

Debugging with Totalview and DDT

Le Yan

*User Services
HPC @ LSU*



Three Steps of Code Development

- Debugging
 - Make sure the code runs and yields correct results
- Profiling
 - Analyze the code to identify performance bottlenecks
- Optimization
 - Make the code run faster and/or consume less resources



Debugging Essentials

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



Debugging Methods

- Write/print/printf
- Compiler flags
 - Array bound check, floating point exception etc.
- Debuggers
 - Command line: gdb
 - Graphic: Totalview, DDT, Valgrind, Eclipse



What a Debugger Can Do

- Tell you where the program crashes
- Help to gain a better understanding of the context
- Debuggers cannot
 - Tell you how to solve the problem
 - Detect a correctness problem
 - Validation is very important



TotalView & DDT

- Powerful debuggers
 - Can be used to debug both serial and parallel programs
 - Support multiple languages
 - Both supports CUDA
 - Supported on most architecture/platforms
 - Graphic user interface
 - Totalview also has a command line interface
 - Numerous other features
 - Array visualization
 - Memory debugging
 - ...



Availability

- TotalView
 - 8.8.0 on Queen Bee (+totalview-8.8.0)
 - 8.3.0 on Queen Bee, Tezpur, Philip and Eric (+totalview-8.3.0.1)
- DDT
 - 2.6 on all LONI and LSU HPC Linux clusters (+ddt-2.6)



Preparing for a Debugging Session

- Compile the program with debugging turned on and optimization turned off (-O0 -g)
- Add softenv keys and resoft
- Make sure X Windows works
- Submit an interactive job session



Working with Debuggers

- One can start debugging by
 - Starting the debugger with the executable
 - Debugging a core dump
 - Attaching to a running (or hanging) process
- Common debugging operations
 - Setting up action points
 - Controlling the execution
 - Examining the value of variables
 - ...



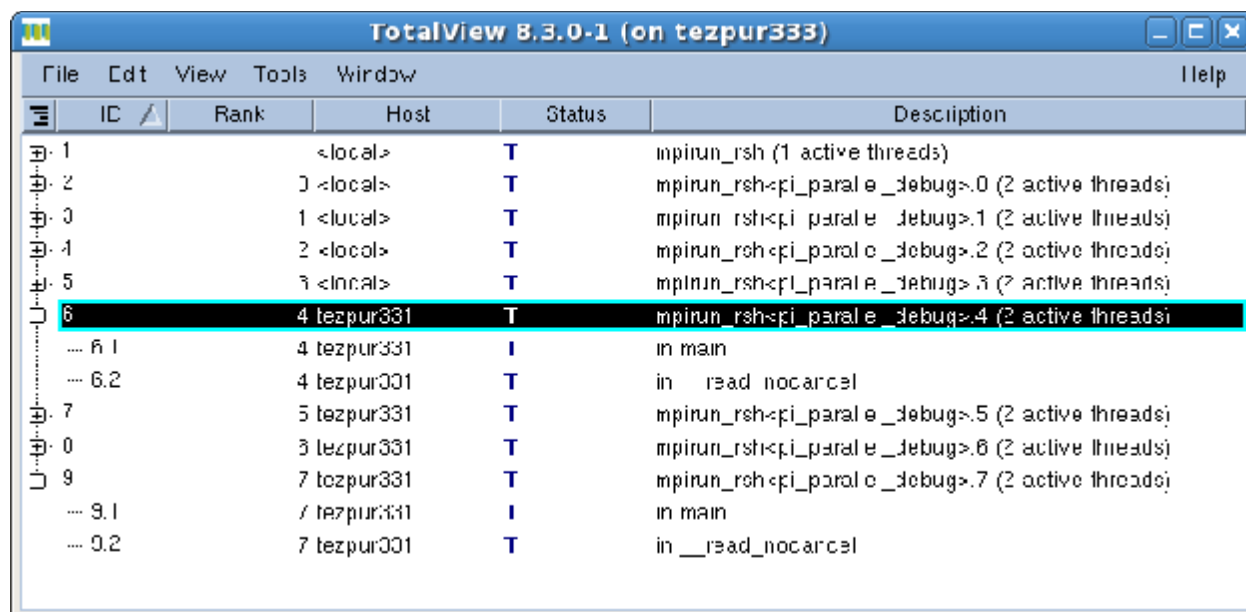
Launching a Debugging Session

- Serial program
 - Totalview
 - `totalview <executable> -a <program options>`
 - DDT
 - `ddt -start <executable> <program options>`
- Parallel program
 - Totalview
 - `mpirun_rsh -tv -np <num_procs> <host list> <executable> <program options>`
 - `mpirun_rsh -tv -np <num_procs> -hostfile <path_to_hostfile> <executable> <program options>`
 - DDT
 - `ddt -start -np <num_procs> <executable> <program options>`



TotalView GUI – Root Window

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

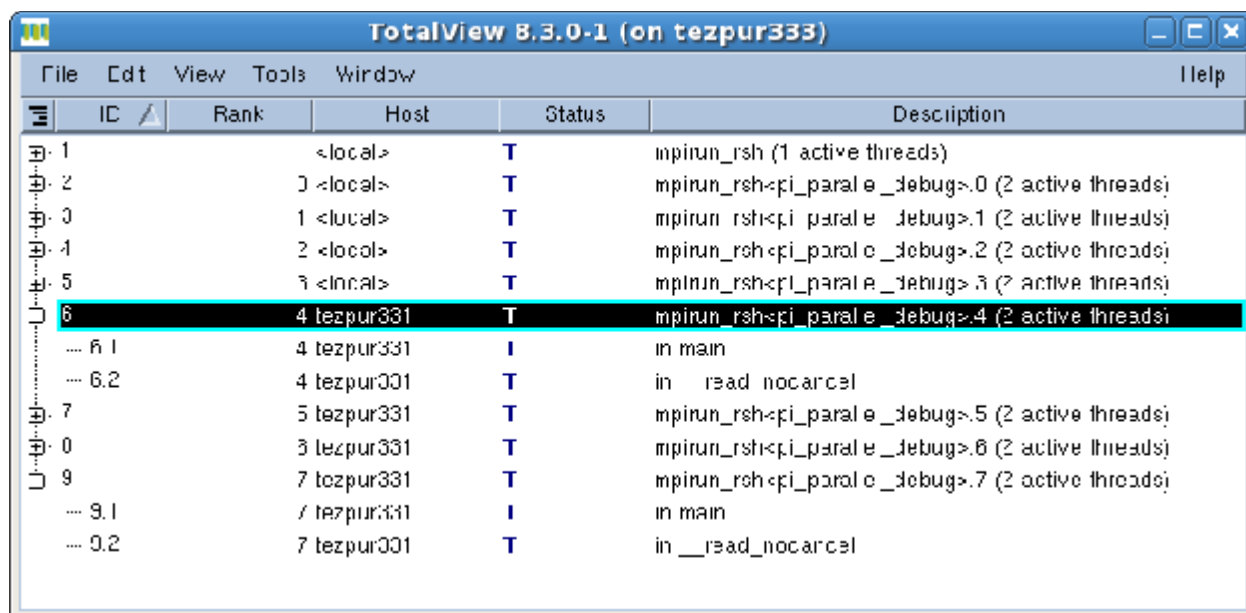


IC	Rank	Host	Status	Description
1	<local>		T	mpirun_rsh (1 active threads)
2	0 <local>		T	mpirun_rsh<pi_parallel_debug>.0 (2 active threads)
3	1 <local>		T	mpirun_rsh<pi_parallel_debug>.1 (2 active threads)
4	2 <local>		T	mpirun_rsh<pi_parallel_debug>.2 (2 active threads)
5	3 <local>		T	mpirun_rsh<pi_parallel_debug>.3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



