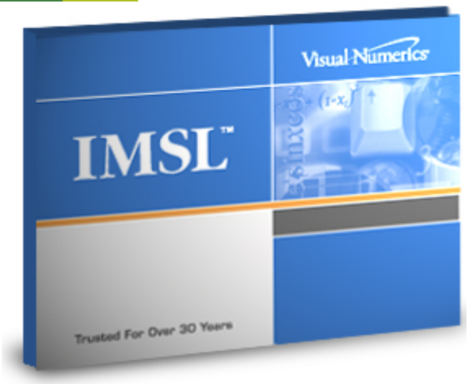
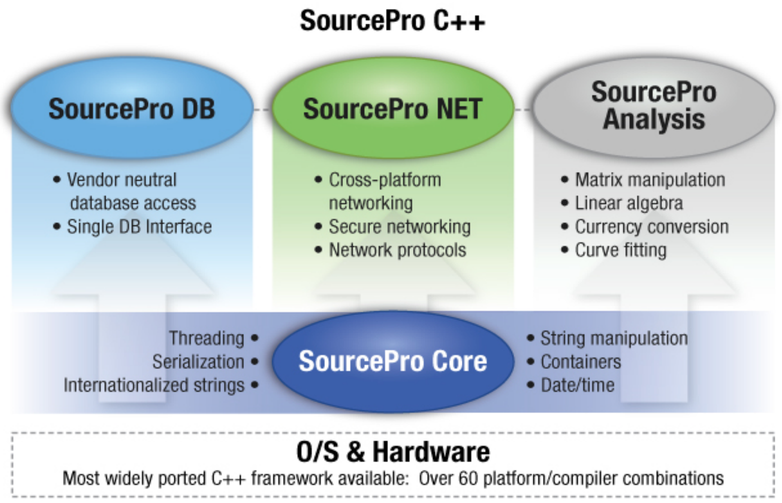
Improving the Productivity of Scalable Application Development with TotalView

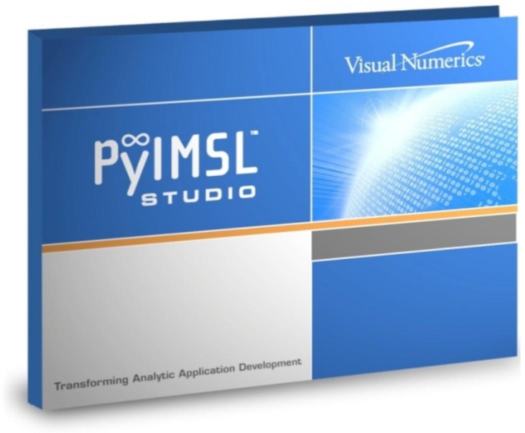
May 18th, 2010

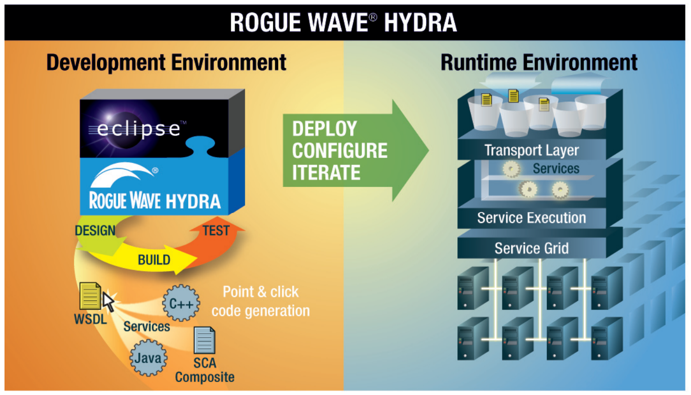
Chris Gottbrath
Principal Product Manager

Rogue Wave Major Product Offerings









Eclipse is a trademark of the Eclipse Foundation, Inc.

TotalView Technologies Family of Products

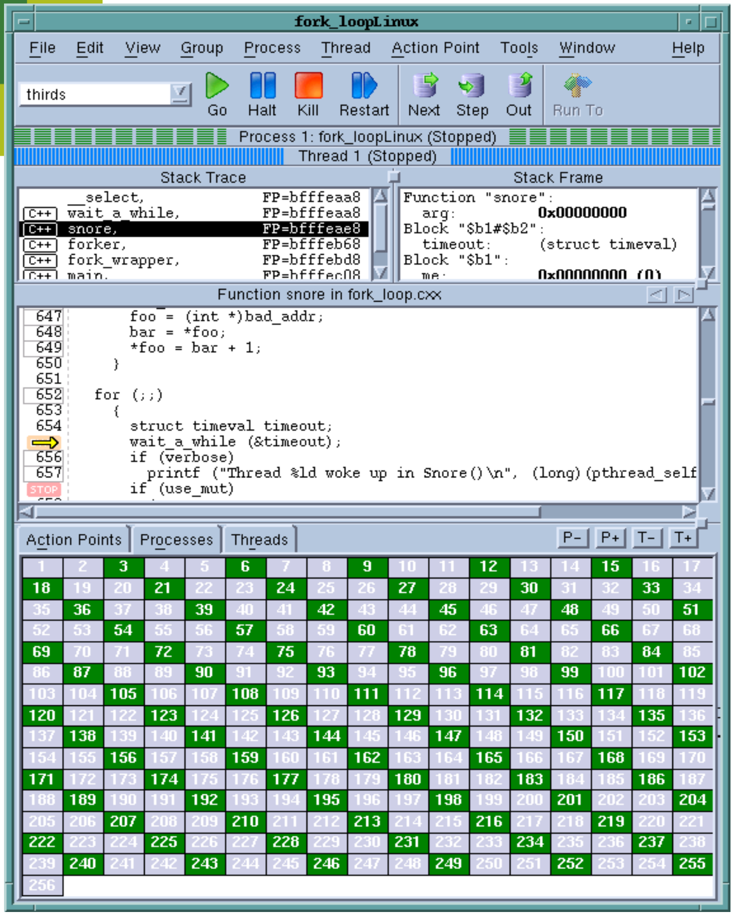
- **TotalView**
  - Highly scalable interactive GUI debugger
    - Supports basic and advanced usage
    - Used from workstations up to the largest HPC systems
  - Makes developers more productive and reduces project risks
  - Improves trace and automated debugging

- **MemoryScape**
  - Parallel memory error detection and memory analysis
  - Inductive user interface
    - Easy for those use use tools less frequently
  - Easily integrated into validation process

- **ReplayEngine**
  - Parallel record and deterministic replay add-on for TotalView
  - Radically simplifies many debugging tasks
  - Allows straightforward investigation of otherwise stochastic bugs

TotalView Technologies

What is TotalView?


fork_loopLinux
File Edit View Group Process Thread Action Point Tools Window Help
thirds
Go Halt Kill Restart Next Step Out Run To
Process 1: fork_loopLinux (Stopped)
Thread 1 (Stopped)
Stack Trace
Stack Frame
Function snore in fork_loop.cxx
Action Points Processes Threads
P- P+ T- T+

```
      __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)
```

```
      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)
```

- **What is TotalView?**
  - Parallel and Multithreaded Debugging and Analysis Tool
  - For scientists and engineers working with C/C++ and Fortran
  - Makes developing, maintaining and supporting critical and cutting edge applications easier and less risky
- **Major Features**
  - Supports Linux, Unix and Mac OS X
  - Parallel Debugging
    - MPI, Pthreads, OMP, UPC
  - Includes a **Remote Display Client** freeing users to work from anywhere
  - Memory Debugging with **MemoryScape**
  - Optional Reverse Debugging with **ReplayEngine**
  - Batch Debugging with TVSCript and the CLI
- **Advantages**
  - Easy to learn graphical user interface with data visualization
  - Wide variety of features so users can tackle unexpected bugs
  - Consistent functionality and look and feel across a wide range of platforms
  - Works robustly with open source and vendor compilers
  - Native debugger core is highly scalable to large clusters, large code and massive datasets

