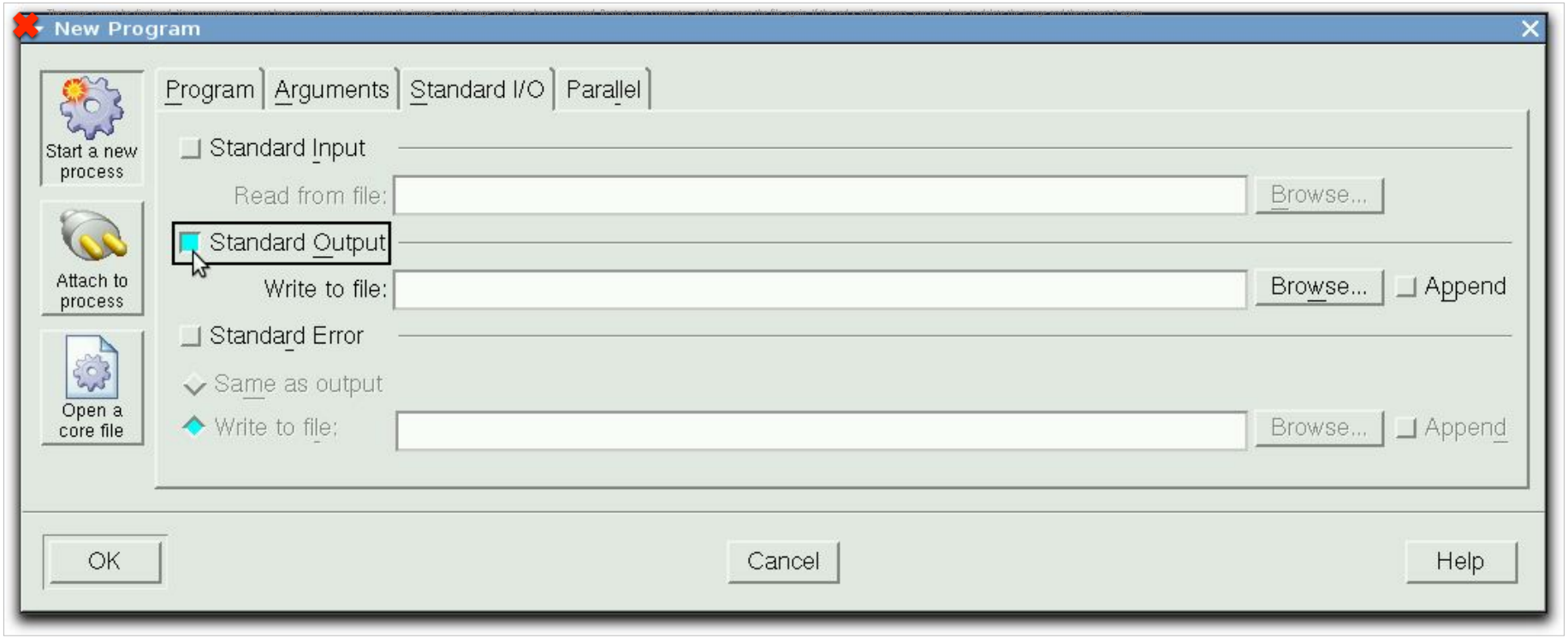
Starting TotalView



Open a Core File

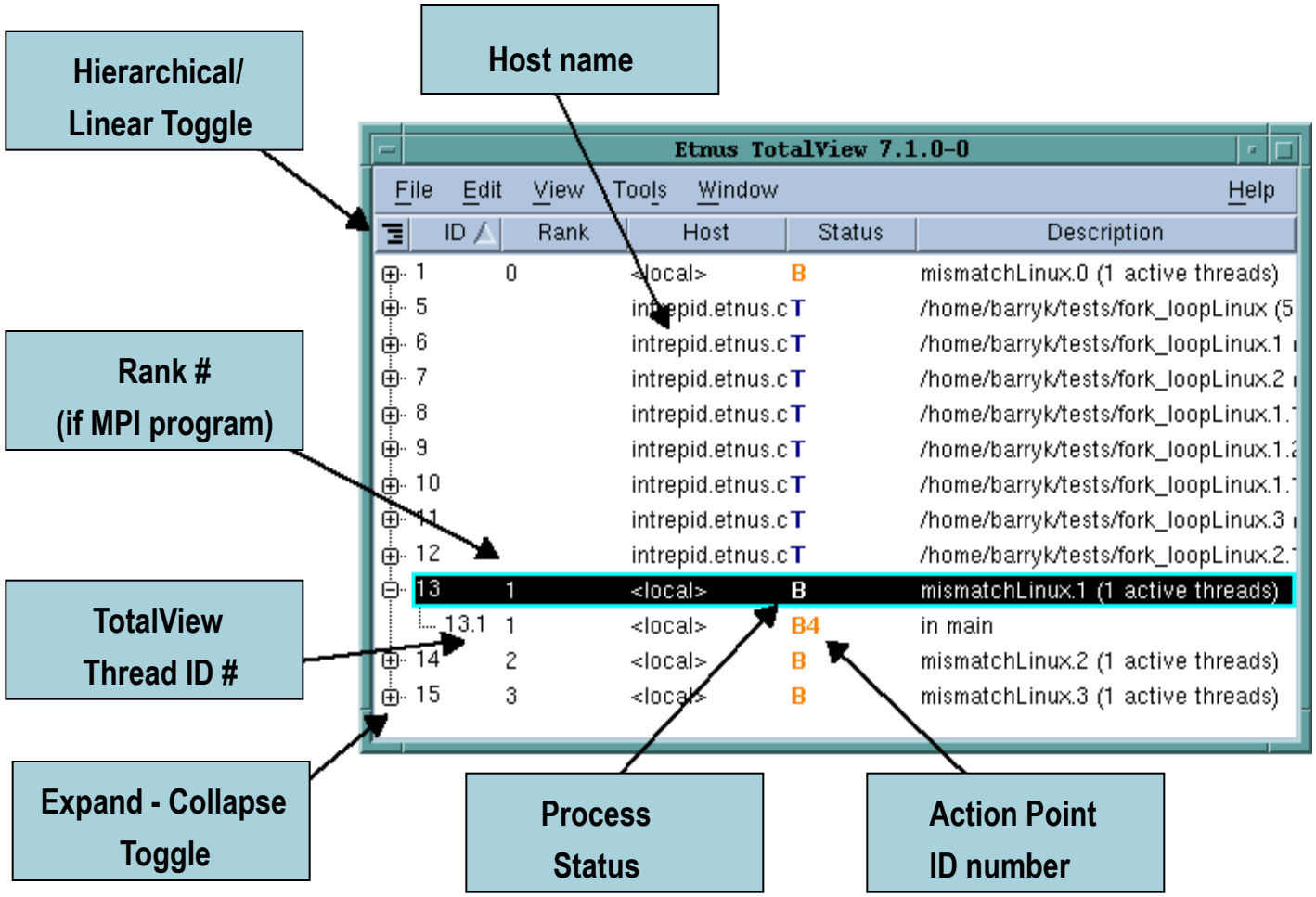
ROGUE WAVE SOFTWARE

TotalView Startup



ROGUE WAVE SOFTWARE

TotalView Root Window



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

ROGUE WAVE SOFTWARE

Process Window Overview

Stack Trace Pane

Toolbar

Stack Frame Pane

Source Pane

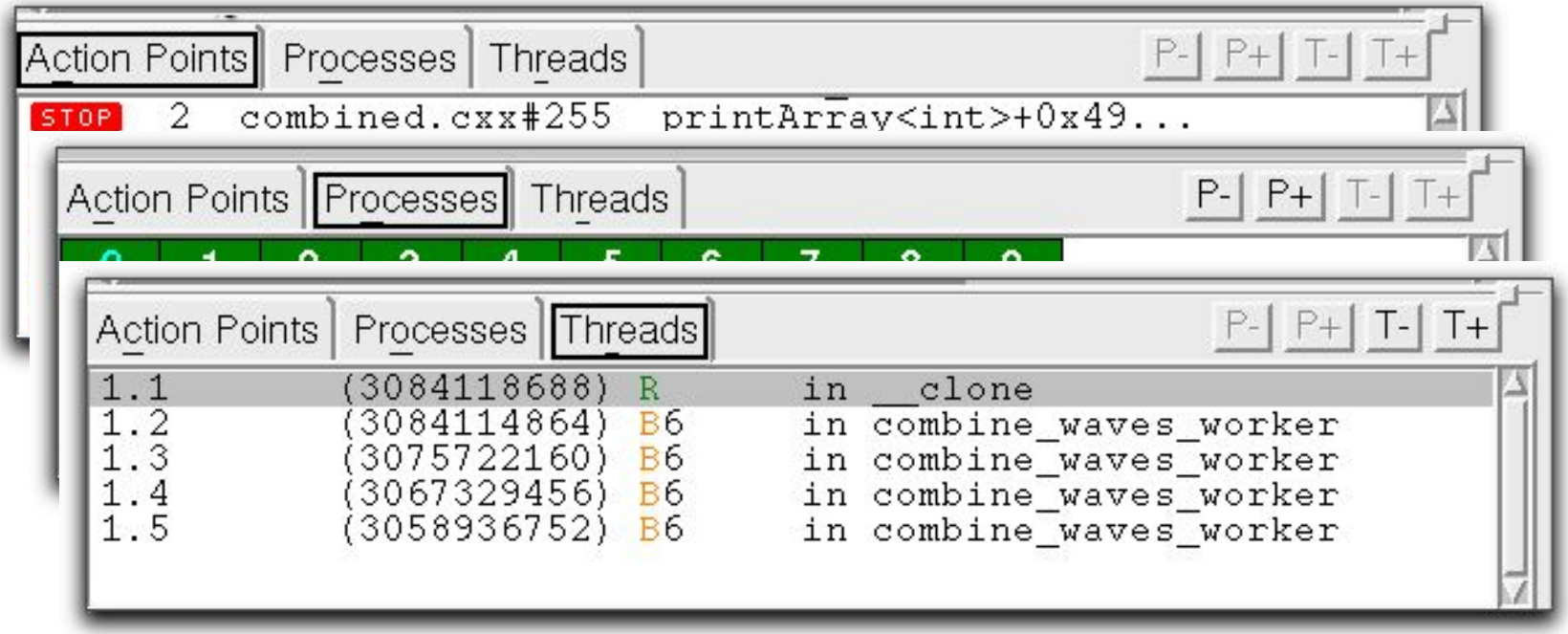
Tabbed Area

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

ROGUE WAVE SOFTWARE

Tabbed Pane



<u>Action Points Tab</u>
all currently defined action points

<u>Processes Tab</u>
all current processes

<u>Threads Tab:</u>
all current threads, ID's, Status

ROGUE WAVE SOFTWARE

Stepping Commands

Based on PC location

Out

Next

Step

Run To

Function sub1 in step.c

```
 2  int sub2(int);
 3  int sub3(int);
 4
 5  int main ()
 6  {
 7    int j, k, i = 0;
 8
 9    j = sub1(i); k = sub3(j);
10    printf ("The value of k is %d\n", k);
11  }
12
13  int sub1(int x)
14  {
15    x = sub2(2);
16    return (x++);
17  }
18
19  int sub2(int y)
20  {
21    y = y + 10;
22    y++;
23    return (y*10);
24  }
25
26  int sub3(int z)
27  {
28    return (z*z);
```


ROGUE WAVE SOFTWARE

Diving on Variables

Example: Dive on Variable "j" from Stack Frame or Source Panes

Process 1 (4508): combined (At Breakpoint 11)

j - arrays - 1.1

File Edit View Tools Window Help

1.1

Expression: j Address: 0xbf9

Type: int

Value

0x00000006 (6)

(At Breakpoint 11)