TotalView GUI – Root Window

Status code	Description
Blank	Exited
B	At breakpoint
E	Error
H	Held
K	In kernel
M	Mixed
R	Running
T	Stopped
W	At watchpoint

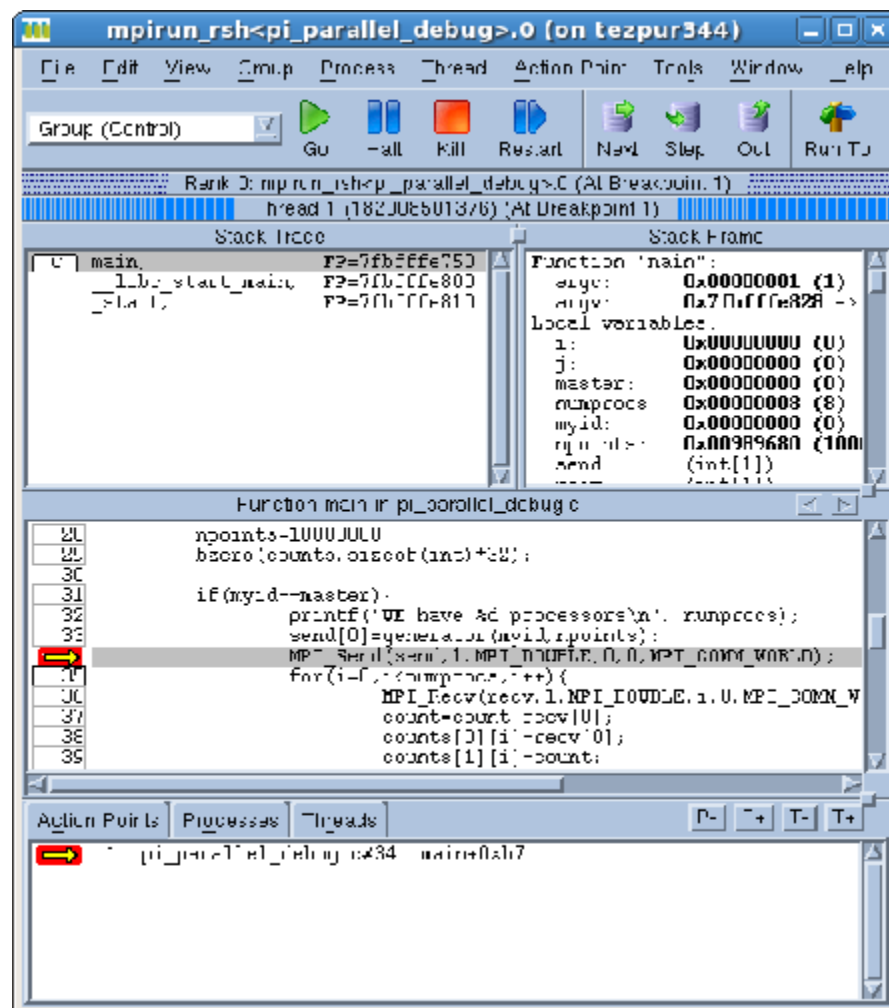


IC	Rank	Host	Status	Description
1	<local>		T	mpirun_rsh (1 active threads)
2	0 <local>		T	mpirun_rsh<pi_parallel_debug>.0 (2 active threads)
3	1 <local>		T	mpirun_rsh<pi_parallel_debug>.1 (2 active threads)
4	2 <local>		T	mpirun_rsh<pi_parallel_debug>.2 (2 active threads)
5	3 <local>		T	mpirun_rsh<pi_parallel_debug>.3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



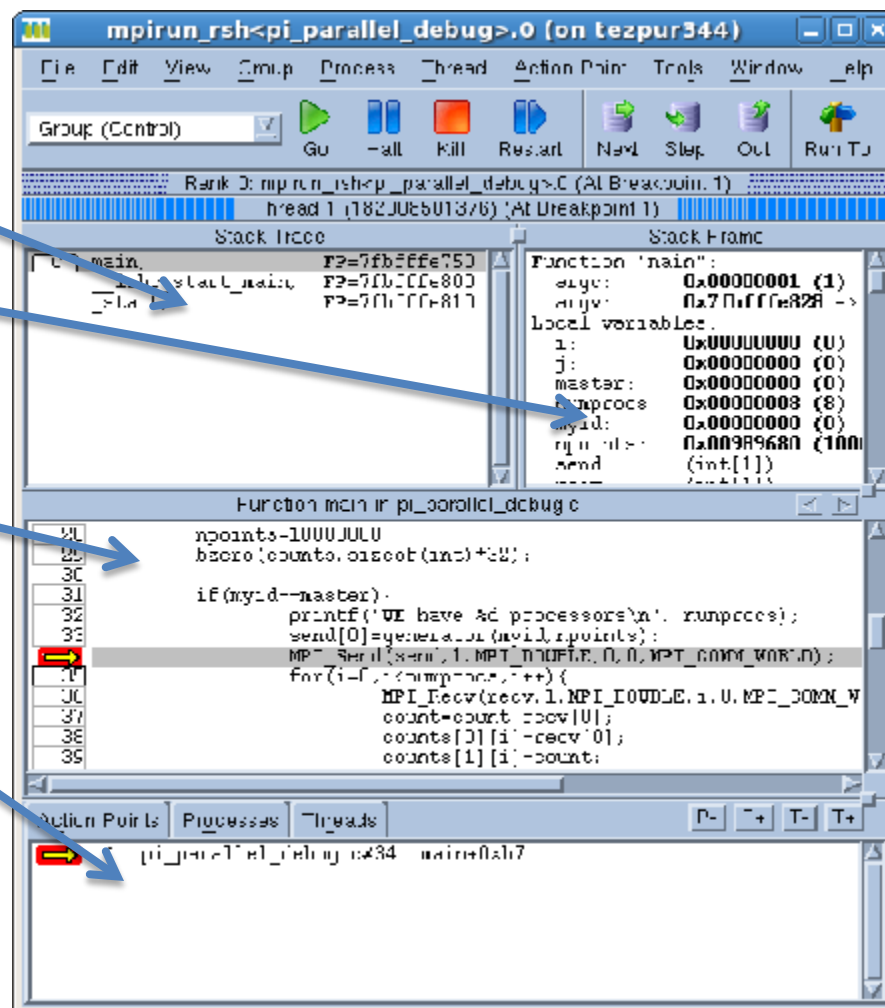
TotalView GUI – Process Window

- Appears when TotalView is started
- For parallel programs each process/thread may have its own process window