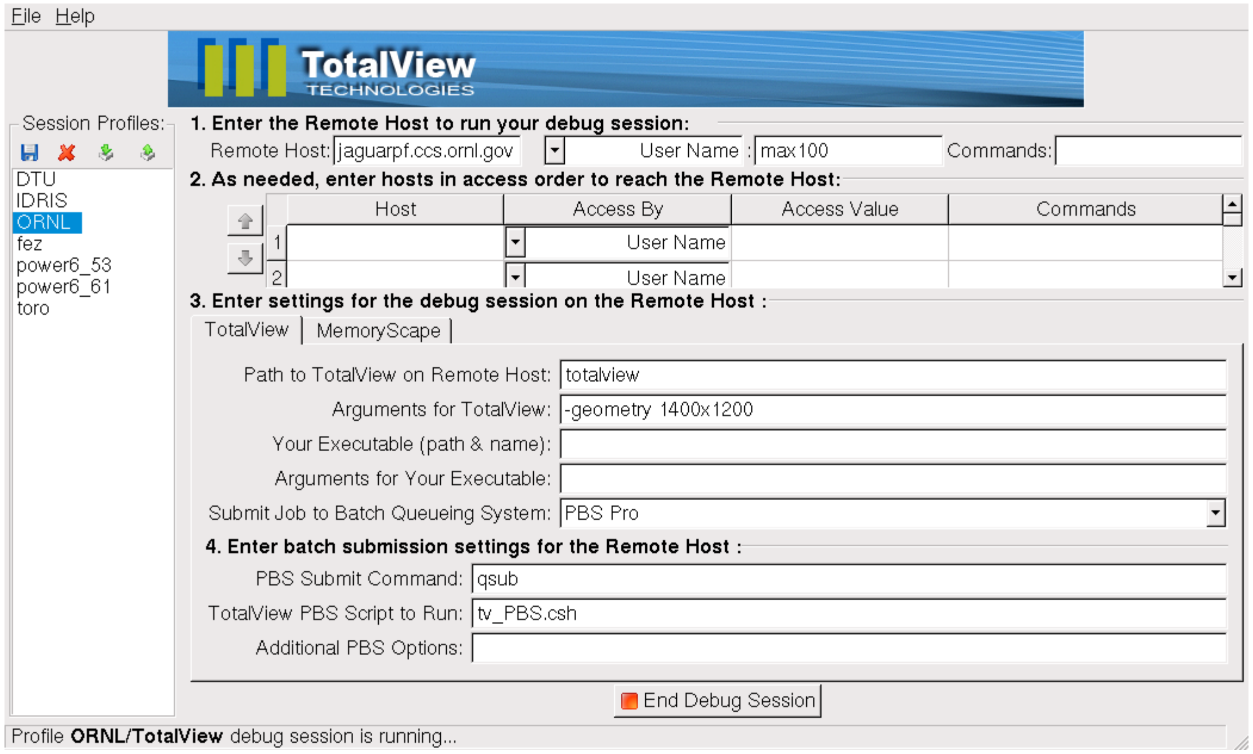
How can TotalView help you?

Debugging means examining a specific controlled instance of program execution

Provides an answer to the question : “What is my program *really* doing?”

- **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 Remote Display Client



Session Profiles:

- DTU
- IDRIS
- ORNL
- fez
- power6_53
- power6_61
- toro

1. Enter the Remote Host to run your debug session:

Remote Host: jaguarpf.ccs.ornl.gov User Name : max100 Commands:

2. As needed, enter hosts in access order to reach the Remote Host:

	Host	Access By	Access Value	Commands
1		User Name		
2		User Name		

3. Enter settings for the debug session on the Remote Host :

TotalView | MemoryScape

Path to TotalView on Remote Host:	totalview
Arguments for TotalView:	-geometry 1400x1200
Your Executable (path & name):	
Arguments for Your Executable:	
Submit Job to Batch Queueing System:	PBS Pro

4. Enter batch submission settings for the Remote Host :

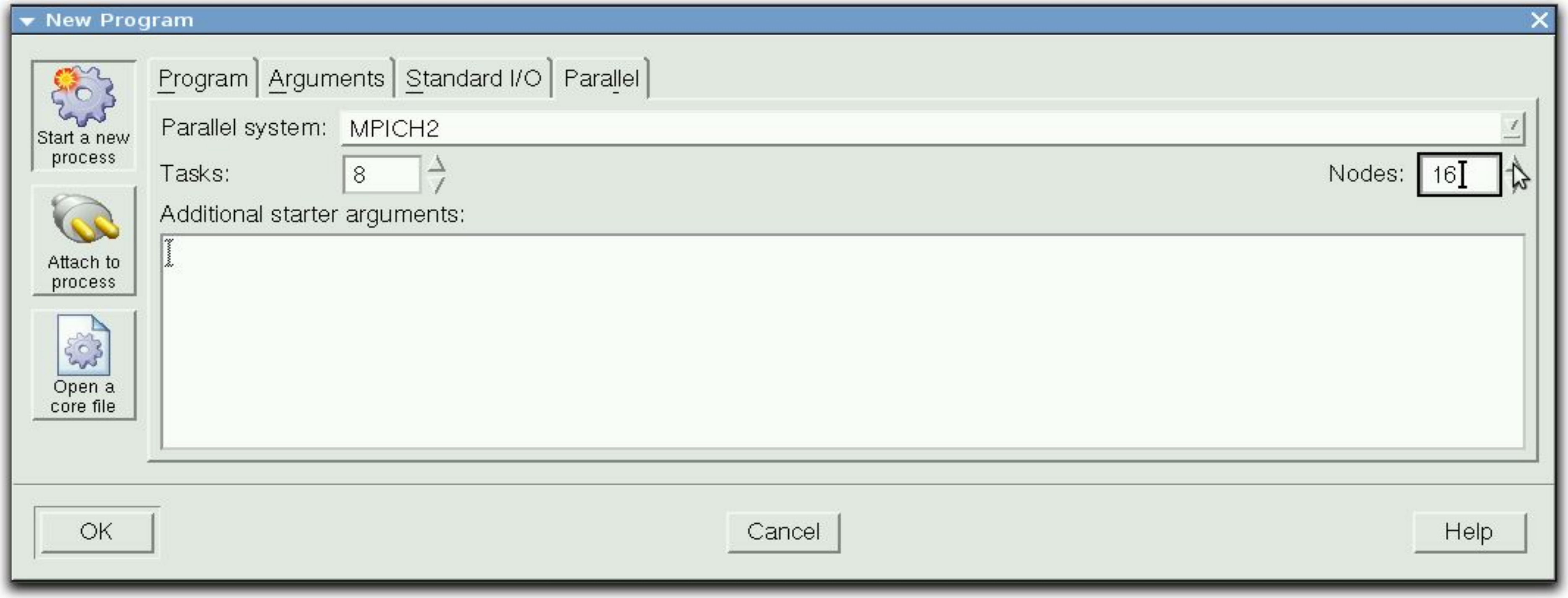
PBS Submit Command:	qsub
TotalView PBS Script to Run:	tv_PBS.csh
Additional PBS Options:	

Profile **ORNL/TotalView** debug session is running...

