



# Debugging with TotalView

Le Yan

HPC Consultant  
User Services

*LONI High Performance Computing Workshop – University of Louisiana at Lafayette  
November 2, 2010*





# Goals

- Learn how to start TotalView on Linux clusters
- Get familiar with TotalView graphic user interface
- Learn basic debugging functions of TotalView





# Outline

- Overview
- Getting Started
- Debugging with TotalView
  - TotalView GUI
  - Basic debugging techniques
  - Features for parallel programs





# Outline

- Overview
- Getting Started
- Debugging with TotalView
  - TotalView GUI
  - Basic debugging techniques
  - Features for parallel programs





# Debugging Essentials

- Reproducibility
  - Find the scenario where the error is reproducible
- Reduction
  - Reduce the problem to its essence
- Deduction
  - Form hypotheses on what the problem might be
- Experimentation
  - Filter out invalid hypotheses





# Debugging Methods

- Write/Print/Printf
- Compiler flags
- Symbolic debugger
  - GDB - GNU debugger
  - IDB - Intel debugger
- Graphic debugger
  - Totalview
  - DDT
  - Valgrind
  - Eclipse





# What A Debugger Can and Cannot Do

- What a debugger can do
  - Tell you where the program crashes
  - Help you to gain a better understanding of the context under which it crashes
- What a debugger cannot do
  - Tell you how to solve the problem
  - Detect a correctness problem
    - Validation is very important





# What Is TotalView

- A very powerful debugger
  - Can be used to debug both serial and parallel programs (especially good for parallel programs)
  - Supports multiple languages
  - Supported on most architecture/platforms
  - Both graphic and command line interfaces
  - Numerous features
    - Common functionalities such as controlled execution and breakpoints
    - Array visualization
    - Memory debugging
    - ...





# Outline

- Overview
- Getting Started
- Debugging with TotalView
  - TotalView GUI
  - Basic debugging techniques
  - Features for parallel programs





# Availability

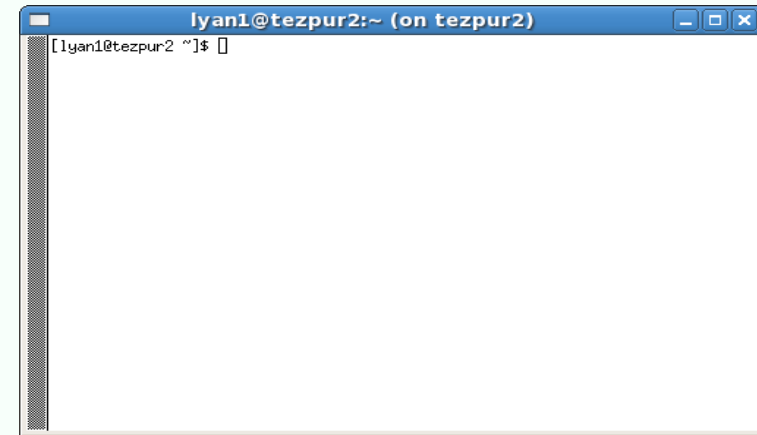
- TotalView is only available on Queen Bee and Eric for LONI users (due to license restriction)
  - Queen Bee and Eric
    - Key: `+totalview-8.3.0.1`
- Add the key to your `.soft` file and `resoft`





# Getting X Window To Work

- Linux
  - `ssh -X -Y username@hostname`
- Mac Os X
  - Use X11
- Windows
  - Install Xming and Putty
  - Enable X11 Forwarding in Putty



[https://docs.loni.org/wiki/X11\\_Forwarding](https://docs.loni.org/wiki/X11_Forwarding)





# Compile Your Program

- Serial program
  - Compile with no optimization and debugging flag turned on:
    - `icc -g -O0 -o myexec myprogram.c`
- Parallel program
  - Use the proper MPI implementation
    - Queen Bee: `+mvapich-1.0-intel10.1-tvdbg`
  - Compile with no optimization and debugging flag turned on:
    - `mpicc -g -O0 -o myexec myprogram.c`





# Getting An Interactive Session

- We need to get an interactive debugging session
  - `qsub -I -X -V -l walltime=hh:mm:ss,nodes=1:ppn=x -A <alloc> -q checkpt`  
  
...  
PBS has allocated the following nodes:  
tezpur333  
tezpur331  
...
  - “-X” for X window tunneling
- Run TotalView in the interactive session





# Start Totalview

- Serial program

- `totalview <options> <executable>`
  - **Example:** `totalview myexec`

- Parallel program

- `mpirun_rsh -tv -np <num_procs> <host list>`  
`<executable>`
  - **Example:** `mpirun_rsh -tv -np 2 tezp333 tezp333 myexec`
- `mpirun_rsh -tv -np <num_procs> -hostfile`  
`<path_to_hostfile> <executable>`
  - **Example:** `mpirun_rsh -tv -np 8 -hostfile $PBS_NODEFILE myexec`





# Outline

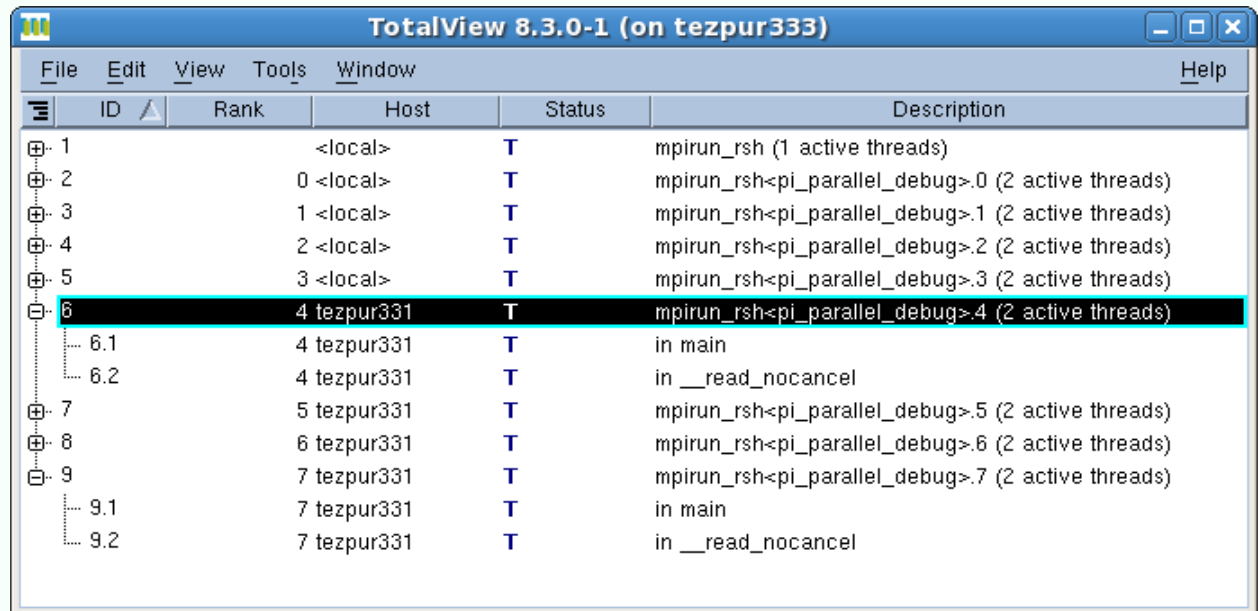
- Overview
- Getting Started
- Debugging with TotalView
  - TotalView GUI
  - Basic debugging techniques
  - Features for parallel programs





# TotalView GUI – Root Window

- Always appears when TotalView is started
- Provides an overview of all processes and threads