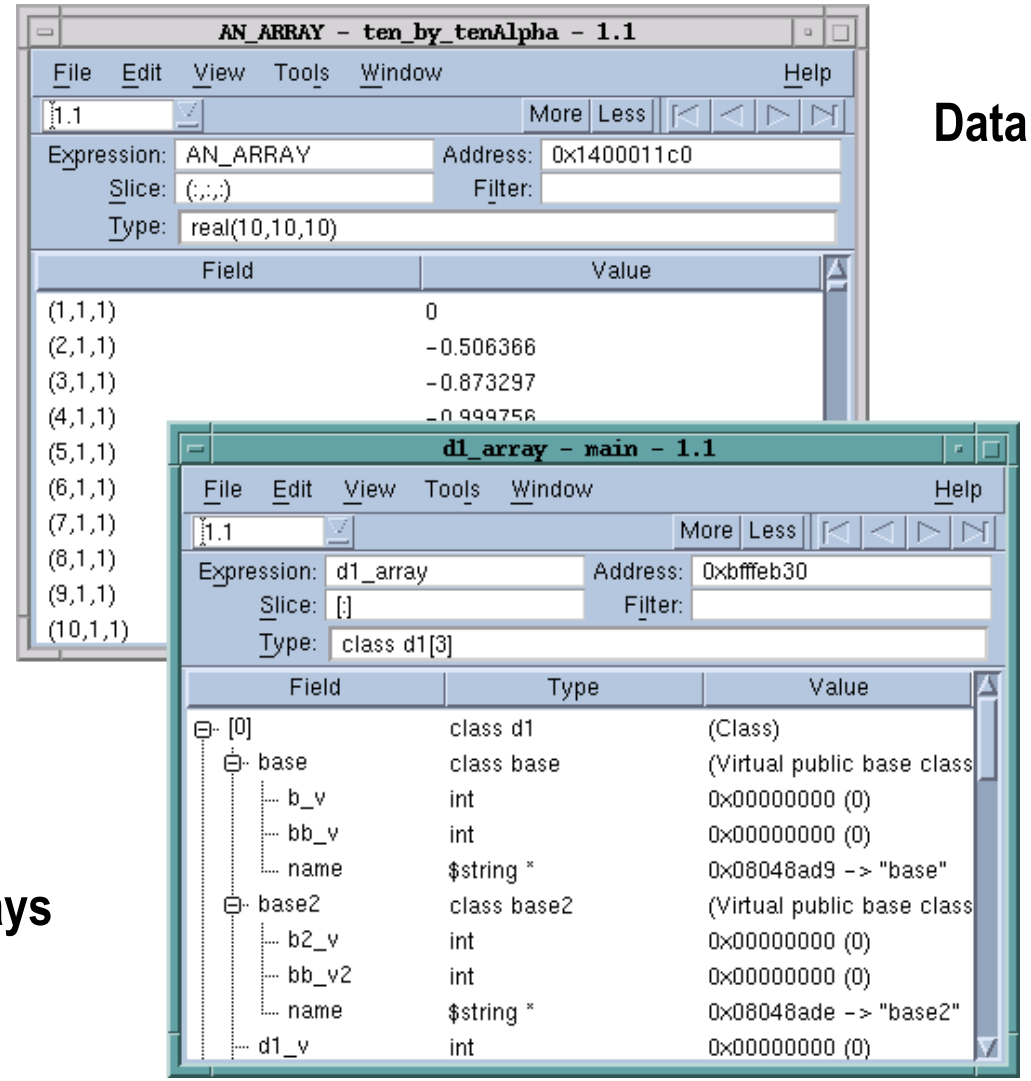
Stack Frame

```
Function "arrays":
  No parameters.
Block "$b1#$b1":
  j:                0x00000006 (6)
Block "$b1":
  i:                0x00000000 (0)
Local variables:
  shape:            (class Shape)
  circle:           (class Circle)
  cylinder:         (class Cylinder)
```

ined.cxx

```
506    cout << "circe = " << (void *)circe << "\n";
507    //  cout << "*circe = " << (void)  *circe << "\n";
508    cout << "&circe = " << (void *) &circe <<  "\n";
509
510    for( int i = 0; i < ARRAYSIZE; i++ ) {
511      for (int j = 0; j < ARRAYSIZE; j++) {
512            cylinder.resize(start + i*step, start + j*step);
513            surf_vol[i][j] = cylinder.volume()/cylinder.area();
514            vol[i][j] = cylinder.volume();
515      }
```

Viewing Arrays



Data Arrays

Structure Arrays

18

ROGUE WAVE SOFTWARE

Slicing Arrays

two_d_array - arraysLinux - 2.1

File Edit View Tools Window Help

2.1 More Less

Expression: two_d_array Address: 0x08097dc0

Slice: (6:10,6:10) Filter:

Type: INTEGER*4(10,10)

Field	Value
(6,6)	216 (0x000000d8)
(7,6)	294 (0x00000126)
(8,6)	384 (0x00000180)
(9,6)	486 (0x000001e6)
(10,6)	600 (0x00000258)
(6,7)	252 (0x000000fc)
(7,7)	343 (0x00000157)
(8,7)	448 (0x000001c0)
(9,7)	567 (0x00000237)
(10,7)	700 (0x000002bc)

Slice notation is `[start:end:stride]`

ROGUE WAVE SOFTWARE

Filtering Arrays

ieee_array - /nfs/netapp0/user/home/barryk/tests/arraysAlpha - 3.

File Edit View Tools Window Help

3.1

Expression: ieee_array Address: 0x1408214a0

Slice: (:) Filter: .eq. $inf

Type: $real_4(6)

Field	Value
(1)	INF
(2)	-INF

ieee_array - /nfs/netapp0/user/home/barryk/tests/arraysAlpha - 3.

File Edit View Tools Window Help

3.1

Expression: ieee_array Address: 0x1408214a0

Slice: (:) Filter: .eq. $denorm

Type: $real_4(6)

Field	Value
(5)	1.4013e-45 <denormalized>
(6)	-1.4013e-45 <denormalized>

ieee_array - /nfs/netapp0/user/home/barryk/tests/arraysAlpha - 3

File Edit View Tools Window

3.1

Expression: ieee_array Address: 0x1408214a0

Slice: (:) Filter:

Type: $real_4(6)