- **The Remote Display Client offers users the ability to easily set up and operate a TotalView debug session that is running on another system.**
- **Provides for a connection that is**
  - Easy
  - Fast
  - Secure
- **The Remote Display Client is available for:**
  - Linux x86
  - Linux x86-64
  - Windows XP
  - Windows Vista
  - Mac OS X Leopard and Snow Leopard
- **The Client also provides for submission of jobs to batch queuing systems PBS Pro and LoadLeveler**

MPI in TotalView with Indirect Launch

In the Parallel tab, select:



Parallel system: MPICH2
Tasks: 8
Nodes: 16
Additional starter arguments:

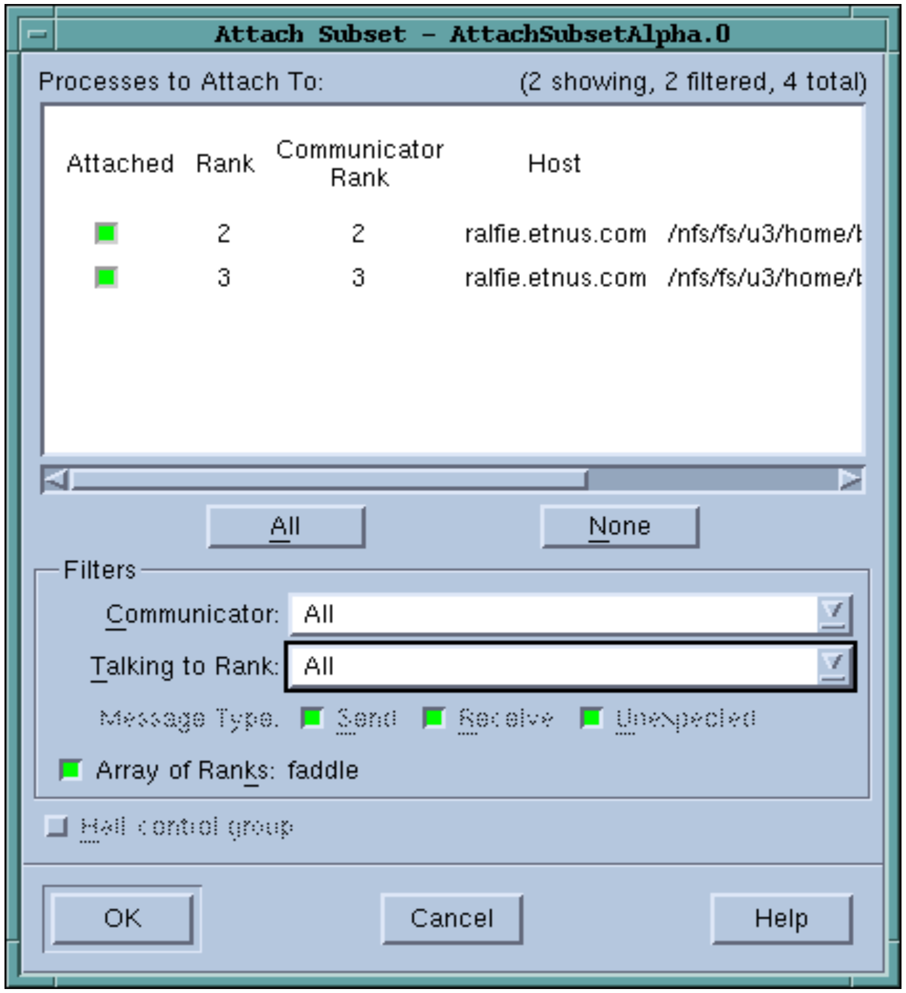
your MPI preference, number of tasks, and number of nodes.

7 … then add any additional starter arguments

Subset Attach

TotalView does not need to be attached to the entire job

- You can be attached to different subsets at different times through the run
- You can attach to a subset, run till you see trouble and then 'fan out' to look at more processes if necessary.
- This greatly reduces overhead
- It also requires a smaller license if you have a TotalView Team license.



Attach Subset - AttachSubsetAlpha.0

Processes to Attach To: (2 showing, 2 filtered, 4 total)

Attached	Rank	Communicator Rank	Host	
🟩	2	2	ralfie.etnus.com	/nfs/fs/u3/home/t
🟩	3	3	ralfie.etnus.com	/nfs/fs/u3/home/t

Filters

Communicator: All

Talking to Rank: All

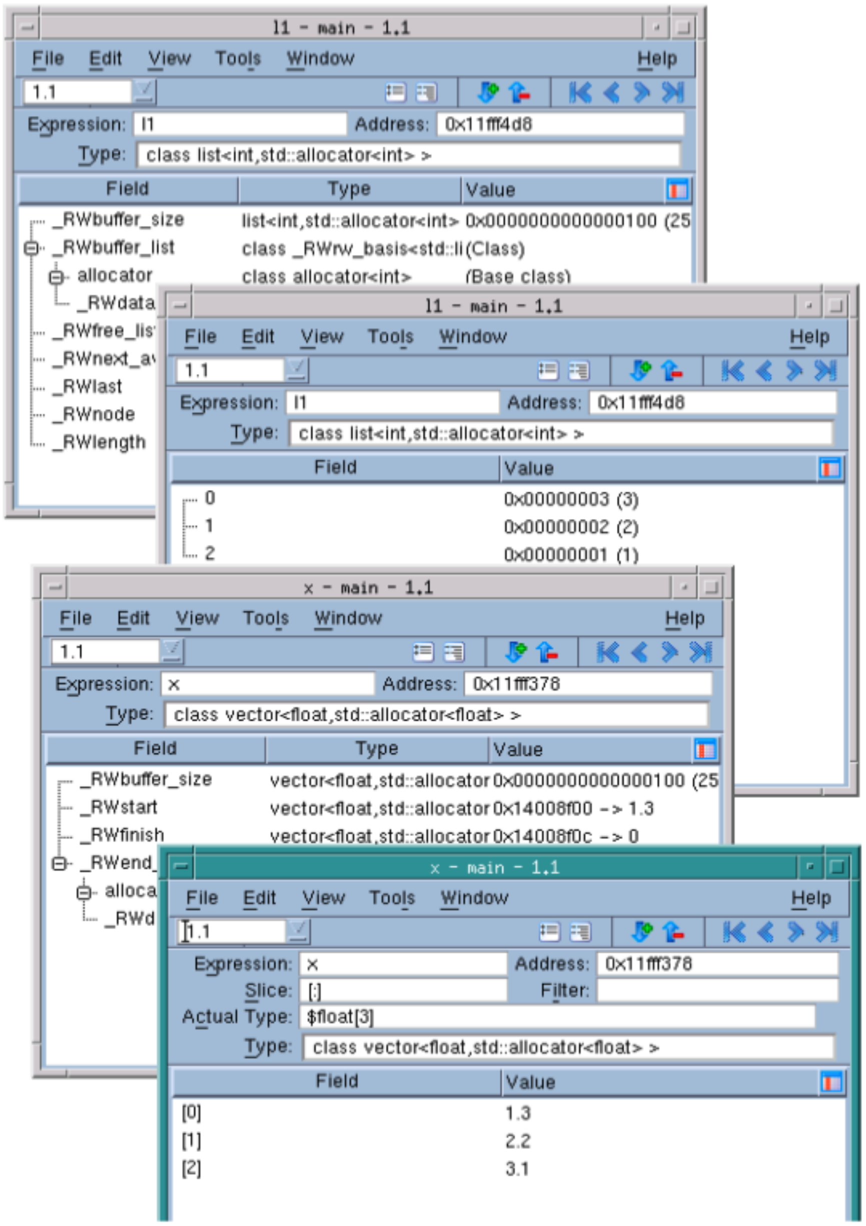
Message Type: 🟩 Send 🟩 Receive 🟩 Unexpected

🟩 Array of Ranks: faddle

☐ Halt control group

STLView

STLView transforms templates into readable and understandable information



l1 - main - 1.1

Expression: l1 Address: 0x11fff4d8

Type: class list<int,std::allocator<int> >

Field	Type	Value
_RWbuffer_size	list<int,std::allocator<int>	0x0000000000000100 (25
_RWbuffer_list	class _RWrw_basis<std::li	(Class)
allocator	class allocator<int>	(Base class)
_RWdata		
_RWfree_lis		
_RWnext_a		
_RWlast		
_RWnode		
_RWlength		