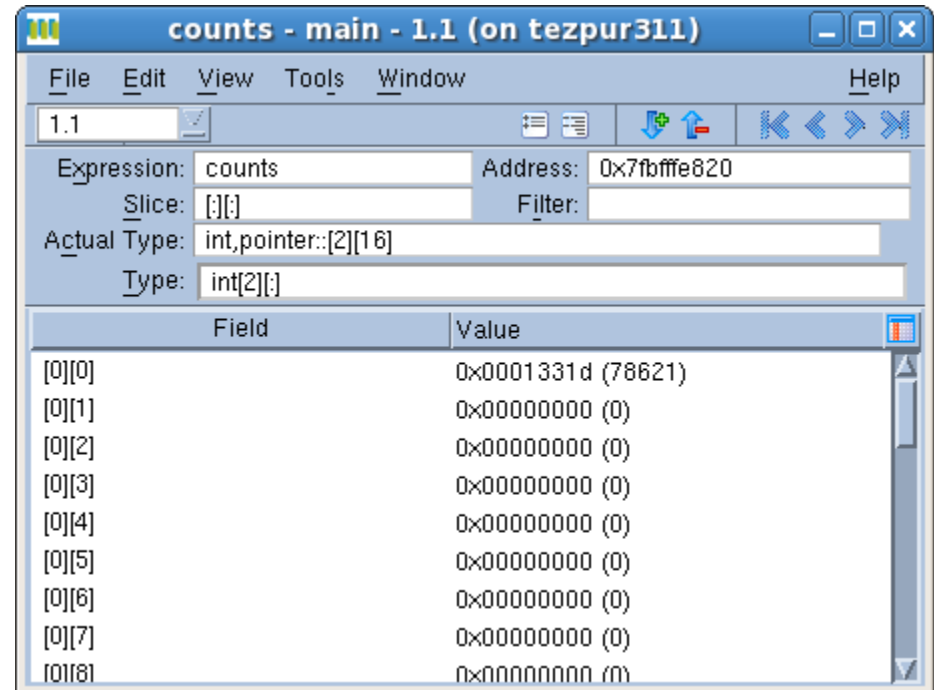
TotalView GUI – Process Window

- 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
 - Lists of action points
 - Lists of processes
 - List of threads

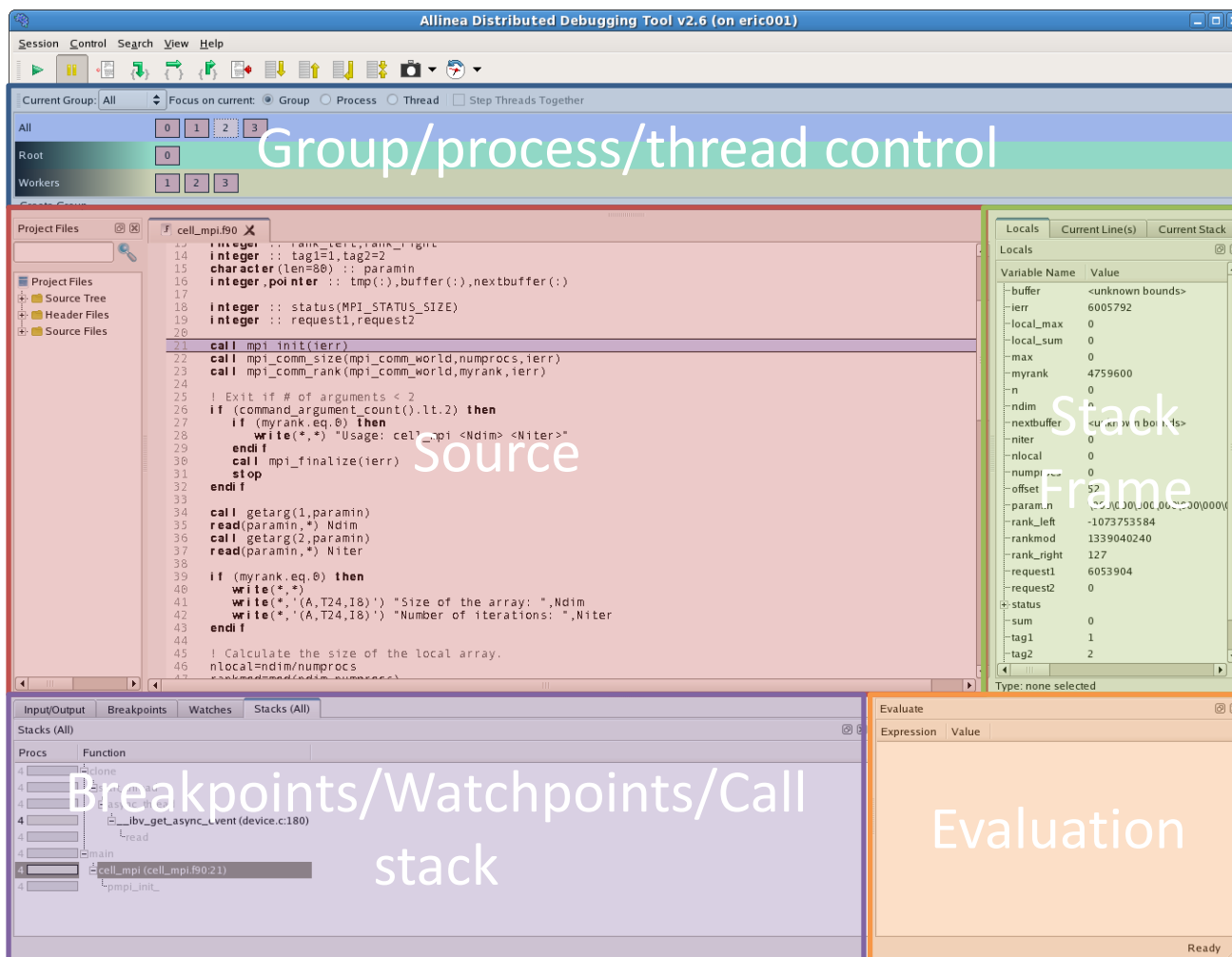


TotalView GUI – Variable Window

- Can be opened by double-clicking on a variable name
 - Called “dive” in Totalview terminology
- Display detailed information of a variable
- One can also edit the data here