Field	Value
(1)	INF
(2)	-INF
(3)	NaNQ
(4)	NaNS
(5)	1.4013e-45 <denormalized>
(6)	-1.4013e-45 <denormalized>

int2_array__ - MAIN__ - 1.1

File Edit View Tools Window Help

1.1

Expression: int2_array__ Address: 0xbfffd450

Slice: (:) Filter: $value > 20 .and. $value < 100

Type: word(100) More

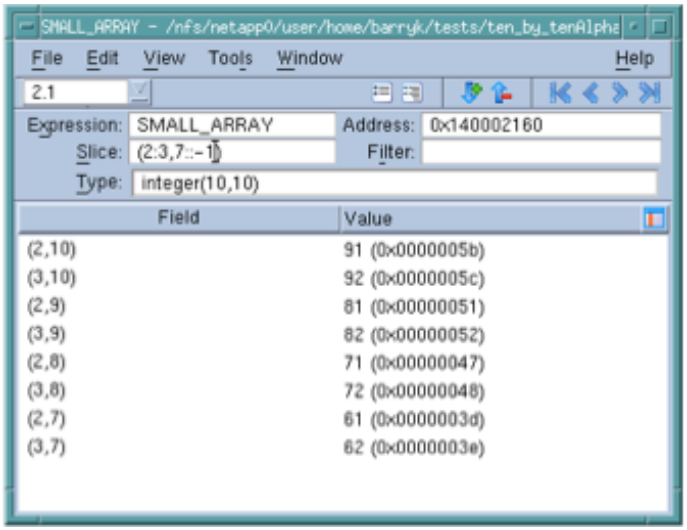
Field	Value
(16)	22 (0x0016)
(17)	24 (0x0018)
(18)	26 (0x001a)
(19)	28 (0x001c)
(20)	30 (0x001e)
(21)	32 (0x0020)
(22)	34 (0x0022)
(23)	36 (0x0024)
(24)	38 (0x0026)
(25)	40 (0x0028)