Untransformed list

l1 - main - 1.1

Expression: l1 Address: 0x11fff4d8

Type: class list<int,std::allocator<int> >

Field	Value
0	0x00000003 (3)
1	0x00000002 (2)
2	0x00000001 (1)

Transformed list

x - main - 1.1

Expression: x Address: 0x11fff378

Type: class vector<float,std::allocator<float> >

Field	Type	Value
_RWbuffer_size	vector<float,std::allocator	0x0000000000000100 (25
_RWstart	vector<float,std::allocator	0x14008f00 -> 1.3
_RWfinish	vector<float,std::allocator	0x14008f0c -> 0
RWend		
alloca		
_RWd		

Untransformed vector

x - main - 1.1

Expression: x Address: 0x11fff378

Slice: [:] Filter:

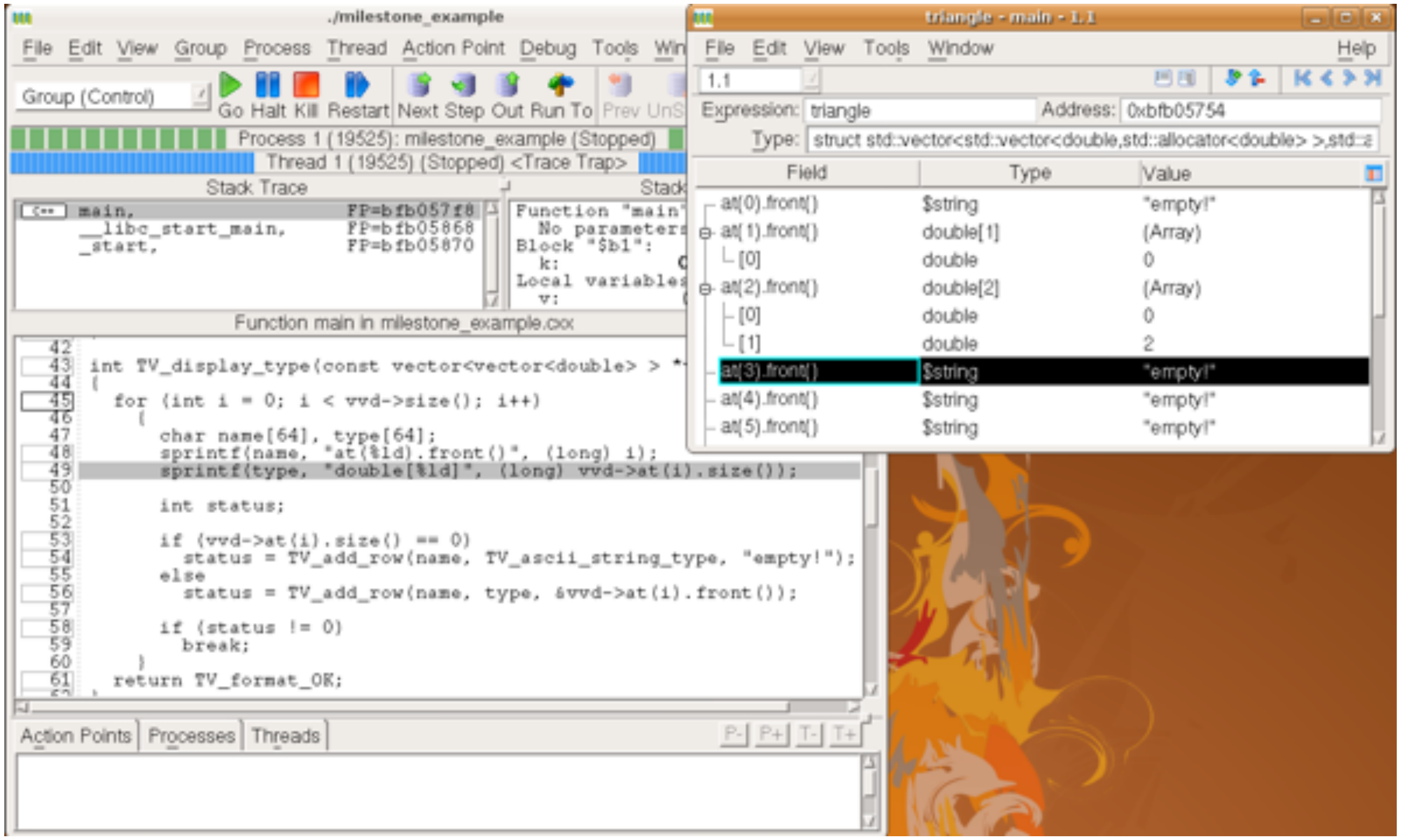
Actual Type: $float[3]

Type: class vector<float,std::allocator<float> >

Field	Value
[0]	1.3
[1]	2.2
[2]	3.1

Transformed vector

Pre-Release: C++View


./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
Process 1 (19525): milestone_example (Stopped)
Thread 1 (19525) (Stopped) <Trace Trap>
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

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

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

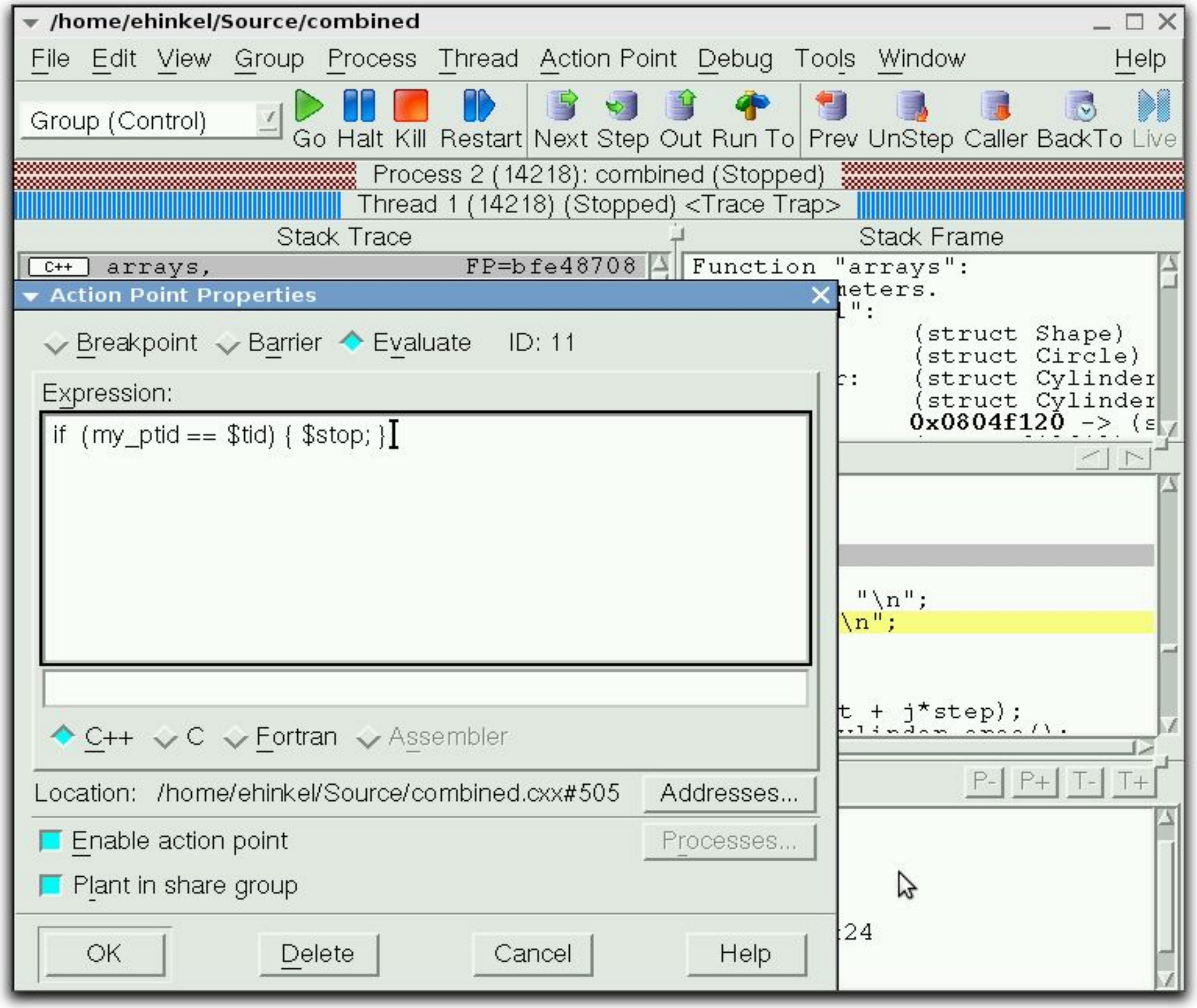
```
int TV_display_type(const vector<vector<double> > *
{
  for (int i = 0; i < vvd->size(); i++)
    {
      char name[64], type[64];
      sprintf(name, "at(%ld).front()", (long) i);
      sprintf(type, "double[%ld]", (long) vvd->at(i).size());

      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!"

http://www.totalviewtech.com/forms/cppview.html

Conditional Breakpoint


/home/ehinkel/Source/combined
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Prev UnStep Caller BackTo Live
Stack Trace
Stack Frame
P- P+ T- T+

Process 2 (14218): combined (Stopped)
Thread 1 (14218) (Stopped) <Trace Trap>