DDT GUI



Group/process/thread control

Source

Stack Frame

Breakpoints/Watchpoints/Call stack

Evaluation



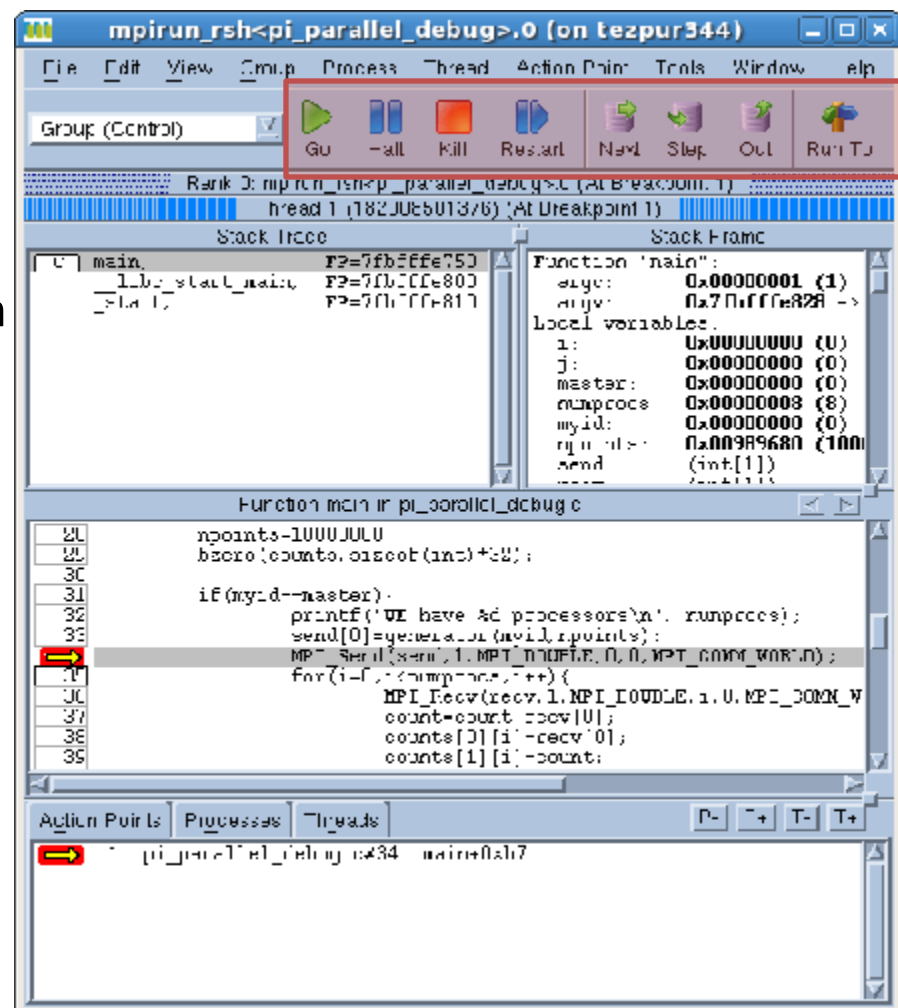
Other Ways of Starting a Debugging Session

- Open a core file
 - Need to select an executable
 - Can only browse variables and evaluate expressions since there is no active process
- Attach to one or more running (or hanging) processes














TotalView: Controlling Execution

- Commonly used commands
 - Go: start/resume execution
 - Halt: stop execution
 - Kill: terminate debugging session
 - Restart: restart a running program
 - Next: run to next source line WITHOUT stepping into another function or subroutine
 - Step: run to next source line
 - Out: run to the completion of a function or subroutine



DDT: Controlling Execution

- Similar commands to TotalView
- A few more commands to move up and down stack frame
 - The “align stack frames” command is useful to bring paused processes to the same place in the program

	Play/Continue	F9
	Pause	F10
	Add Breakpoint...	
	Step Into	F5
	Step Over	F8
	Step Out	F6
	Run To Line...	
	Down Stack Frame	Ctrl+D
	Up Stack Frame	Ctrl+U
	Bottom Stack Frame	Ctrl+B
	Align Stack Frames With Current	Ctrl+A



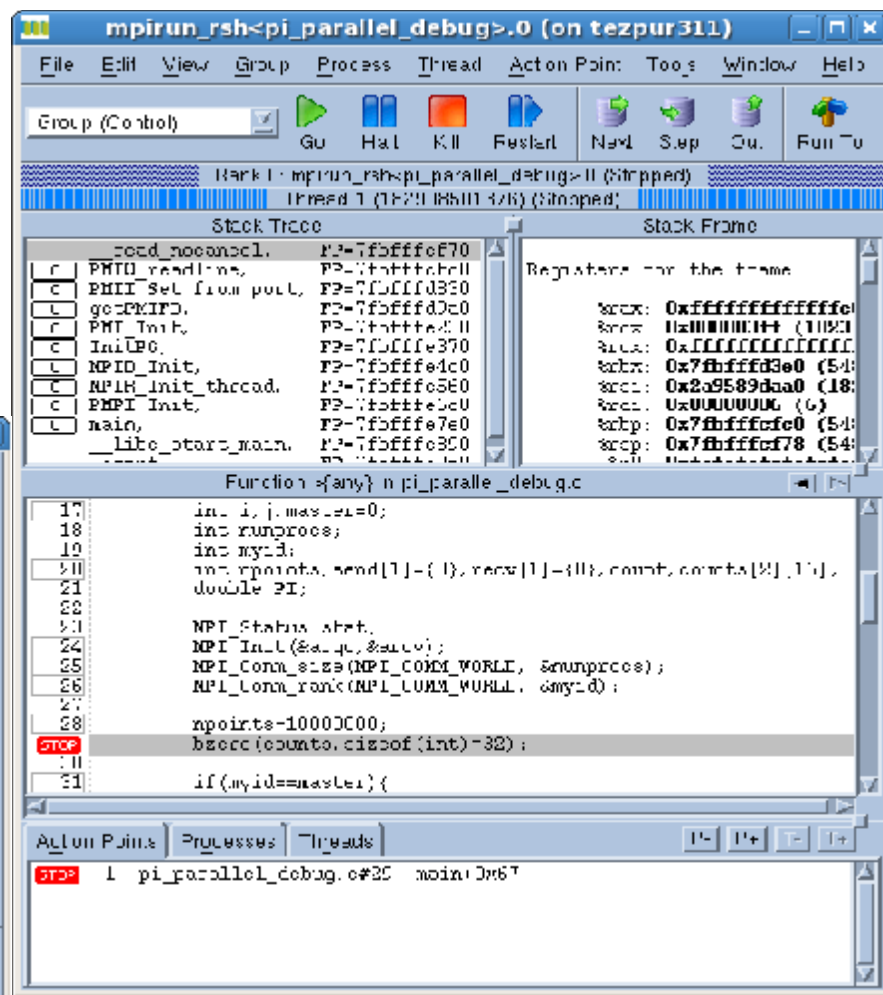
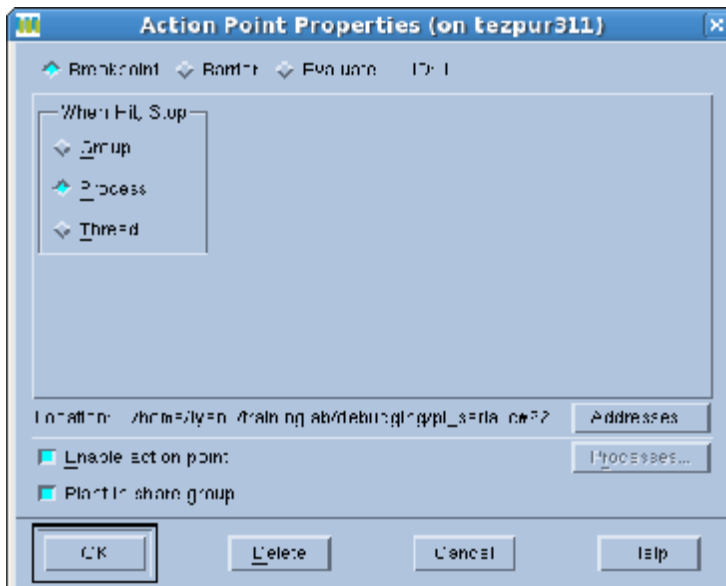
Action Points