ID	Rank	Host	Status	Description
1	<local>	<local>	T	mpirun_rsh (1 active threads)
2	0 <local>	<local>	T	mpirun_rsh<pi_parallel_debug>.0 (2 active threads)
3	1 <local>	<local>	T	mpirun_rsh<pi_parallel_debug>.1 (2 active threads)
4	2 <local>	<local>	T	mpirun_rsh<pi_parallel_debug>.2 (2 active threads)
5	3 <local>	<local>	T	mpirun_rsh<pi_parallel_debug>.3 (2 active threads)
6	4 tezpur331	tezpur331	T	mpirun_rsh<pi_parallel_debug>.4 (2 active threads)
6.1	4 tezpur331	tezpur331	T	in main
6.2	4 tezpur331	tezpur331	T	in __read_nocancel
7	5 tezpur331	tezpur331	T	mpirun_rsh<pi_parallel_debug>.5 (2 active threads)
8	6 tezpur331	tezpur331	T	mpirun_rsh<pi_parallel_debug>.6 (2 active threads)
9	7 tezpur331	tezpur331	T	mpirun_rsh<pi_parallel_debug>.7 (2 active threads)
9.1	7 tezpur331	tezpur331	T	in main
9.2	7 tezpur331	tezpur331	T	in __read_nocancel





# TotalView GUI – Root Window

Host name

Status

Status Code	Description
Blank	Exited
B	At Breakpoint
E	Error
H	Held
K	In kernel
M	Mixed
R	Running
T	Stopped
W	At Watchpoint

ID	Rank	Host	Status	Description
1	<local>		T	mpirun_rsh (1 active threads)
2	0 <local>		T	mpirun_rsh<pi_parallel_debug>.0 (2 active threads)
3	1 <local>		T	mpirun_rsh<pi_parallel_debug>.1 (2 active threads)
4	2 <local>		T	mpirun_rsh<pi_parallel_debug>.2 (2 active threads)
5	3 <local>		T	mpirun_rsh<pi_parallel_debug>.3 (2 active threads)
6	4 tezp331		T	mpirun_rsh<pi_parallel_debug>.4 (2 active threads)
6.1	4 tezp331		T	in main
6.2	4 tezp331		T	in __read_nocancel
7	5 tezp331		T	mpirun_rsh<pi_parallel_debug>.5 (2 active threads)
8	6 tezp331		T	mpirun_rsh<pi_parallel_debug>.6 (2 active threads)
9	7 tezp331		T	mpirun_rsh<pi_parallel_debug>.7 (2 active threads)
9.1	7 tezp331		T	in main
9.2	7 tezp331		T	in __read_nocancel

TotalView ID

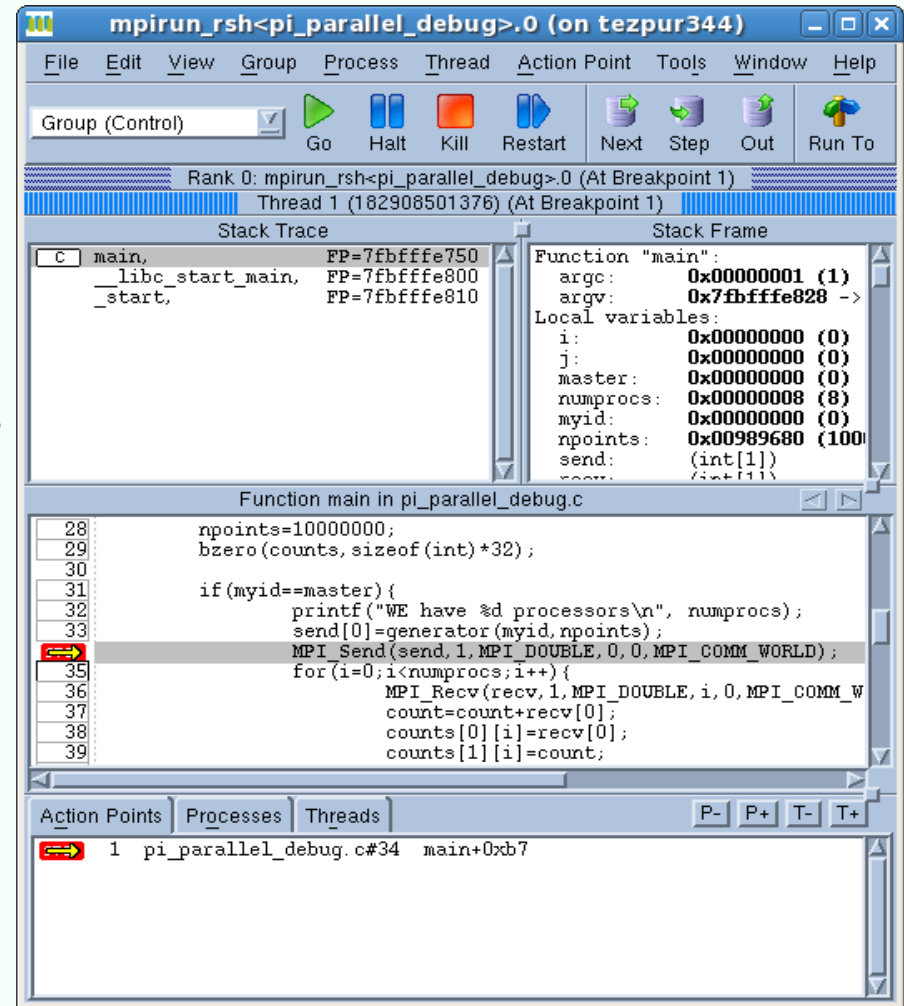
MPI Rank





# TotalView GUI – Process Window

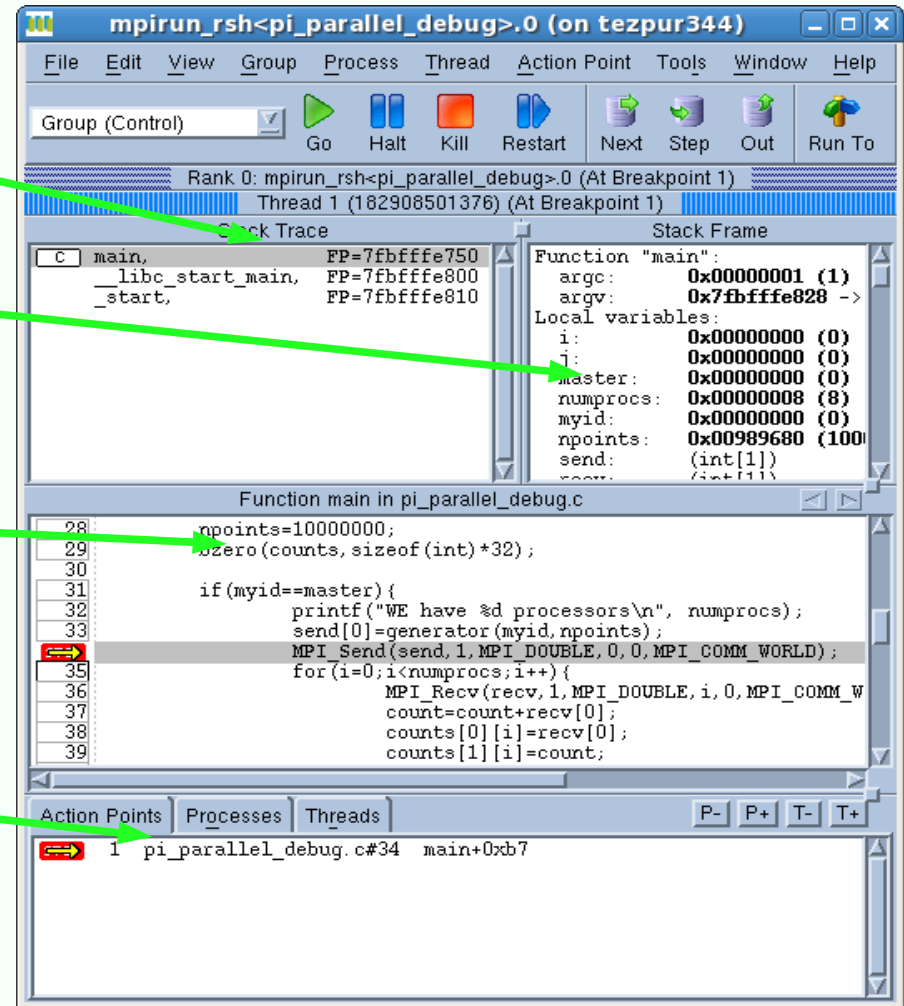
- Appears when TotalView is started (if a program is specified)
- For parallel programs each process/thread may have its own process window