```
C++  arrays,        FP=bfe48708
Function "arrays":
```

Action Point Properties

Breakpoint   Barrier   [✓ Evaluate]   ID: 11

Expression:

```
if (my_ptid == $tid) { $stop; }
```

[✓ C++]   C   Fortran   Assembler

Location: /home/ehinkel/Source/combined.cxx#505

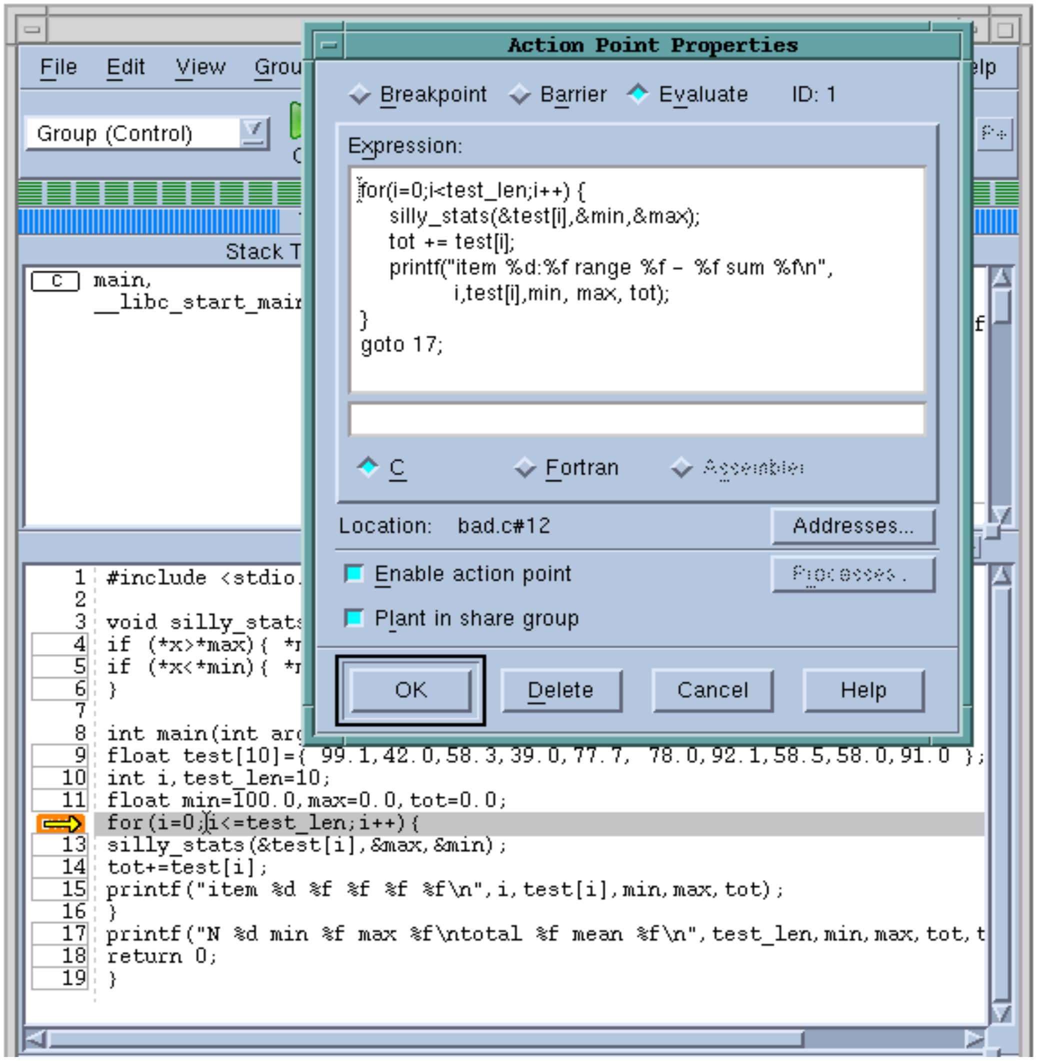
[✓] Enable action point

[✓] Plant in share group

OK   Delete   Cancel   Help   Addresses...   Processes...

Evaluation Breakpoint… Test Fixes on the Fly!

- Test small source code patches
- Call functions
- Set variables
- Test conditions
- C/C++ or Fortran
- Can't use C++ constructors
- Use program variables
- Can't modify variables or call functions with replay engine



```
for(i=0;i<test_len;i++) {
    silly_stats(&test[i],&min,&max);
    tot += test[i];
    printf("item %d:%f range %f - %f sum %f\n",
           i,test[i],min, max, tot);
}
goto 17;
```

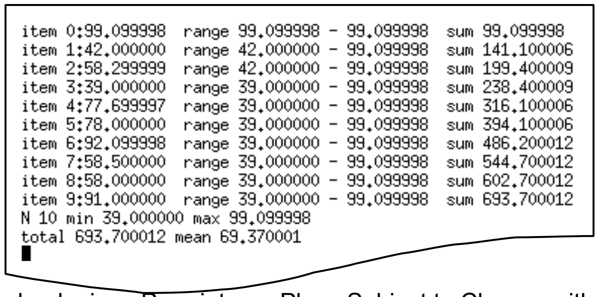
Location: bad.c#12

Enable action point [✓]

Plant in share group [✓]

```
main,
__libc_start_mair
```

```
 1  #include <stdio.
 2
 3  void silly_stats
 4  if (*x>*max){ *r
 5  if (*x<*min){ *r
 6  }
 7
 8  int main(int arc
 9  float test[10]={ 99.1,42.0,58.3,39.0,77.7, 78.0,92.1,58.5,58.0,91.0 };
10  int i,test_len=10;
11  float min=100.0,max=0.0,tot=0.0;
    for(i=0;i<=test_len;i++){
13  silly_stats(&test[i],&max,&min);
14  tot+=test[i];
15  printf("item %d %f %f %f %f\n",i,test[i],min,max,tot);
16  }
17  printf("N %d min %f max %f\ntotal %f mean %f\n",test_len,min,max,tot,t
18  return 0;
19  }
```