- Break points stop the execution when reached
 - Can be conditional
- Barrier points synchronize a set of processes or threads
- Evaluation points cause a code segment to be executed when reached
- Watch points allow the programmer monitor a location in memory
 - Can stop execution or evaluate an expression when its value changes



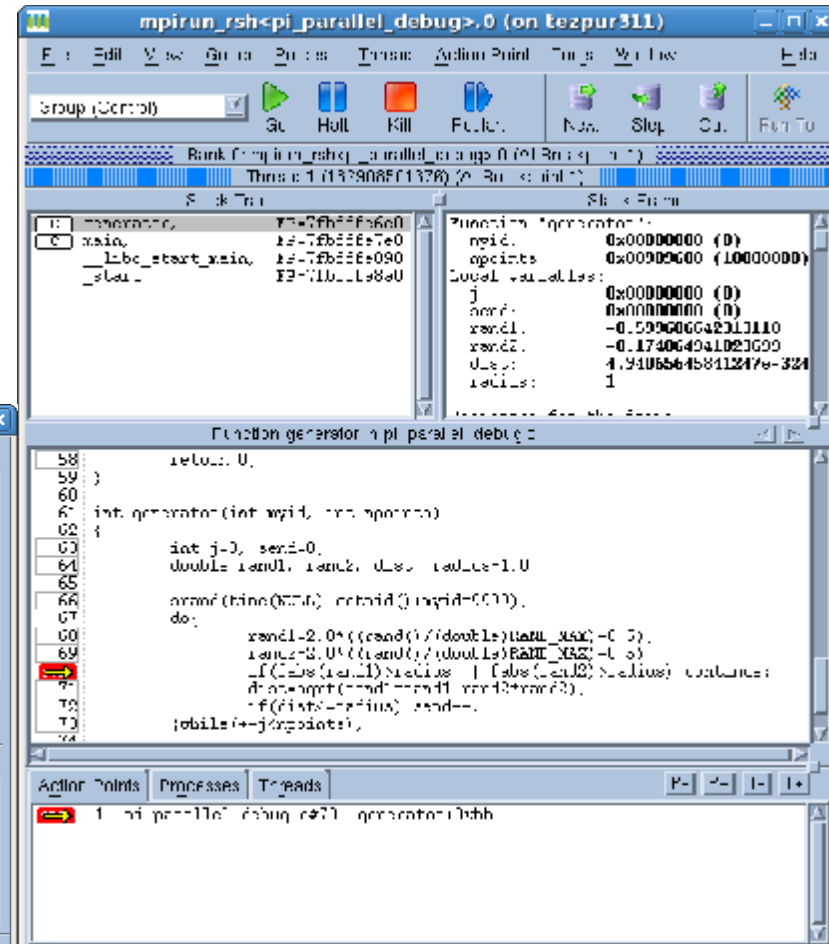
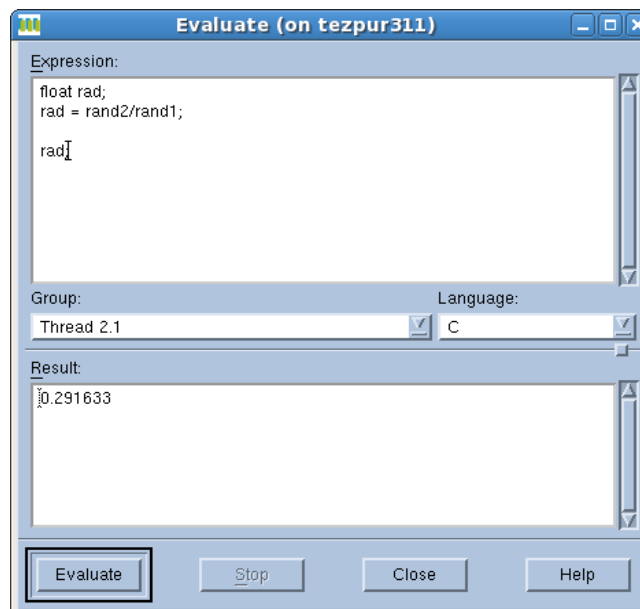
TotalView: Break points

- How to set
 - Left click on the line number
 - Right click on a line -> "set breakpoint"
- Will appear in the action point list



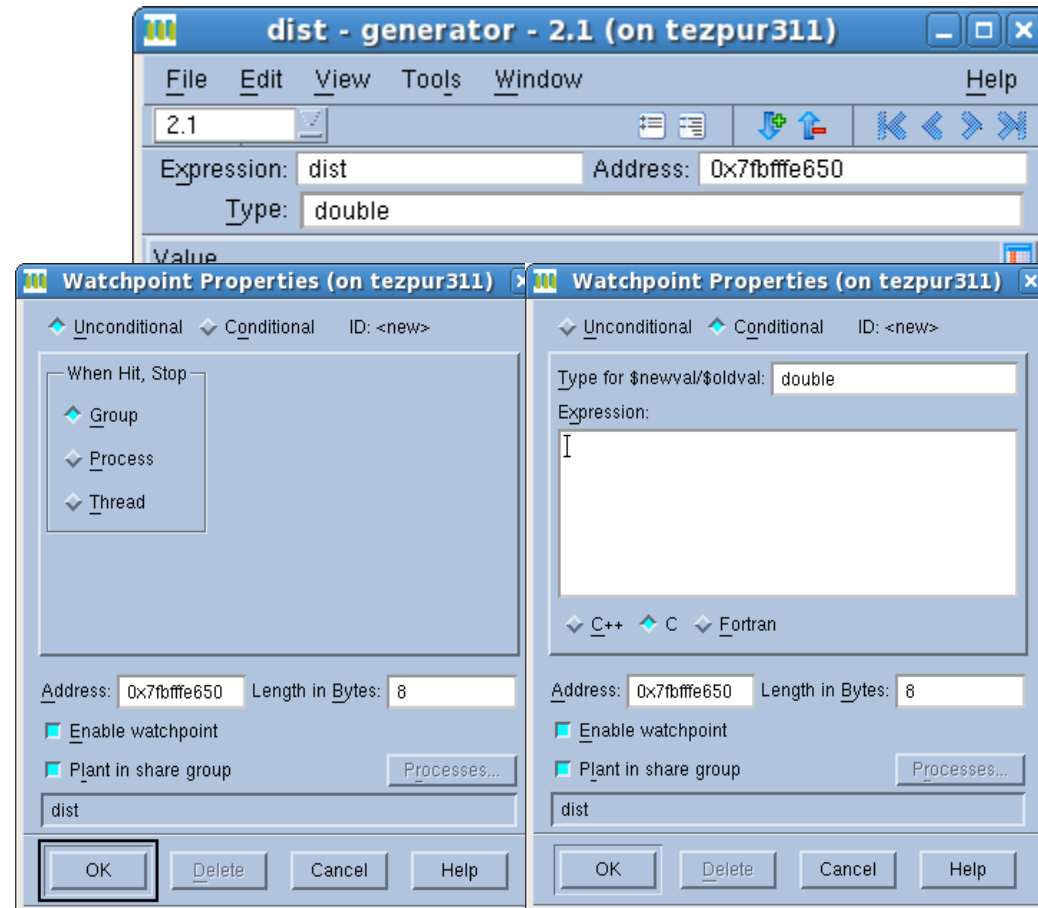
TotalView: Evaluation Points

- How to set
 - “Tools” -> “Evaluate”
- Execute a small segment of code at specified location
 - Useful when testing on-the-fly fixes