# TotalView GUI – Process Window

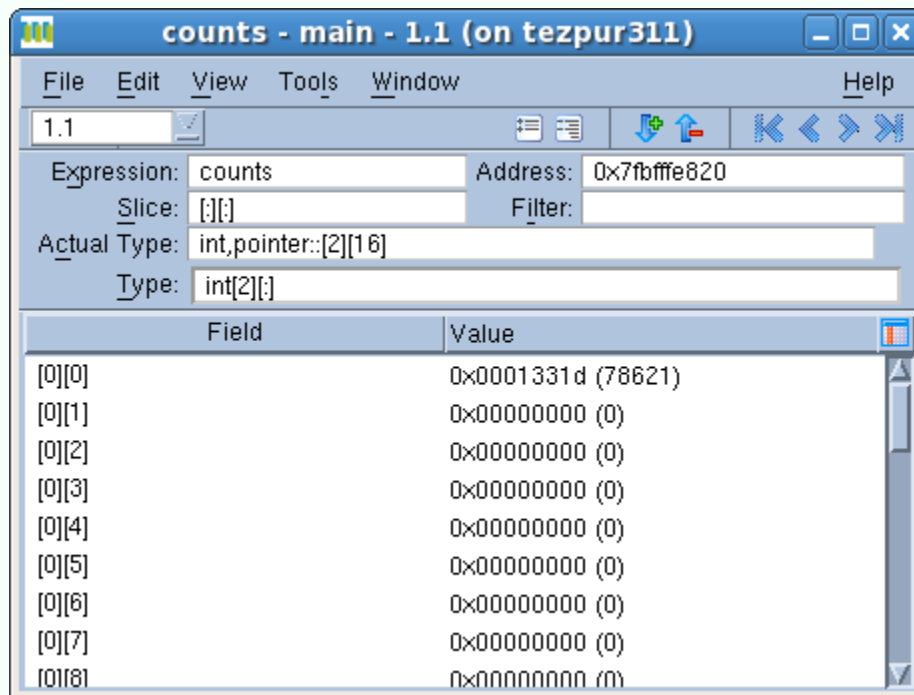
- Stack trace pane
  - Call stack of routines
- Stack frame pane
  - Local variables, registers and function parameters
- Source pane
  - Source code
- Action points, processes, threads pane





# TotalView GUI – Variable Window

- Can be opened by double-clicking on a variable name (dive)
- Display detailed information of a variable
- One can also edit the data here





# Outline

- Overview
- Getting Started
- Debugging with TotalView
  - TotalView GUI
  - Basic debugging techniques
  - Features for parallel programs





# Working With TotalView

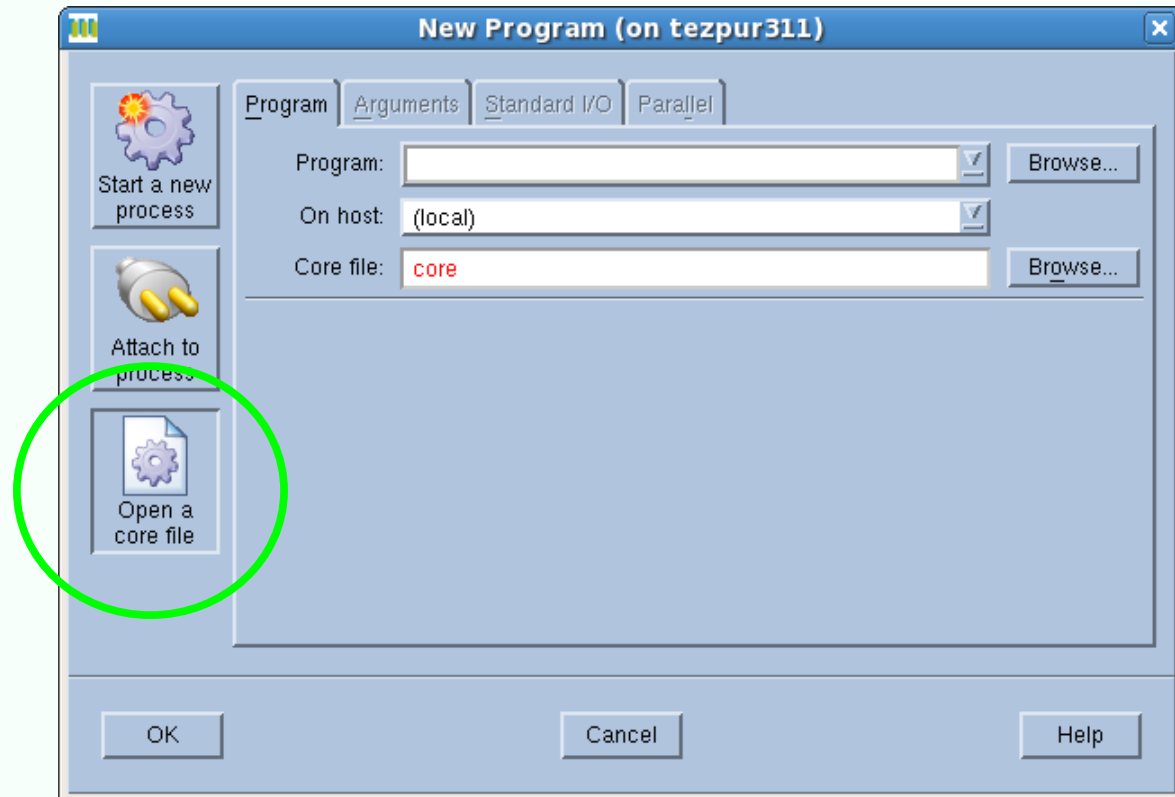
- One can start debugging with Totalview by
  - Starting TotalView with the executable
  - Debugging a core file
  - Attaching to a running (or hung) process
- We usually debug a code by
  - Setting up action points
  - Controlling the execution
  - Examining the value of variables
  - Editing the source code
  - ...