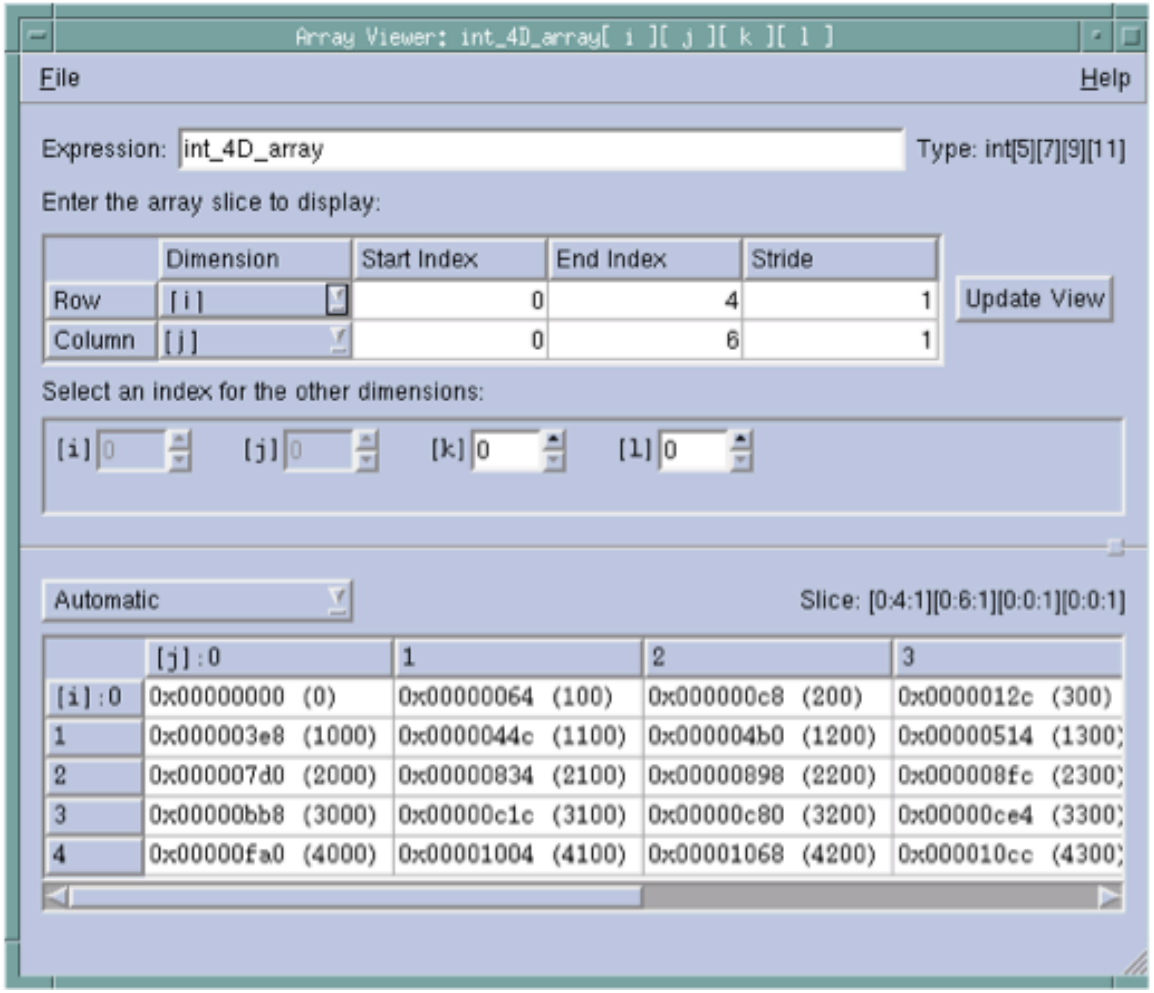
20

ROGUE WAVE SOFTWARE

Multi-Dimensional Array Viewer

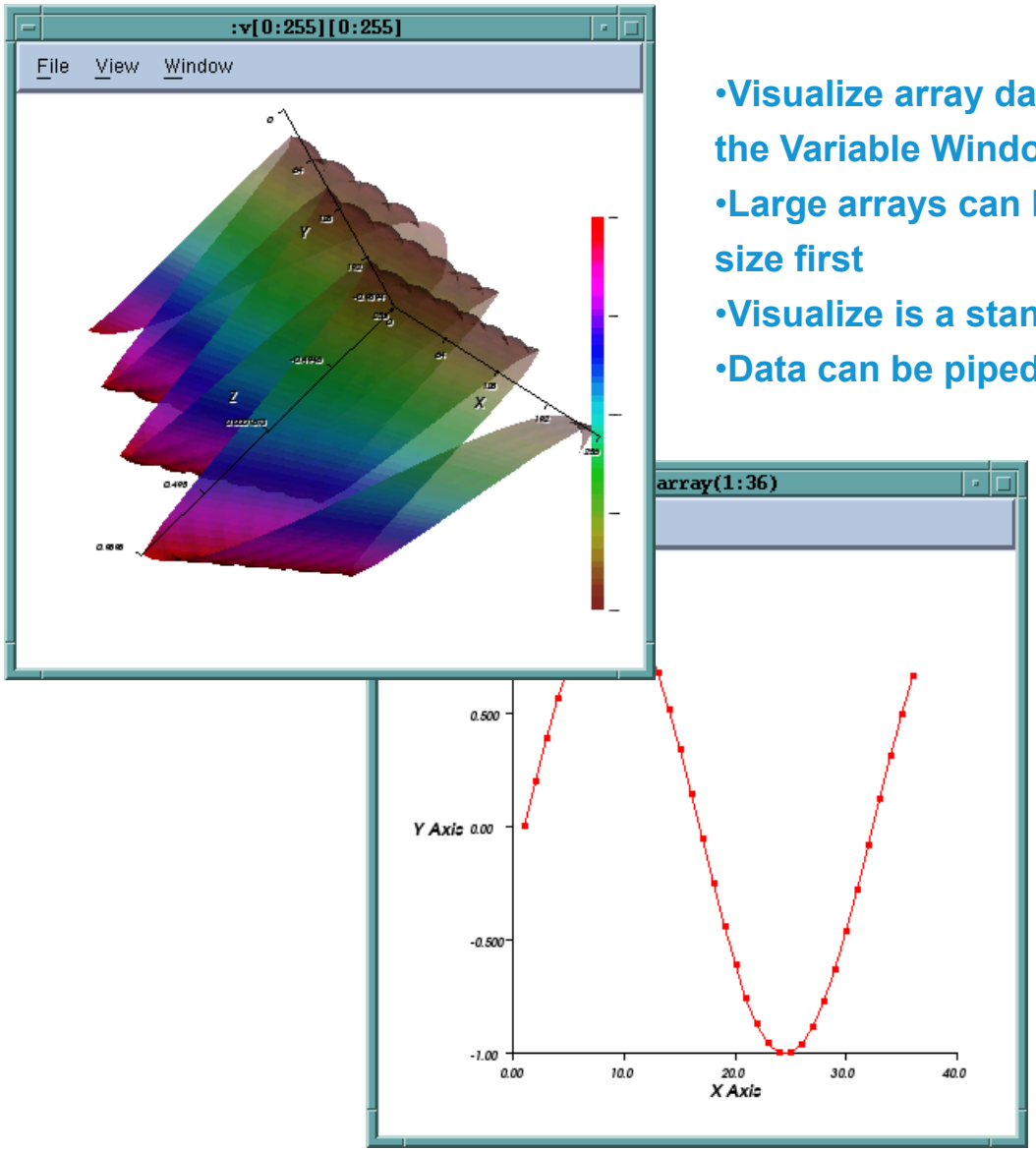
- See your arrays on a “Grid” display
- 2-D, 3-D… N-D
- Arbitrary slices
- Specify data representation
- Windowed data access
  - Fast





ROGUE WAVE SOFTWARE

Visualizing Arrays

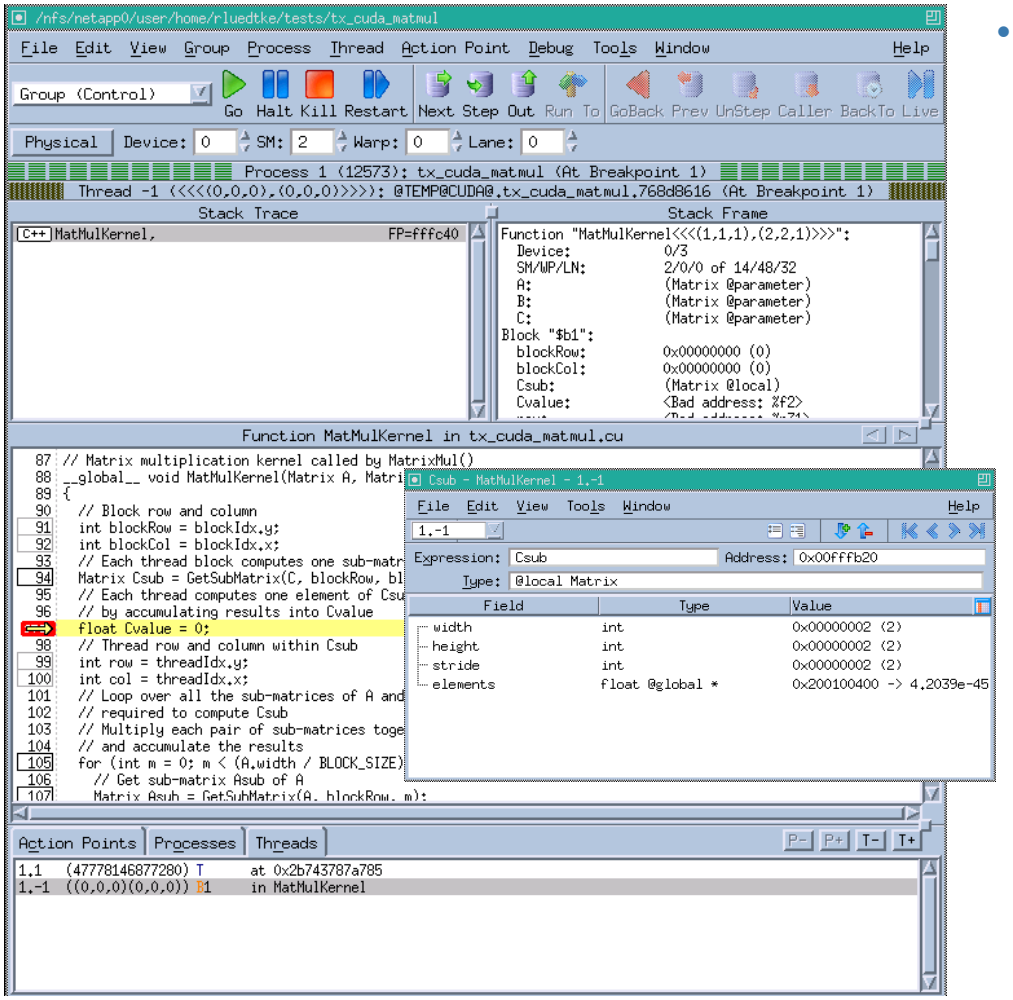


- Visualize array data using Tools > Visualize from the Variable Window
- Large arrays can be sliced down to a reasonable size first
- Visualize is a standalone program
- Data can be piped out to other visualization tools

- 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.