```
item 0:99.099998  range 99.099998 - 99.099998  sum 99.099998
item 1:42.000000  range 42.000000 - 99.099998  sum 141.100006
item 2:58.299999  range 42.000000 - 99.099998  sum 199.400009
item 3:39.000000  range 39.000000 - 99.099998  sum 238.400009
item 4:77.699997  range 39.000000 - 99.099998  sum 316.100006
item 5:78.000000  range 39.000000 - 99.099998  sum 394.100006
item 6:92.099998  range 39.000000 - 99.099998  sum 486.200012
item 7:58.500000  range 39.000000 - 99.099998  sum 544.700012
item 8:58.000000  range 39.000000 - 99.099998  sum 602.700012
item 9:91.000000  range 39.000000 - 99.099998  sum 693.700012
N 10 min 39.000000 max 99.099998
total 693.700012 mean 69.370001
█
```

Batch Debugging with TVScript

- **TVScript**
  - Defines events
    - Breakpoints, memory errors, etc..
  - Actions to take in response to these events
    - Print variables or create memory reports
  - Runs a serial or MPI program towards 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

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 blocks sample with tvscript:

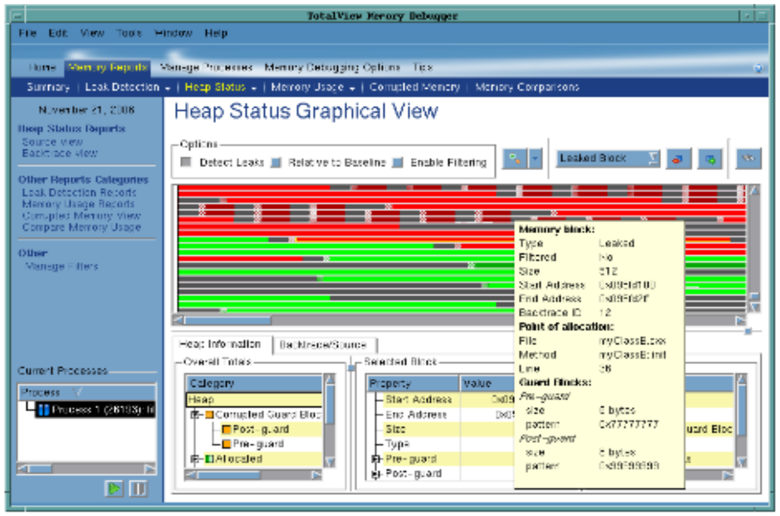
```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Print
!
! Process:
!   ./server (Debugger Process ID:  1, System ID:  12110)
! Thread:
!   Debugger ID:  1.1, System ID:  3083946656
! Time Stamp:
!   06-26-2008 14:04:09
! Triggered from event:
!   actionpoint
! Results: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

tvscript

- tvscript lets you define what events to act on, and what actions to take
- 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*}
- tvscript also supports external script files, utilizing TCL within a CLI file allowing the generation of even more complex actions to events

What is MemoryScape?

Simple to use, intuitive memory debugging

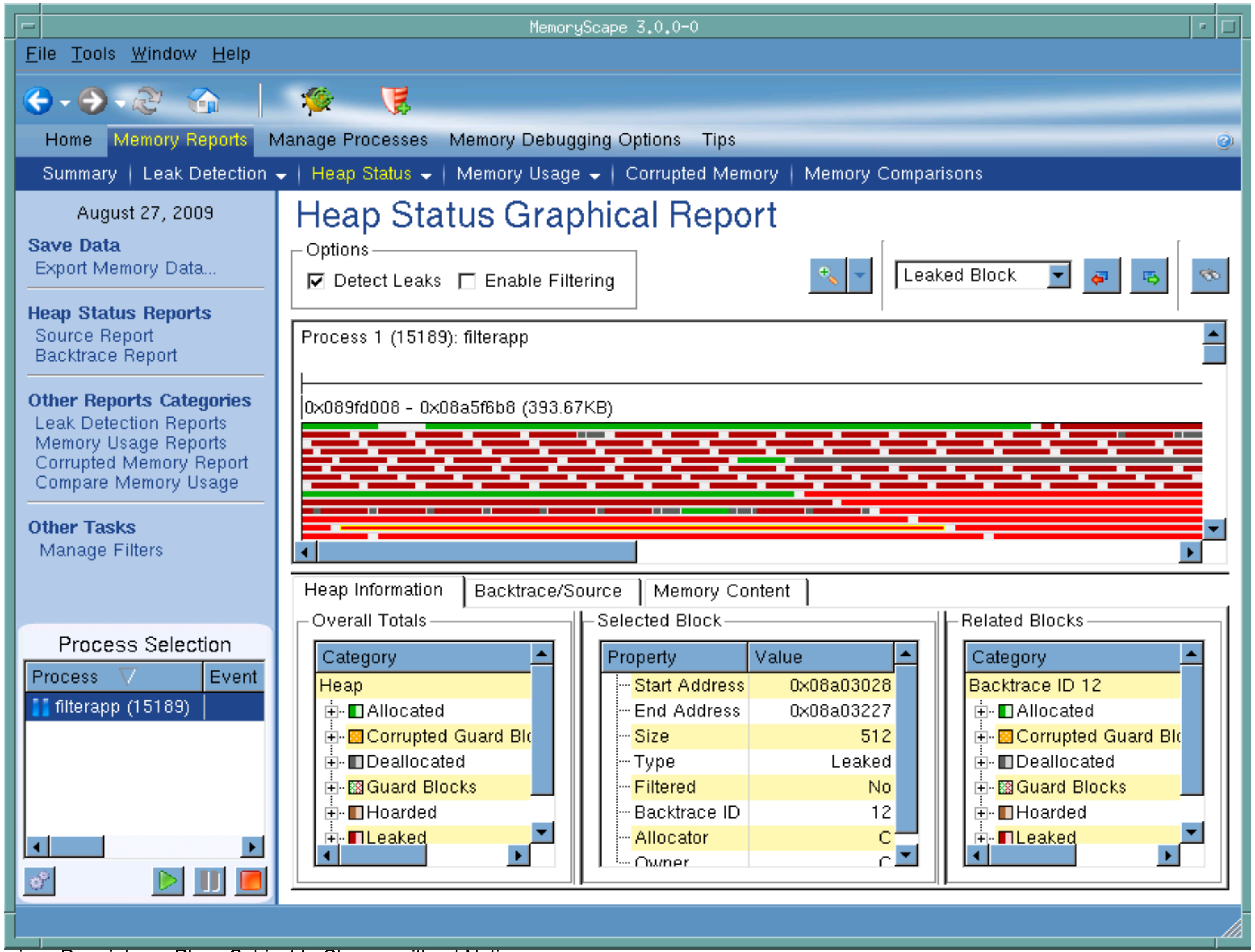


Memory block:
Type Leaked
Filtered No
Size 512
Start Address 0x09e1d100
End Address 0x09e1d2ff
Backtrace ID 12
Point of allocation:
File myClassB.cxx
Method myClassB::init
Line 36
Guard Blocks:
Pre-guard
size 8 bytes
pattern 0x77777777
Post-guard
size 8 bytes
pattern 0x99999999

- **What is MemoryScape?**
  - Streamlined
  - Lightweight
  - Intuitive
  - Collaborative
  - Memory Debugging
- **Features**
  - Shows
    - Memory errors
    - Memory status
    - Memory leaks
    - Buffer overflows
  - MPI memory debugging
  - Remote memory debugging

- **Technical Advantages**
  - Low overhead
  - No Instrumentation
- **Interface**
  - Inductive
  - Collaboration
  - Multi-process

Heap Graphical View



Heap Status Graphical Report

Options

[✓] Detect Leaks [ ] Enable Filtering

Leaked Block

Process 1 (15189): filterapp

0x089fd008 - 0x08a5f6b8 (393.67KB)

Process Selection

Process	Event
filterapp (15189)	

Overall Totals

Category
Heap
Allocated
Corrupted Guard Blo
Deallocated
Guard Blocks
Hoarded
Leaked

Selected Block

Property	Value
Start Address	0x08a03028
End Address	0x08a03227
Size	512
Type	Leaked
Filtered	No
Backtrace ID	12
Allocator	C
Owner	C