# Debugging A Core File

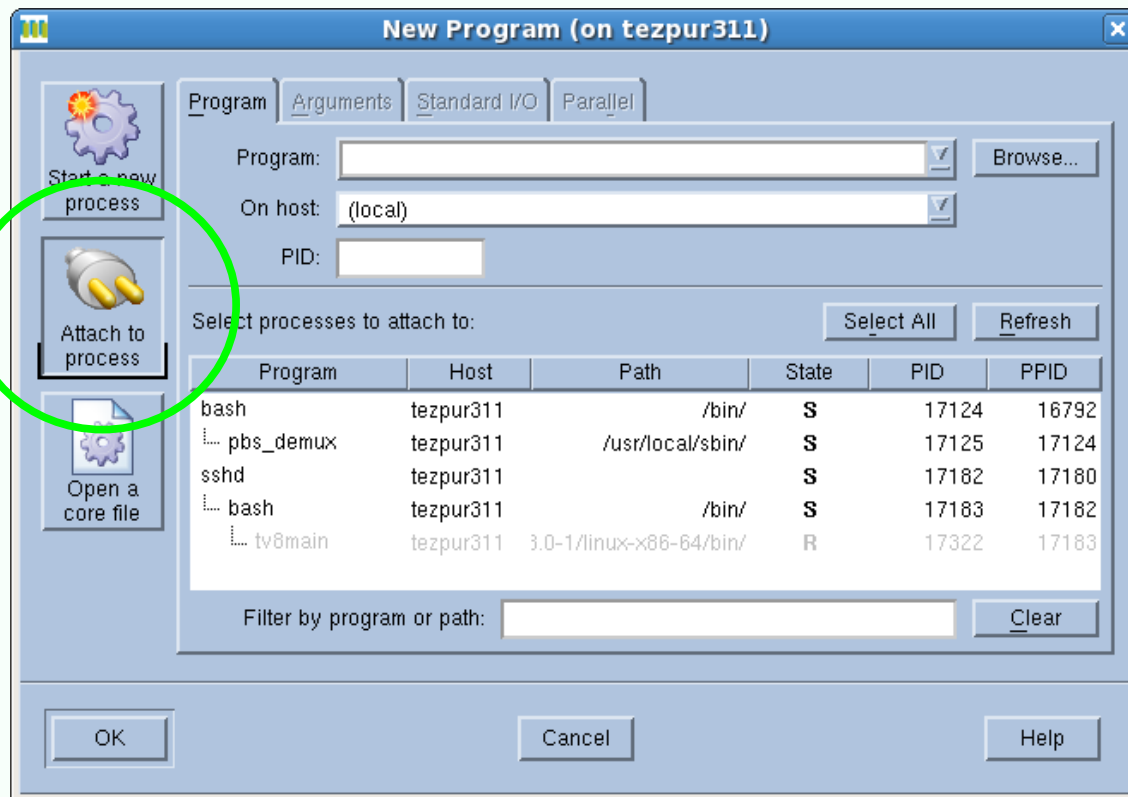
- Start TotalView without specifying the name of the program
- Open the program and core file through the 'New Program' window





# Attaching To A Running Process

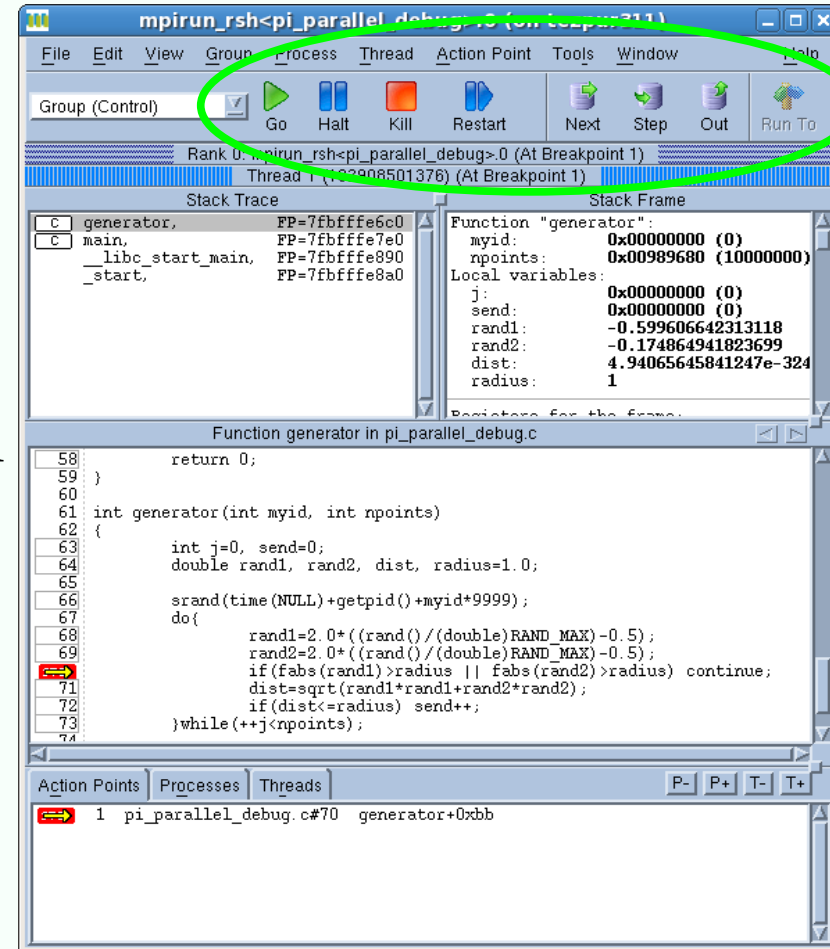
- Again, start TotalView without specifying the name of the program
- Attach to a running process through the 'New Program' window





# Controlling Execution

- Control commands
  - Go – start/resume execution
  - Halt – stop execution
  - Kill – terminate the job
  - Restart – Restarts a running program
  - Next – run to next source line without stepping in to another subroutine/function
  - Step – run to next source line
  - Out – run to the completion of a function
  - ...