22

TotalView for CUDA



- **Characteristics**
  - Full visibility of both Linux threads and GPU device threads
  - Fully represent the hierarchical memory
  - Detailed device status display
  - Supports Unified Virtual Addressing and GPUDirect
  - Thread and Block Coordinates
  - Device thread control
  - Handles both inlined functions and CUDA callstack
  - Support for CUDA C++
  - Reports memory access errors
  - Handles CUDA exceptions and assert
  - Full Multi-Device Support
  - Can be used with MPI
- TV 8.9.2 supports CUDA 3.2 and 4.0
- TV 8.10 will support CUDA 4.1

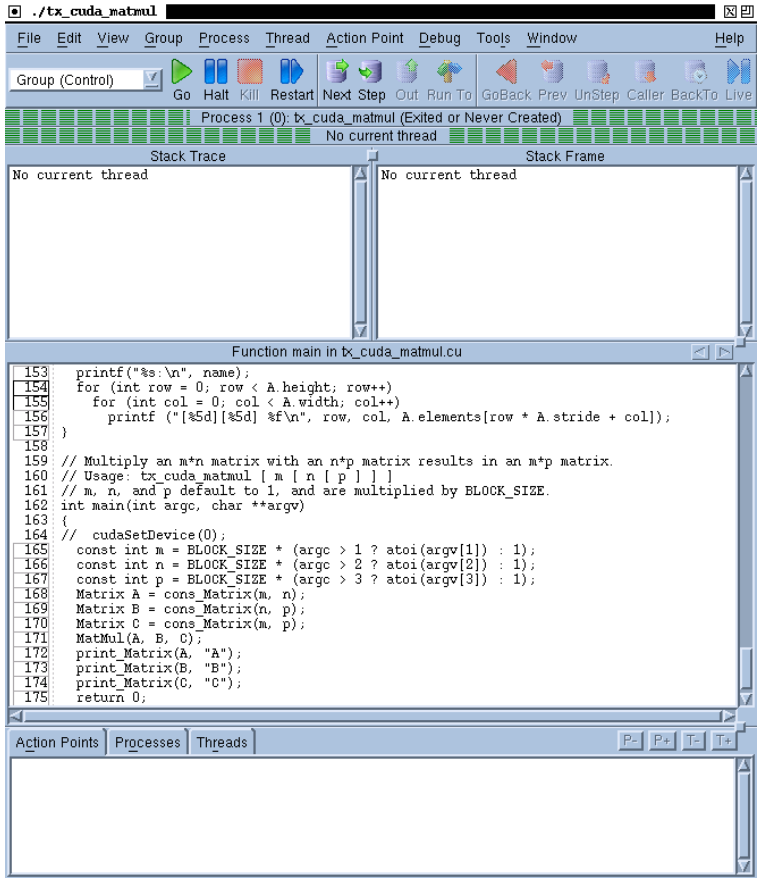
ROGUE WAVE SOFTWARE

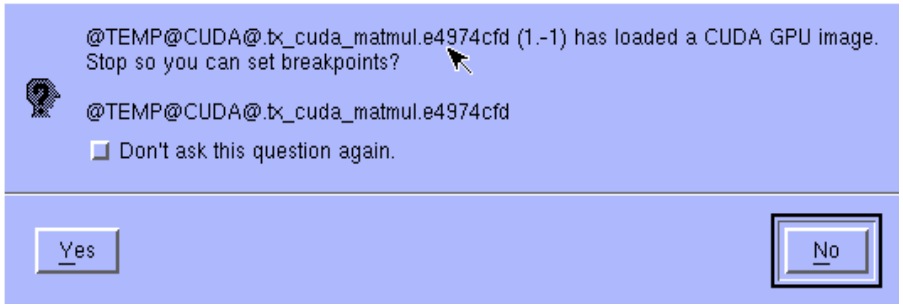
TV 8.10 support for CUDA 4.1 specific features

- Works with the CUDA 4.1 SDK and Runtime
  - New Compiler Front End
  - New Debug API
- Support for no copy pinned memory
  - This was broken at the driver level in 4.0
- New support for CUDA device assertions
- New support for multiple CUDA contexts from the same process on the same device
- Support for CUDA on the Cray XK environment