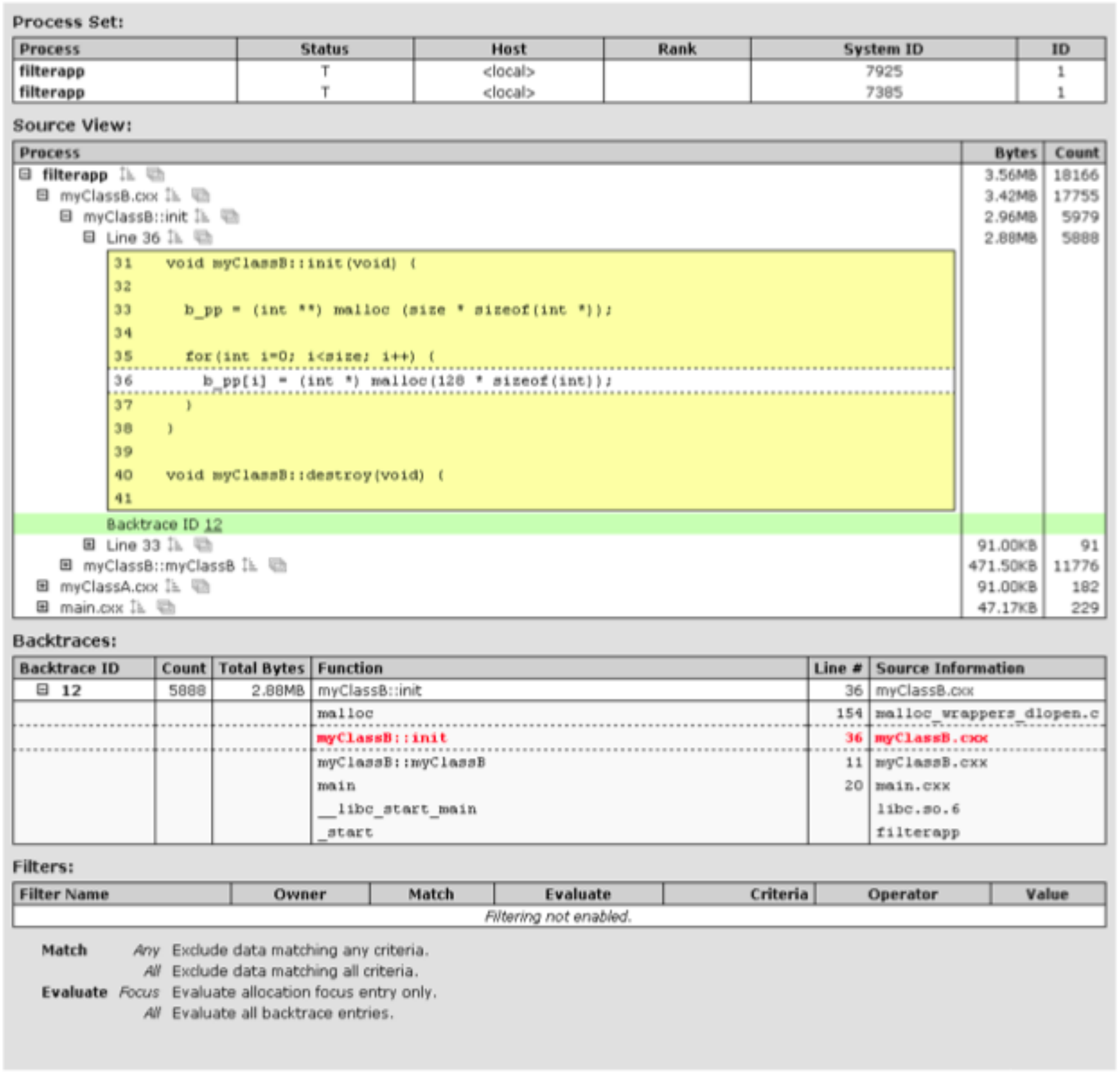
Related Blocks

Category
Backtrace ID 12
Allocated
Corrupted Guard Blo
Deallocated
Guard Blocks
Hoarded
Leaked

MemoryScape Reporting

- **Allocations and Leaks**
- **Filtered**
- **HTML**
  - Collaboration
- **Text**
  - Scripting

- **Heap Memory File**
  - Export/Reload
  - Diff-style comparison



Process Set:

Process	Status	Host	Rank	System ID	ID
filterapp	T	<local>		7925	1
filterapp	T	<local>		7385	1

Source View:

Process	Bytes	Count
⊟ filterapp	3.56MB	18166
⊟ myClassB.cxx	3.42MB	17755
⊟ myClassB::init	2.96MB	5979
⊟ Line 36	2.88MB	5888

```
31    void myClassB::init(void) {
32
33      b_pp = (int **) malloc (size * sizeof(int *));
34
35      for(int i=0; i<size; i++) {
36        b_pp[i] = (int *) malloc(128 * sizeof(int));
37      }
38    }
39
40    void myClassB::destroy(void) {
41
```

Process	Bytes	Count
Backtrace ID 12		
⊞ Line 33	91.00KB	91
⊞ myClassB::myClassB	471.50KB	11776
⊞ myClassA.cxx	91.00KB	182
⊞ main.cxx	47.17KB	229

Backtraces:

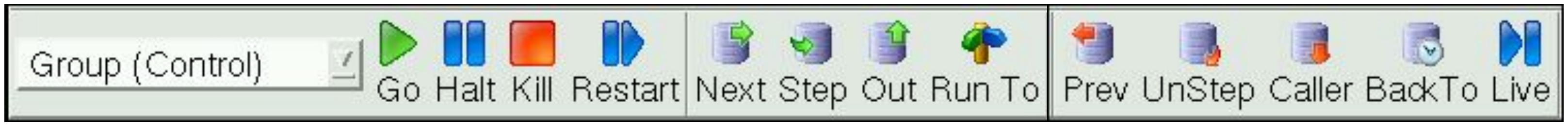
Backtrace ID	Count	Total Bytes	Function	Line #	Source Information
⊟ 12	5888	2.88MB	myClassB::init	36	myClassB.cxx
			malloc	154	malloc_wrappers_dlopen.c
			myClassB::init	36	myClassB.cxx
			myClassB::myClassB	11	myClassB.cxx
			main	20	main.cxx
			__libc_start_main		libc.so.6
			_start		filterapp

Filters:

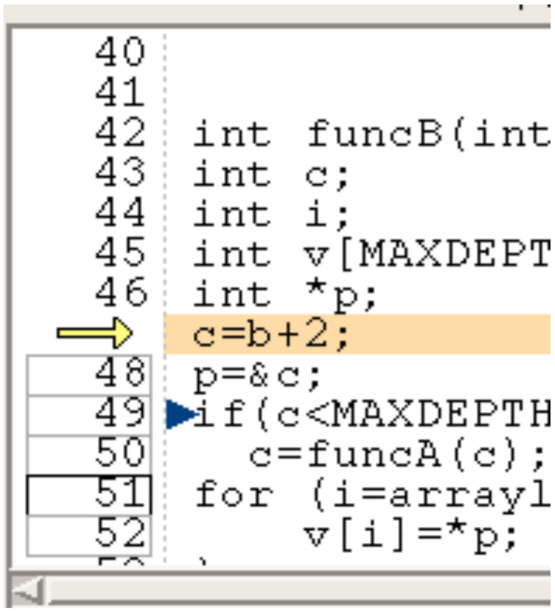
Filter Name	Owner	Match	Evaluate	Criteria	Operator	Value
Filtering not enabled.						

Match	*Any*	Exclude data matching any criteria.
	All	Exclude data matching all criteria.
Evaluate	*Focus*	Evaluate allocation focus entry only.
	All	Evaluate all backtrace entries.

What is ReplayEngine?