# Action Points in TotalView

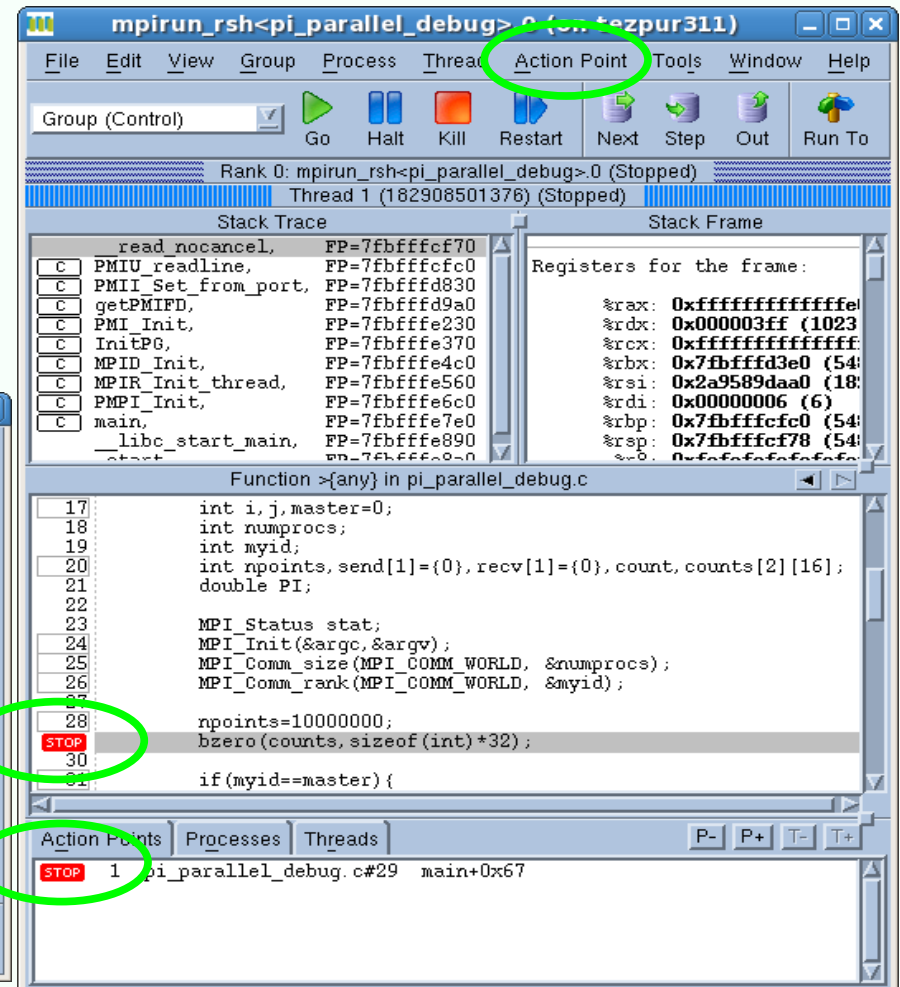
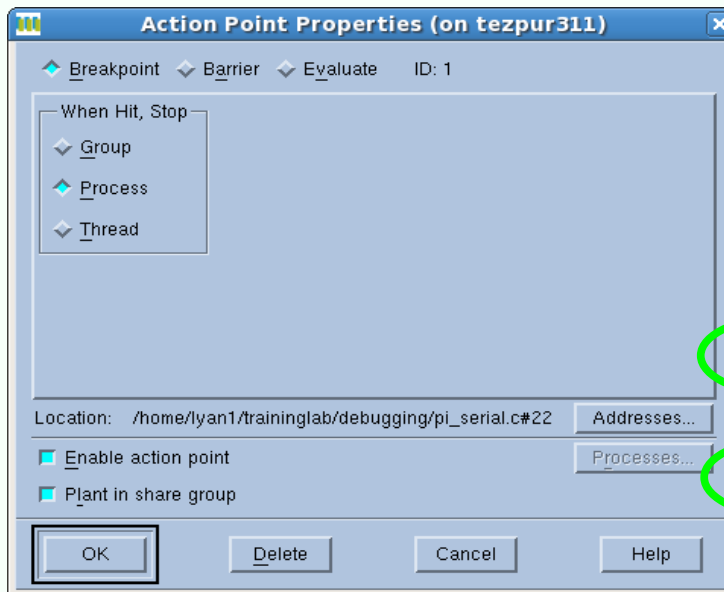
- **Breakpoints** stop the execution of the processes and threads that reach it
  - Can be conditional or unconditional
- **Process barrier points** synchronize a set of processes or threads
- **Evaluation points** cause a code fragment to be executed when reached
- **Watchpoints** let the programmer monitor a location in memory, and stop execution or evaluate an expression when its value changes





# Breakpoints

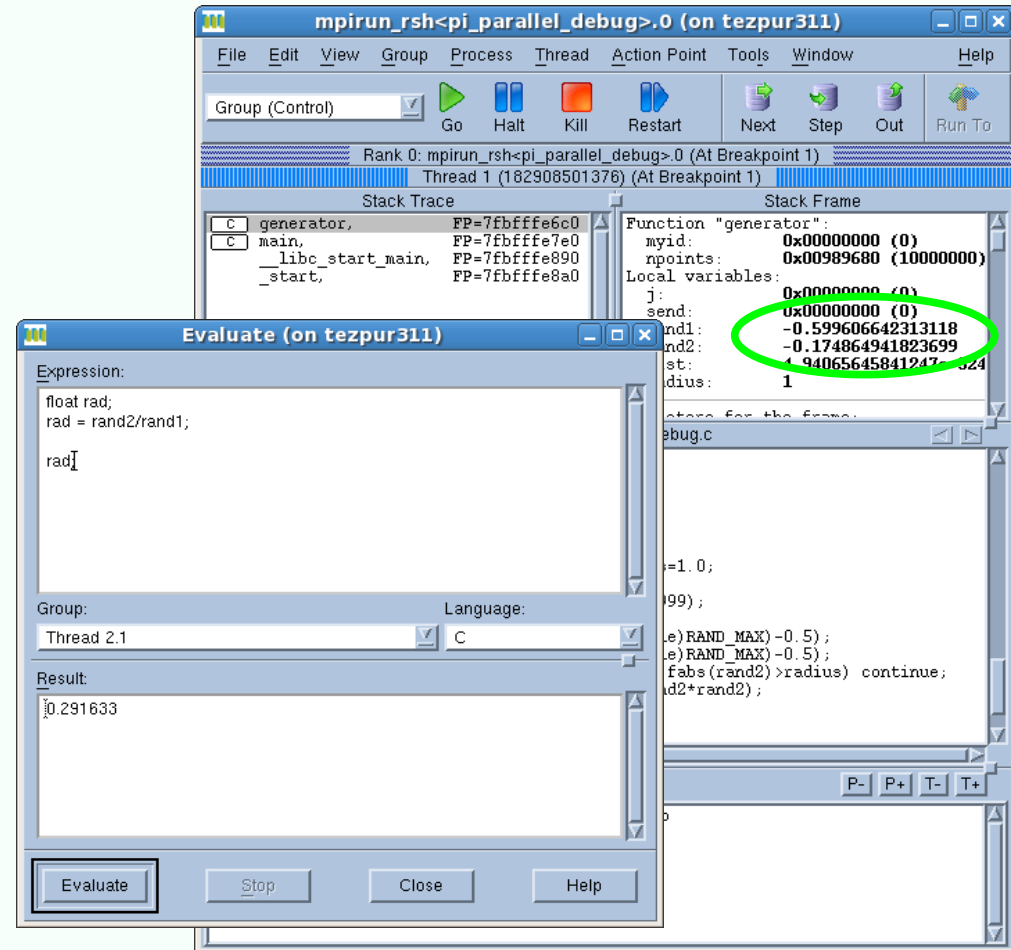
- The most basic action point
- To add a breakpoint
  - Click on the line number
  - Right click on a source line -> Set breakpoint





# Evaluation Points

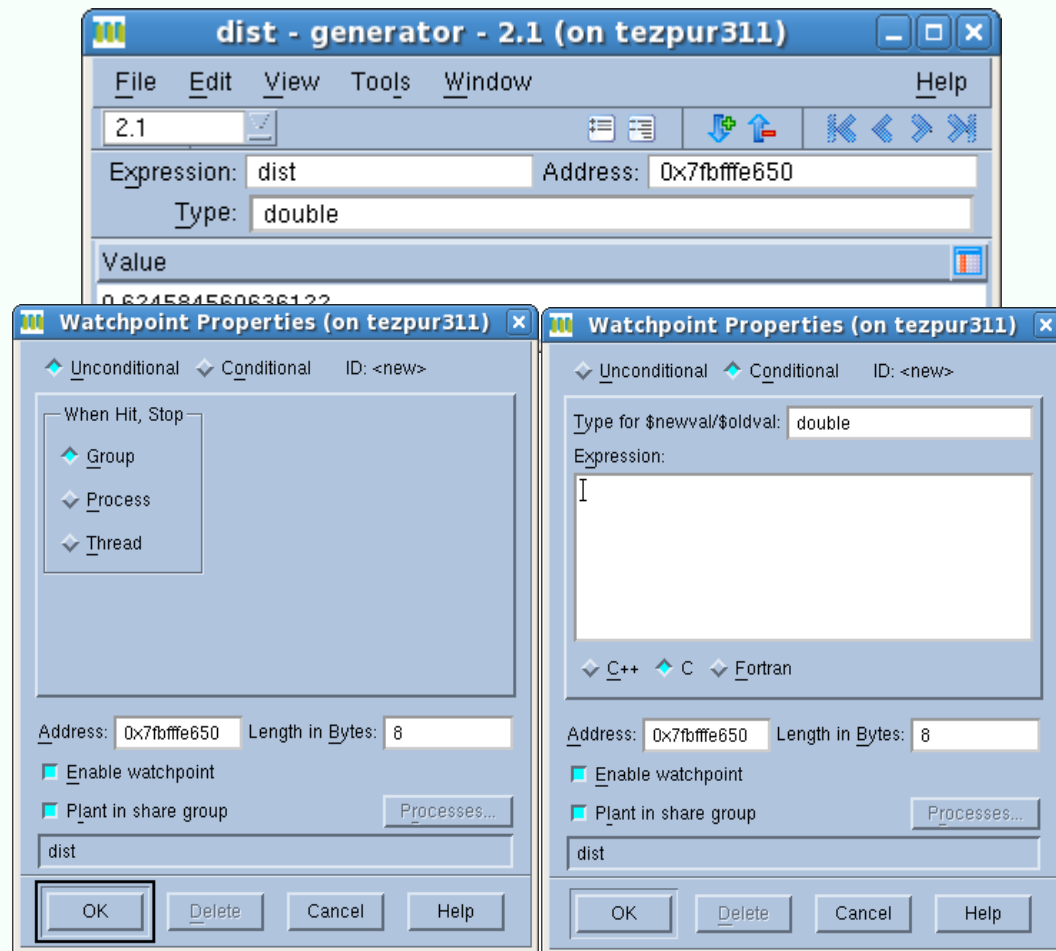
- Stop and execute a code fragments when reached
  - Useful when testing small patches
- To add a code fragment
  - Tools > Evaluate