ROGUE WAVE SOFTWARE

Starting TotalView

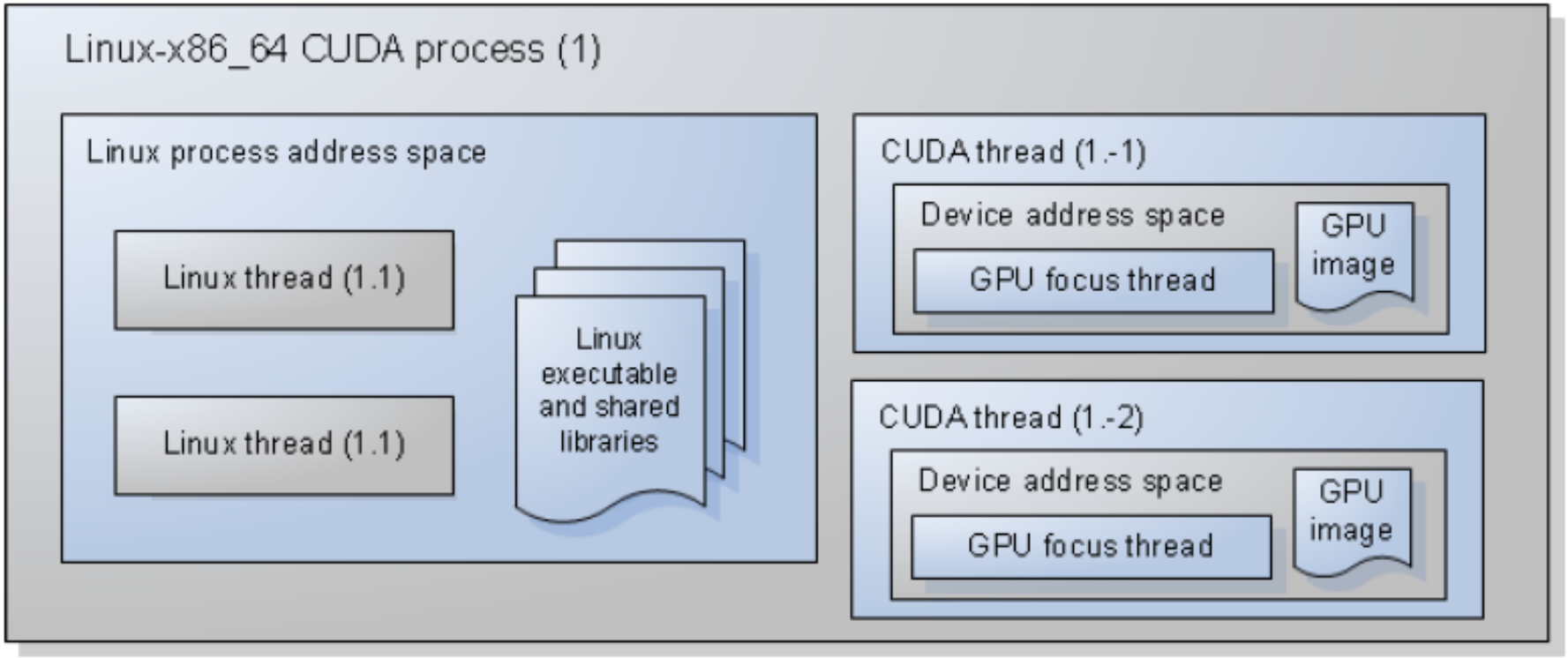




- When a new kernel is loaded you get the option of setting breakpoints

- You can debug the CUDA host code using the normal TotalView commands and procedures

TotalView CUDA Debugging Model



ROGUE WAVE SOFTWARE

Debugging CUDA

Thread (x,y,z)

Block (x,y,z)

GPU focus thread selector for changing the block (x,y) and thread (x,y,z) indexes of the CUDA thread

Select a line number in a box to plant a breakpoint

CUDA host threads have a positive TotalView thread ID

CUDA GPU threads have a negative TotalView thread ID

Running to a Breakpoint in the GPU code

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Call

Physical Device: 0 SM: 2 Warp: 0 Lane: 0

Process 1 (26318): tx_cuda_matmul (At Breakpoint 2)

hread -1 (<<<(0,0,0),(0,0,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.768d8616 (At Breakpoint

Stack Trace

```
C++ MatMulKernel.          FP=
```

Stack Frame

```
Function "MatMulKernel<<<(1,1,1),(2,2,1)>>>":
  Device:                 0/3
  SM/WP/LN:               2/0/0 of 14/48/32
  A:                      (Matrix @parameter)
  B:                      (Matrix @parameter)
  C:                      (Matrix @parameter)
Block "$b1#$b1#$b1":
  Asub:                   (Matrix @local)
  Bsub:                   (Matrix @local)
  As:                     (float[2][2] @shared)
  Bs:                     (float[2][2] @shared)
Block "$b1#$b1":
  m:                 0x00000000 (0)
Block "$b1":
  blockRow:          0x00000000 (0)
  blockCol:          0x00000000 (0)
  Csub:                   (Matrix @local)
  Cvalue:            0
  row:               0x00000000 (0)
  col:               0x00000000 (0)
```

Function MatMulKernel in tx_cuda_matmul.cu

```
 89 {
 90   // Block row and column
 91   int blockRow = blockIdx.y;
 92   int blockCol = blockIdx.x;
 93   // Each thread block computes one sub-matrix Csub of C
 94   Matrix Csub = GetSubMatrix(C, blockRow, blockCol);
 95   // Each thread computes one element of Csub
 96   // by accumulating results into Cvalue
 97   float Cvalue = 0;
 98   // Thread row and column within Csub
 99   int row = threadIdx.y;
100   int col = threadIdx.x;
101   // Loop over all the sub-matrices of A and B that are
102   // required to compute Csub
103   // Multiply each pair of sub-matrices together
104   // and accumulate the results
105   for (int m = 0; m < (A.width / BLOCK_SIZE); ++m) {
106     // Get sub-matrix Asub of A
107     Matrix Asub = GetSubMatrix(A, blockRow, m);
108     // Get sub-matrix Bsub of B
109     Matrix Bsub = GetSubMatrix(B, m, blockCol);
110     // Shared memory used to store Asub and Bsub respectively
111     __shared__ float As[BLOCK_SIZE][BLOCK_SIZE];
112     __shared__ float Bs[BLOCK_SIZE][BLOCK_SIZE];
113     // Load Asub and Bsub from device memory to shared memory
114     // Each thread loads one element of each sub-matrix
115     As[row][col] = GetElement(Asub, row, col);
```

Action Points | Processes | Threads P- P+ T- T+

```
2  tx cuda matmul.cu#107  MatMulKernel+0x390...
```

GPU focus thread logical coordinates

Stack backtrace (3.2) and inlined functions (3.1)

PC arrow for the warp

CUDA grid and block dimensions, lanes/warp, warps/SM, SMs, etc.

Parameter, register, local and shared variables

Dive on a variable name to open a variable window

ROGUE WAVE SOFTWARE

Stepping GPU Code

- single-step operation advances all of the GPU hardware threads in the *same* warp
- To advance the execution of more than one warp, you may either:
  - set a breakpoint and continue the process, or
  - select a line number in the source pane and select “Run To”.

ROGUE WAVE SOFTWARE

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**

Name	Description
Device 0/3	
Device Type	gf100
Lanes	32
SM 2/1	
Valid Warps	0000000000000001
Warp 00/48	Block (0,0,0)
Lane 00/32	Thread (0,0,0)
PC	0000000019aa94d8
Lane 01/32	Thread (1,0,0)
PC	0000000019aa94d8
Lane 02/32	Thread (2,0,0)
PC	0000000019aa94f0
Lane 03/32	Thread (3,0,0)
PC	0000000019aa94f0
Lane 04/32	Thread (4,0,0)
PC	0000000019aa94f0
Lane 05/32	Thread (5,0,0)
PC	0000000019aa94f0
Lane 06/32	Thread (6,0,0)
PC	0000000019aa94f0
Lane 07/32	Thread (7,0,0)
PC	0000000019aa94f0
Lane 08/32	Thread (8,0,0)
PC	0000000019aa94f0
Lane 09/32	Thread (9,0,0)
PC	0000000019aa94f0
Valid/Active/Divergent	000003ff, 000003fc, 00000003
SM Type	sm_20
SMs	14
Warps	48
Device 1/3	
Device Type	gt200
Lanes	32
SM Type	sm_13