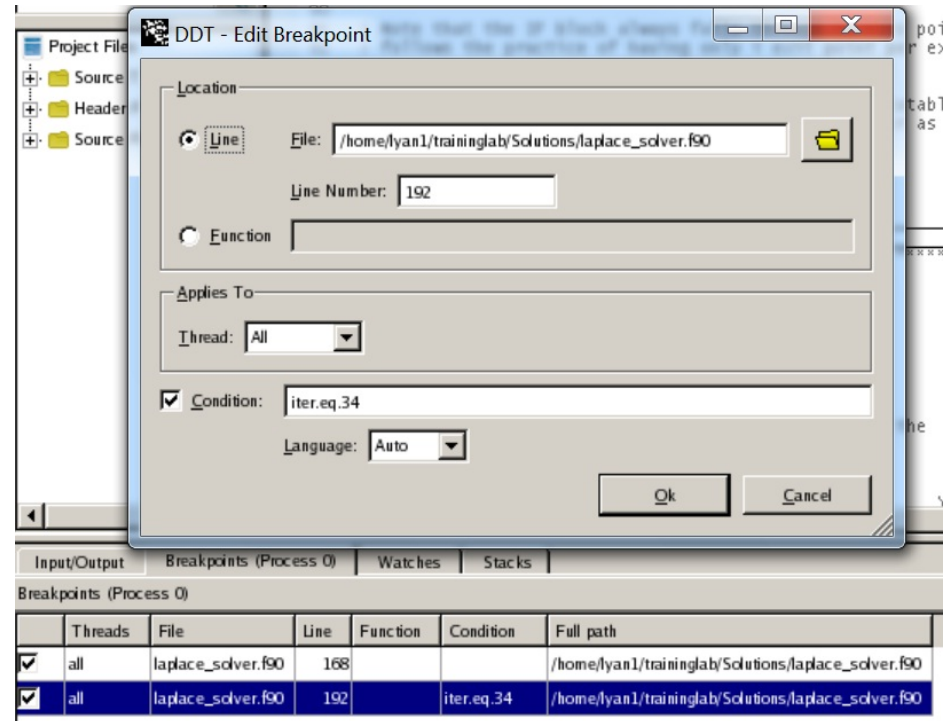
TotalView: Watch Points

- Monitor a memory location and stop execution when it is overwritten
- How to set
 - Right click on a variable -> “Create watchpoint”
- Can be conditional
 - Example: only watch this memory location after a certain number of iterations



DDT: Breakpoints

- How to set
 - Double click on a line
 - Right click on a line -> "Add breakpoint"
- Will appear in the breakpoint list



DDT: Evaluation and Watch Points

- How to set
 - Right click on variable -> “Add to Evaluations” or “Add to Watches”
- DDT does not provide as many options for evaluation and watch points

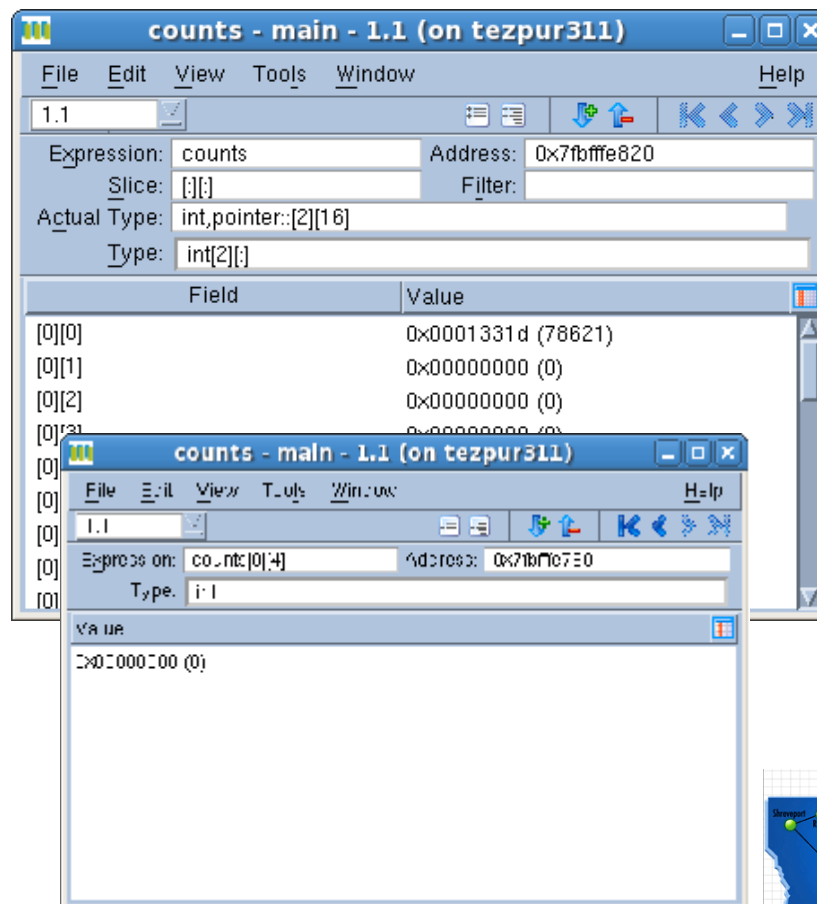
Evaluate	
Expression	Value
dt	1.0430250141410511
dt*i-j	21.555532084666538

Watches	
Expression	Stack frame
dt	#0 laplace
t2	#0 laplace



TotalView: Diving On An Object

- “Diving” means “showing more details on an object”
- One can dive on
 - Variables
 - Processes/threads
 - Subroutines
- Use “undive” to go back