# Watchpoints

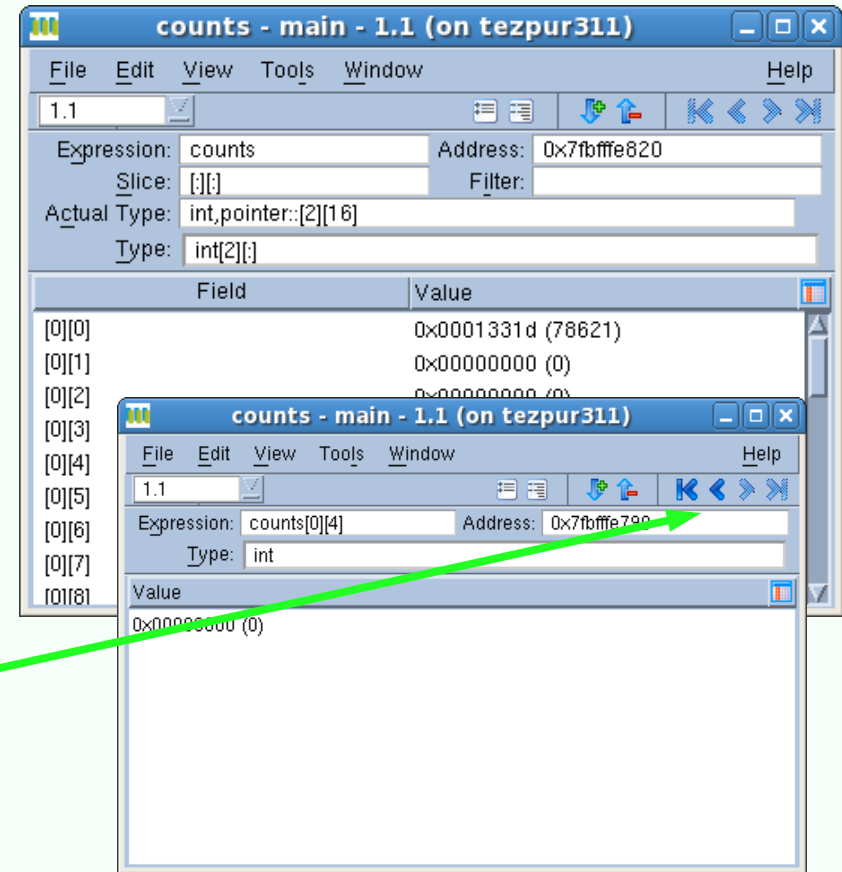
- Two types of watchpoints
  - Unconditional
  - Conditional
    - Example: stop the execution after 50k iterations
- To add a watchpoint
  - Right click on a variable -> Create watchpoint





## Diving On An Object

- “Diving” means “showing more details on an object”
- By double clicking, one can dive on
  - Variables
  - Processes/threads
  - Subroutines
  - ...
- Use the 'undive' button to go back





# Viewing/Editing Data

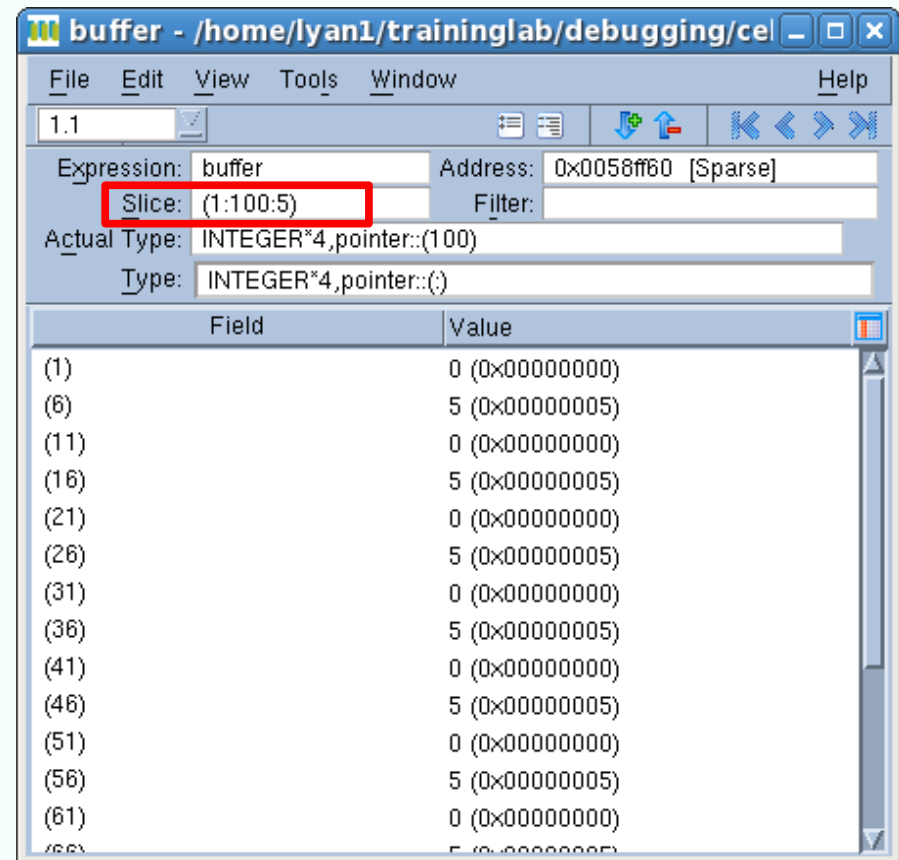
- View values and types of variables
  - By hovering mouse over the variable
  - In stack frame
  - In variable window
- Edit variable value and type
  - In stack frame
  - In variable window





# Handling Arrays (1)

- Slicing
  - Display array subsection by editing the **slice** field in the variable window
  - Form
    - [upper bound:lower bound:stride]