- **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**



```
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;
```

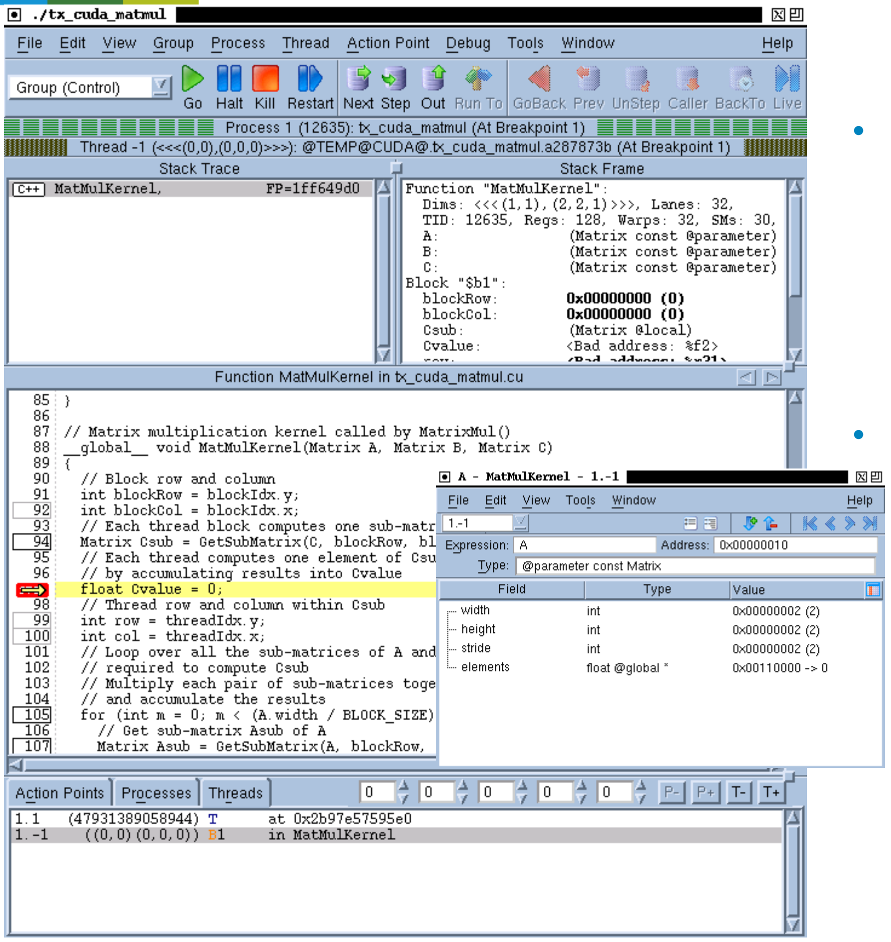
ReplayEngine Parallel Support

- **MPI**
  - ReplayEngine treats MPI communication as input.
  - The history of a single process can be explored without altering the state of any other process.
  - MPI Support
    - MPICH and MPICH 2
    - OpenMPI and LAM-MPI
    - MVAPICH and MVAPICH2
    - Intel MPI
    - HP-MPI
    - SGI MPT
- **Threads**
  - OpenMP and pthreads are both supported
  - Threads are serialized and once recorded the sequencing of threads is immutable.

ReplayEngine Recent Enhancements

- **Recording and Replaying high I/O and long running applications**
  - The user specifies a buffer size limit for recorded history.
  - ReplayEngine records all events as the program runs till this buffer gets full.
    - Lots of I/O (specifically input)
    - Long running apps
  - If the buffer fills up the oldest history is discarded, more recent history is available for replay.
- **Backwards continue command**
  - Breakpoints and watchpoints can be set and enabled at any point
  - Run back to the last time any breakpoint or watchpoint would have triggered
  - Works with expression points and expression watchpoints
- **Support for programs that make use of shared memory**
  - This can be through explicit usage of **mmap(MAP_SHARED)** or through the use of libraries that make use of shared memory
  - Certain MPIs use shared memory for low latency
    - MPICH2 nemesis
    - OpenMPI (certain drivers)
    - Intel MPI
- **Support for Cray XT running CLE**
  - Requires: TotalView 8.8 or later and TotalView Support 1.1.0

TotalView debugger for CUDA


./tx_cuda_matmul
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 Caller BackTo Live
Stack Trace
Stack Frame
Action Points
Processes
Threads
P- P+ T- T+
A - MatMulKernel - 1.-1
File Edit View Tools Window Help
1.-1

```
Process 1 (12635): tx_cuda_matmul (At Breakpoint 1)
Thread -1 (<<<(0,0),(0,0,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.a287873b (At Breakpoint 1)
```

```
C++  MatMulKernel,            FP=1ff649d0
```

```
Function "MatMulKernel":
  Dims: <<<(1,1),(2,2,1)>>>, Lanes: 32,
  TID: 12635, Regs: 128, Warps: 32, SMs: 30,
  A:                   (Matrix const @parameter)
  B:                   (Matrix const @parameter)
  C:                   (Matrix const @parameter)
Block "$b1":
  blockRow:            0x00000000 (0)
  blockCol:            0x00000000 (0)
  Csub:                (Matrix @local)
  Cvalue:              <Bad address: %f2>
```

Function MatMulKernel in tx_cuda_matmul.cu

```
}

// Matrix multiplication kernel called by MatrixMul()
__global__ void MatMulKernel(Matrix A, Matrix B, Matrix C)
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matr
  Matrix Csub = GetSubMatrix(C, blockRow, bl
  // Each thread computes one element of Csu
  // 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
  // required to compute Csub
  // Multiply each pair of sub-matrices toge
  // and accumulate the results
  for (int m = 0; m < (A.width / BLOCK_SIZE)
    // Get sub-matrix Asub of A
    Matrix Asub = GetSubMatrix(A, blockRow,
```

Expression: A   Address: 0x00000010
Type: @parameter const Matrix

Field	Type	Value
width	int	0x00000002 (2)
height	int	0x00000002 (2)
stride	int	0x00000002 (2)
elements	float @global *	0x00110000 -> 0

```
1.1    (47931389058944) T      at 0x2b97e57595e0
1.-1     ((0,0)(0,0,0)) B1     in MatMulKernel
```

- **What is TotalView?**
  - Parallel and Multi-threaded Debugging and Analysis Tool
  - For developers, scientists and engineers using C/C++
  - Makes developing, maintaining and supporting critical and cutting edge applications easier and less risky
- **Debugging CUDA**
  - Currently being extended to support **CUDA** programming on **NVIDIA Tesla** and **Fermi** cards
  - Native debugging of both the host (CPU) C or C++ code and the device (GPU) CUDA code.
  - Participate in the **Early Experience Program** to help influence the product direction
- **Other Major TotalView Features**
  - Supports Linux, Unix and Mac OS X
  - Parallel Debugging for Clusters and Many-Core
  - Memory Debugging with **MemoryScape**
  - Batch Debugging with TVScript and the CLI

Advantages

- Easy to learn graphical user interface with data visualization
- Wide variety of features so users can tackle unexpected bugs

GP-GPU Early Experience Program

- **Way for users to participate in the development of TotalView for CUDA**
  - Provide input into development efforts
  - Review and help refine usage models
    - How to group threads and provide status data without overwhelming the user
    - How to manage and control device threads
    - How to display data from 10k + threads
    - How to debug accelerated clusters using MPI and CUDA
  - Get early access to pre-release software before it is available to the public
  - NDA program
  - Sign up now
    - http://www.totalviewtech.com/
    - Contact :Chris.Gottbrath@totalviewtech.com

Questions and comments?

- **www.roguewave.com**
  - IMSL, Py-IMSL, PV-WAVE, Source Pro, and TotalView info
- **www.TotalViewTech.com**
  - Free Fully Featured Evaluation Licenses Available on the Web
  - Videos, White Papers, Product Documentation
- **TotalView Early Experience Program : GP-GPU track**
  - Sign up at **www.totalviewtech.com**
  - Email: chris.gottbrath@totalviewtech.com