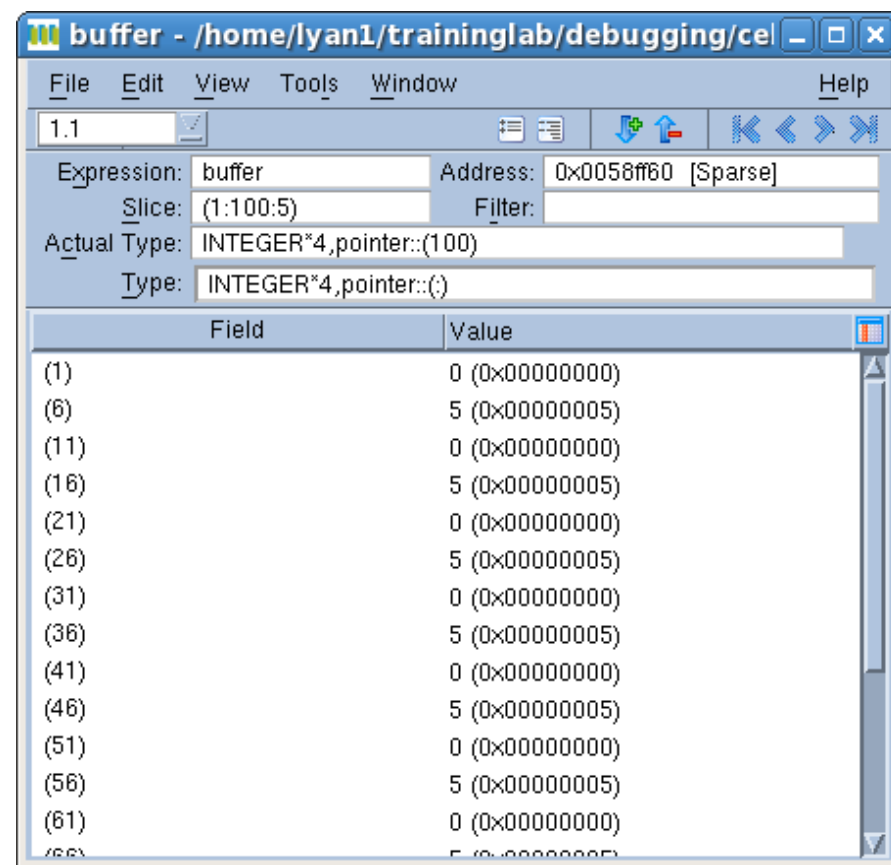
TotalView: Viewing/Editing Data

- 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



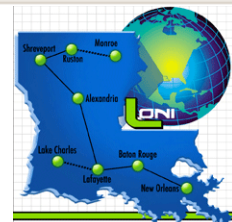
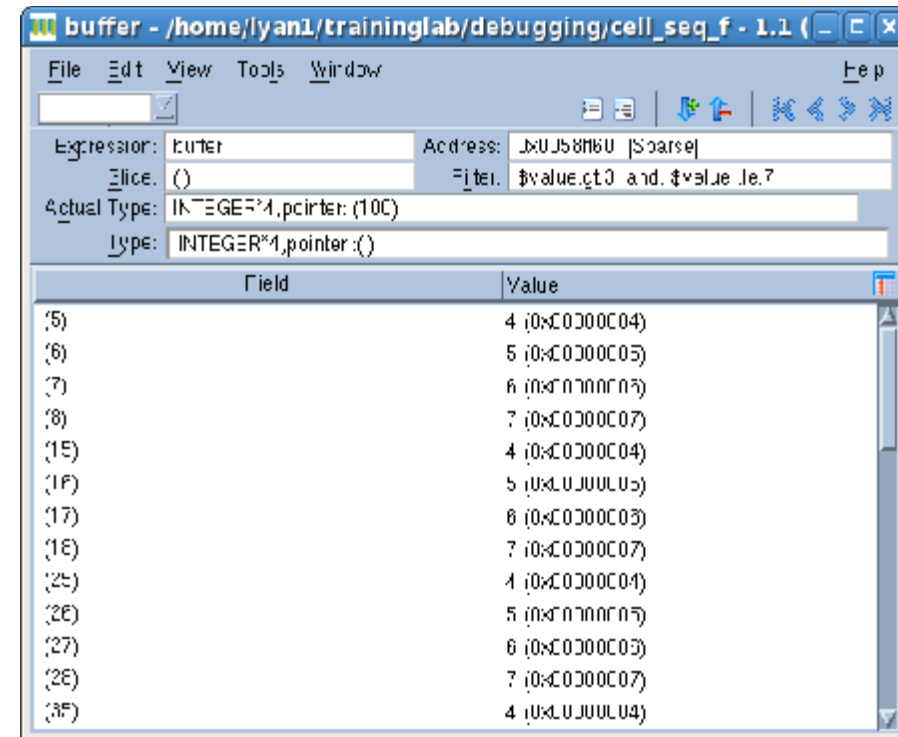
TotalView: Handling Arrays (1)

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



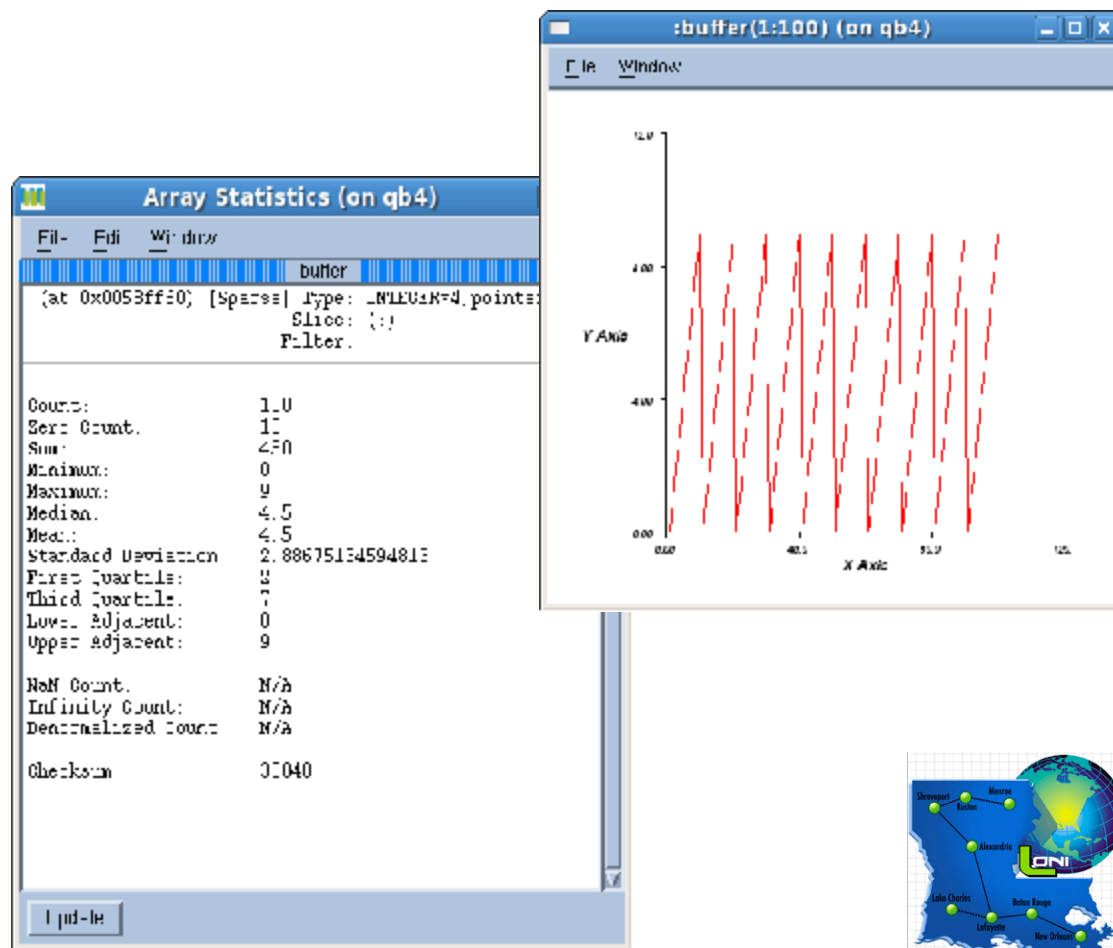
TotalView: Handling Arrays (2)

- 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



TotalView: Handling Arrays (3)