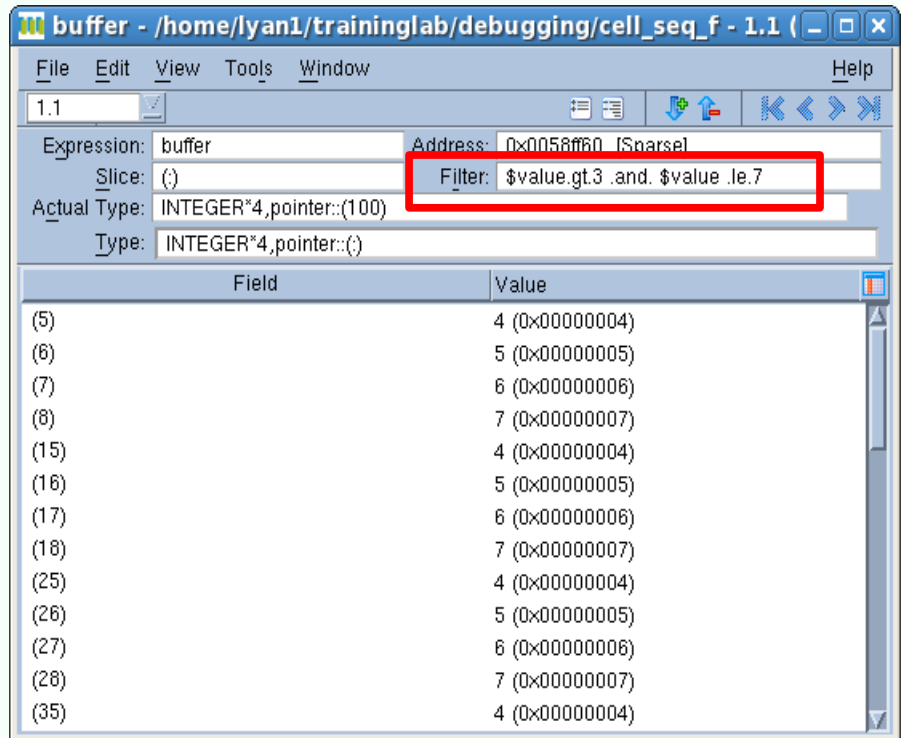




# Handling Arrays (2)

- Filtering

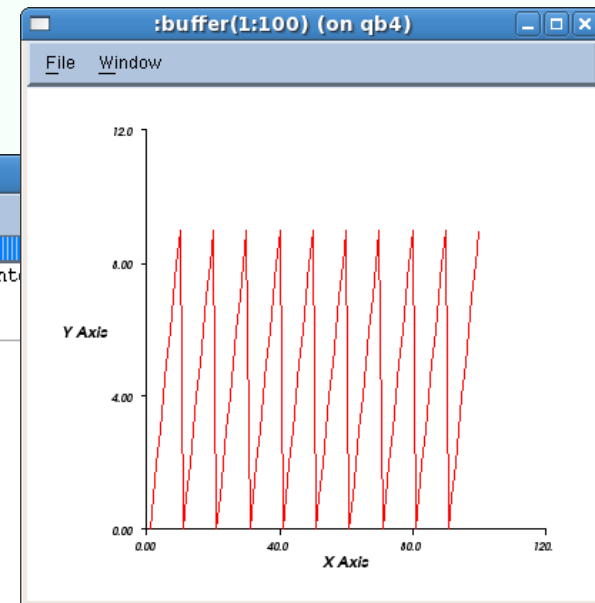
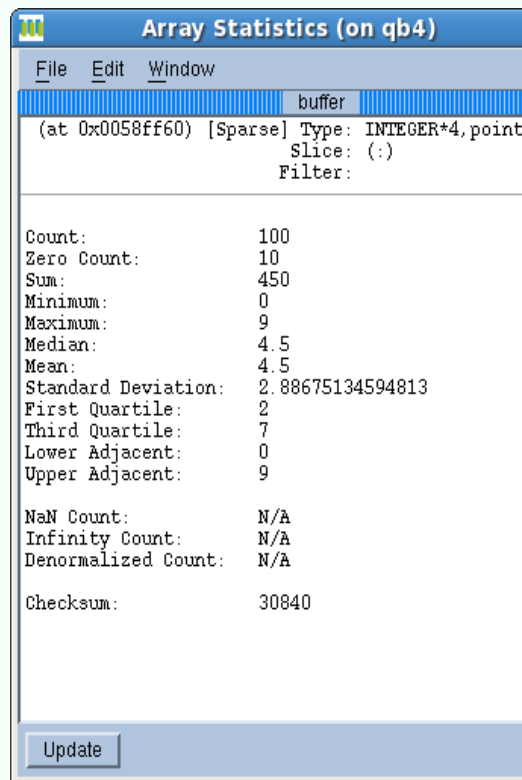
- Display array subsection by applying a filter (**filter** field in the variable window)
- Available filter options
  - Arithmetic comparison to a constant
  - Comparison to NaNs and Infs
  - Conditions can be combined by using logic operators





## Handling Arrays (3)

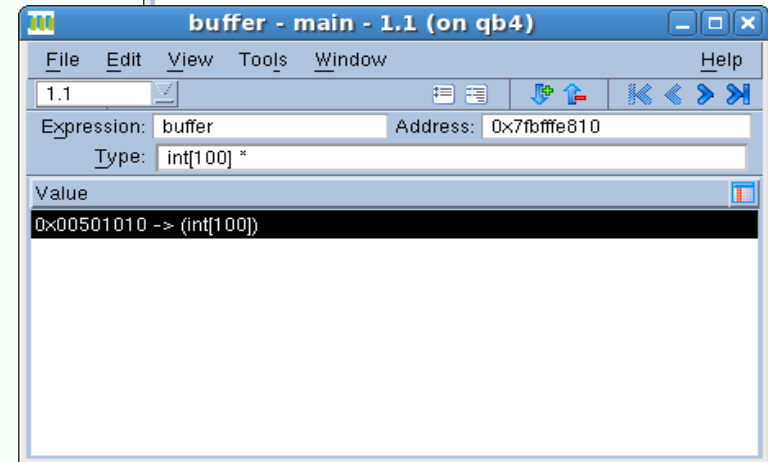
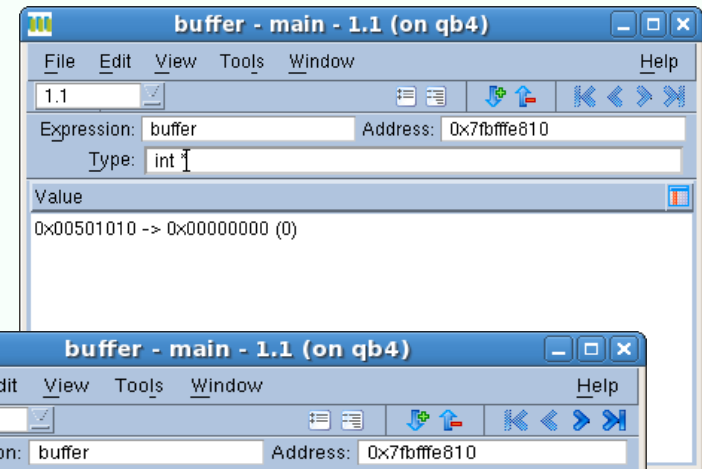
- Visualization
  - Variable window -> Tools -> Visualization
- Statistics
  - Variable window -> Tools -> Statistics





# Viewing Dynamic Arrays in C/C++

- Edit “type” in the variable window
- Tell TotalView how to interpret the memory from a starting location
- Example
  - To view an array of 100 integers
    - `int * -> int[100]*`





# Outline

- Overview
- Getting Started
- Debugging with TotalView
  - TotalView GUI
  - Basic debugging techniques
  - Features for parallel programs





# Bugs in Parallel Programs

- Parallel programs are prone to the usual bugs found in sequential programs, plus
  - Erroneous use of language features
    - Mismatched parameters, missing mandatory calls etc.
  - Defective space decomposition
  - Incorrect/improper synchronization
  - Hidden serialization
  - ...