Example of Divergent GPU threads

Different PC for two groups of Lanes

State of Lanes inside the warp

ROGUE WAVE SOFTWARE

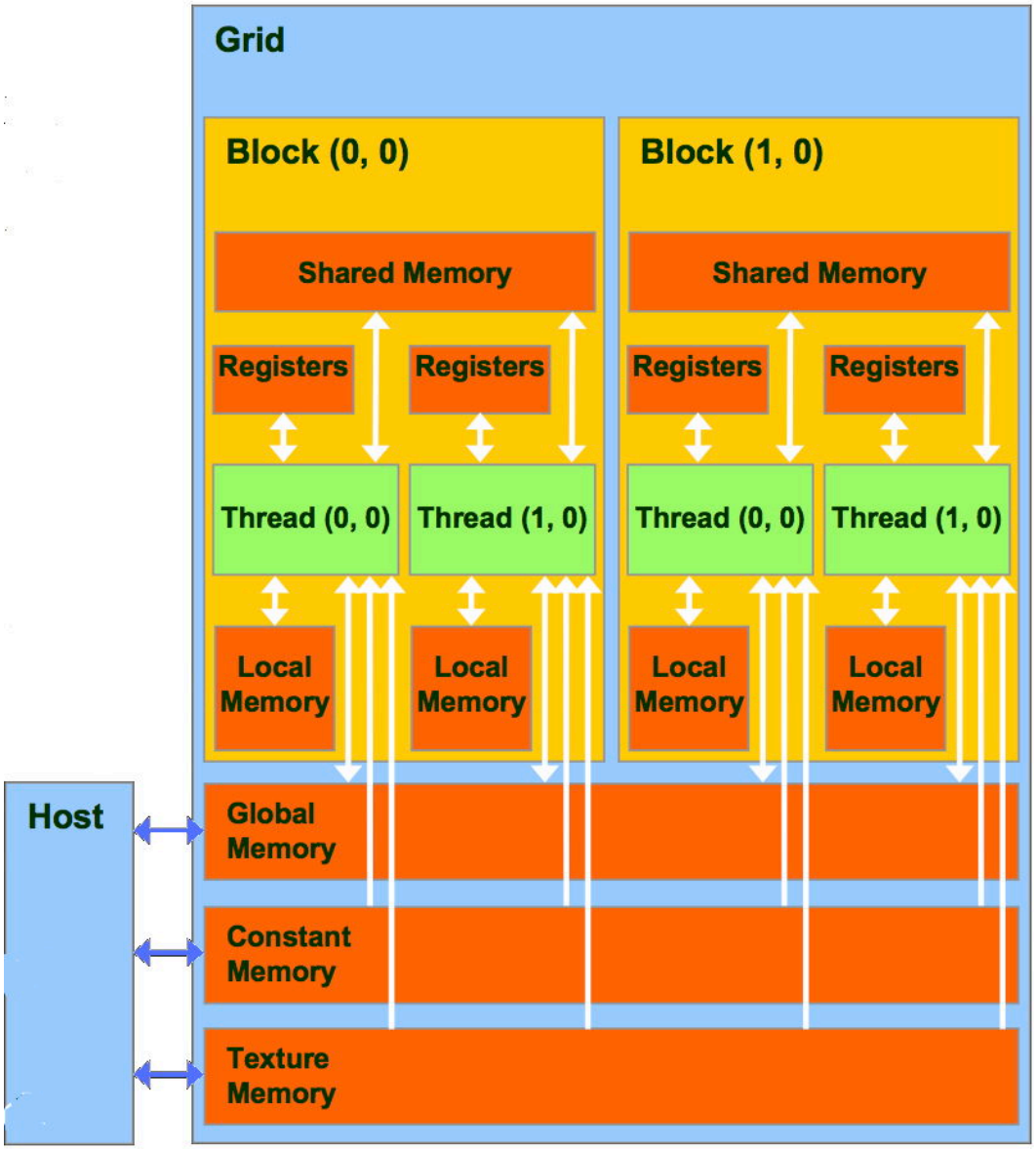
CUDA Memory Types

Memory	Scope	Locality
Global	Device	External
Shared	Block	Chip
Local	Thread	Chip
Constant	Device	Chip (Cache)
Texture	Device	Chip (Cache)
Register	Thread	Chip

ROGUE WAVE SOFTWARE

GPU Memory Hierarchy

- Hierarchical memory with many layers
  - Local (thread)
  - Shared (block)
  - Global (GPU)
  - System (host)



TotalView Type Storage Qualifiers

@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.

ROGUE WAVE SOFTWARE

HRL Case Study

- Center for Neural and Emergent Systems at HRL
- Using a CUDA accelerated cluster to model the brain

- "In the first full day of using TotalView, we were quickly able to solve the bug that had us stumped for weeks. With TotalView we were able to step into a specific thread, and then into specific CUDA kernels to identify what went wrong. We could resolve the bugs quickly, and focus our development effort on adding features."

- "We noticed a dramatic drop in our development cycle – what used to take us more than two weeks to develop and fully test now takes less than one week. By scaling down the development cycle we were able to add more features, even going beyond the requirements of our release cycle. Most important, we were able to focus on the performance of our code, resulting in much better utilization of our existing hardware and allowing us to scale past 100 GPUs."

- For more information look at the HRL case study on the following page

  http://www.roguewave.com/resources/case-studies.aspx

ROGUE WAVE SOFTWARE

Thanks!

- **Contact me**
  - Chris.Gottbrath@RogueWave.com
  - Ed.Hinkel@RogueWave.com
- **or for more information**

  Check out: www.roguewave.com

  Email: sales@roguewave.com

ROGUE WAVE SOFTWARE

GPU TECHNOLOGY CONFERENCE

Register for GPU Tech Conference 2012
May 14-17 | San Jose, CA

By the numbers…

- 4 full days
- 1000s of developers, computational scientists, and researchers
- 3 keynotes
- 275 sessions
- 30 topic areas
- 150 research posters
- 2 superb co-located events – Los Alamos HPC Symposium & InPar 2012
- 1 Emerging Companies Summit
- Limitless opportunities for formal and informal networking

Register at www.gputechconf.com

Thank You