# Debugging Parallel programs with TotalView

- Everything we talked about TotalView still works (well, almost)
  - Exceptions: stepping over a communication call while the other processes are stopped or being held
- Additional features
  - Scope of Control Commands
    - Group/Process/Thread
  - Displaying message queues (MPI programs)





# Scope of Control Commands

- For serial programs
  - Not an issue because there is only one execution stream
- For parallel programs, we need to decide the scope to which a control command applies
  - The process window always focuses on one process/thread
  - Need to set the appropriate scope when
    - Giving control commands
    - Setting action points
  - Switch between process/threads
    - “p+/p-” and “t+/t-” button
    - Through the root window
    - Through the process/thread tab





# Process/Thread Groups

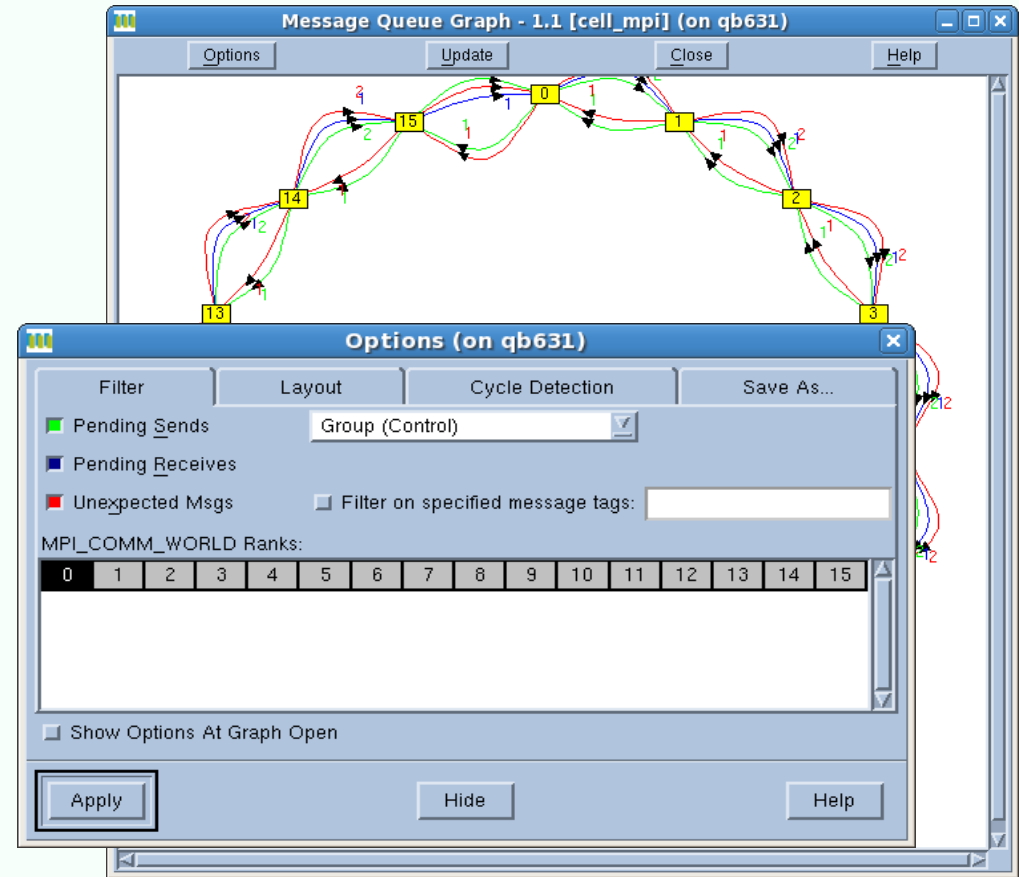
- Group (control): all processes and threads
- Group (workers): all threads that are executing user code
- Rank X: current process and its threads
- Process (workers): user threads in the current process
- Thread X.Y: current thread
- User defined group
  - Group -> Custom Groups, or
  - Create in call graph





# Displaying Message Queues

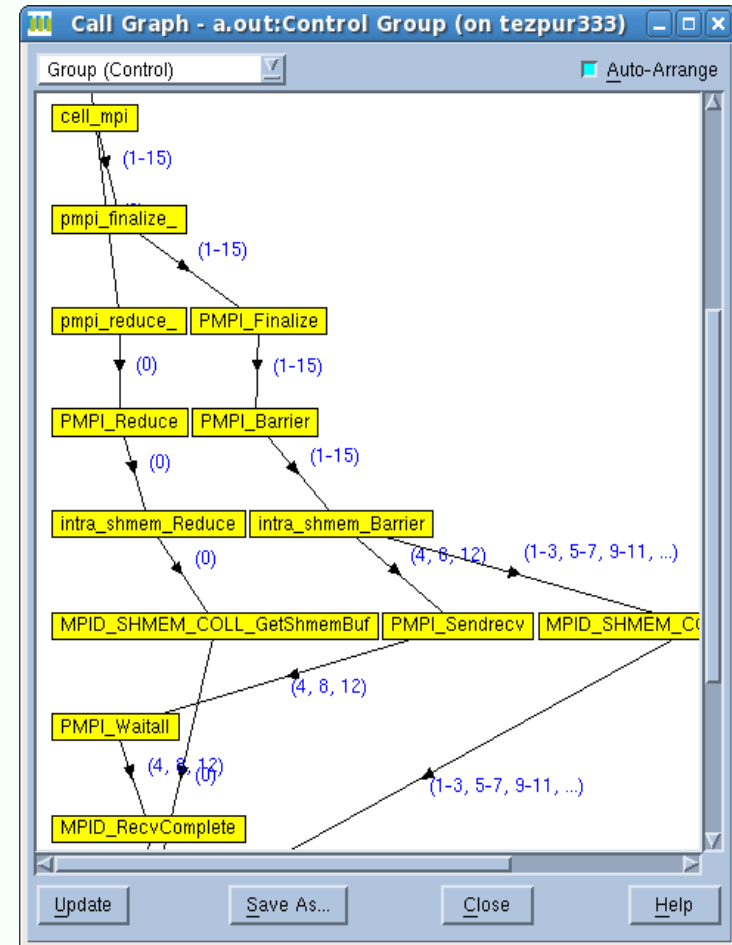
- Detect
  - Deadlocks
  - Load balancing issues
- To access
  - Tools -> Message Queue Graph





# Displaying Call Graph

- Quick view of program state
  - Nodes are functions
  - Edges are calls
  - Look for outliers
- To access
  - Tools -> Call Graph





# Not Covered

- Memory debugging
  - Leak detection
  - Heap status
  - Memory usage
  - Memory comparison
  - ...
- Command line interface
- Command line options





# References and Additional Resources

- TotalView user manual
  - <http://www.totalviewtech.com/support/documentation/totalview/>
- LLNL Total View tutorial
  - <https://computing.llnl.gov/tutorials/totalview/>
- HPCBugBase
  - [http://www.hpcbugbase.org/index.php/Main\\_Page](http://www.hpcbugbase.org/index.php/Main_Page)





# Exercise

- Files and instructions
  - <http://www.cct.lsu.edu/~lyan1/totalview/uno.php>
- Use the serial program to get familiar with the features of TotalView
  - It is bug-free
- Debug the parallel program
  - Check the result by comparing to that of the serial program