- Visualization
- Statistics



DDT: Handling Arrays

DDT - Multi-Dimensional Array Viewer

Array Expression: `t(i, j)`

Range of `i`

From: 401 To: 501 Display: Columns

Range of `j`

From: 401 To: 501 Display: Rows

Aggregate Function: Sum

☒ Filter: > 0

☐ Auto-update

Evaluate Cancel

Data Table Statistics

	469	470	471	472	473	474
414	4.4885971595261109e-18	2.9624739970742382e-16	9.6325294547121972e-15	2.0569444729145469e-13	3.2446542390234457e-12	4.03199631046892e-11
415	4.4994390158148434e-18	2.9696298786507916e-16	9.6557961775694907e-15	2.0619129847971988e-13	3.2524915189005686e-12	4.0417354801499458e-11
416	4.5102812856938822e-18	2.9767855203449673e-16	9.6790633971487421e-15	2.0668814147848342e-13	3.260328889727027e-12	4.0514745639228883e-11
417	4.5211231419826147e-18	2.9839412737085257e-16	9.7023302626946912e-15	2.071849891044953e-13	3.2681662011524048e-12	4.0612136988136901e-11
418	4.5319649982713473e-18	2.9910970270720841e-16	9.7255971282406404e-15	2.0768183673050719e-13	3.2760085125777826e-12	4.0709528337044919e-11
419	4.5428073291569251e-18	3.000536697662169e-16	9.7499512137169912e-15	2.0819353373337373e-13	3.283846994643411e-12	4.0800019131734343e-11

Expression "`t(i, j)`" evaluated for process 0 at 22:49.

Visualize in 3D Export to Spreadsheet... Close



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

- 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 摺+/t-?button
 - Through the root window
 - Through the process/thread tab



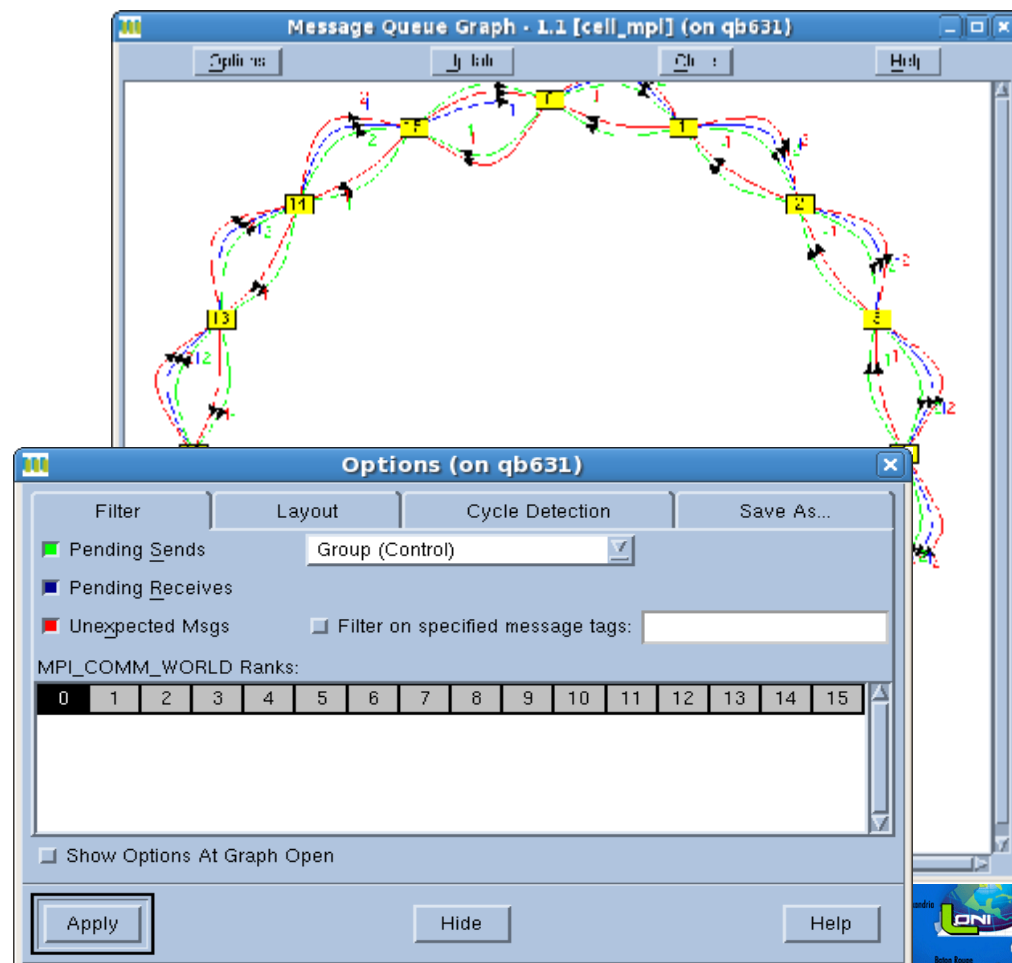
Process/Thread Groups

- 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



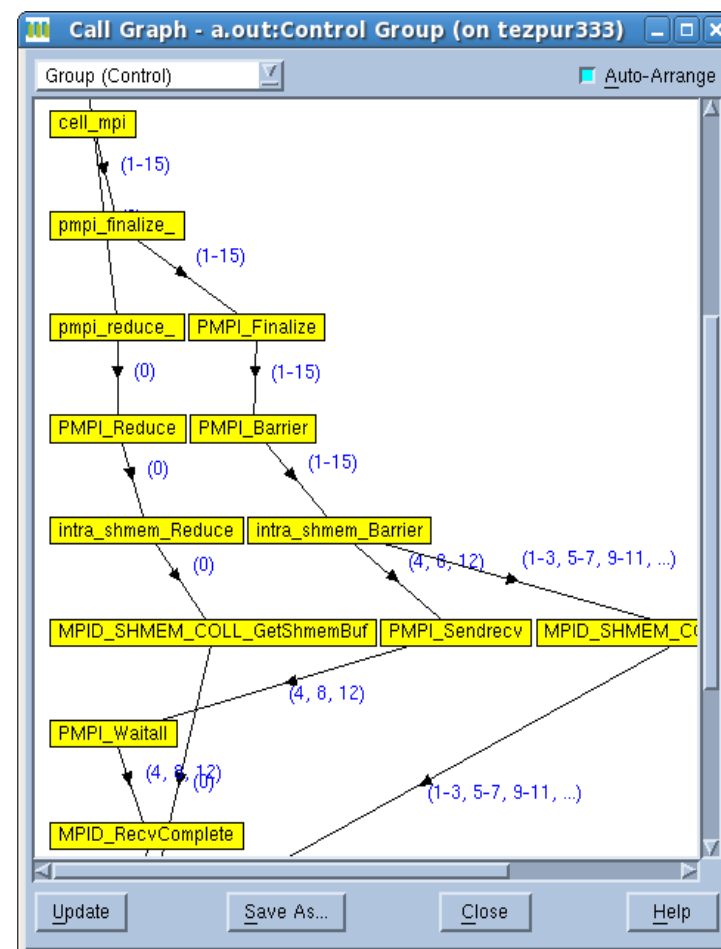
Displaying Message Queues

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



TotalView: Displaying Call Graph

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



DDT: Parallel Stack View

- Shows a tree of functions merged from every process in a group of processes
- Can create process groups based on their location
- Very helpful when dealing with a large number of processes

Input/Output*		Breakpoints	Watches	Stacks (All)
Stacks (All)				
Procs	Function			
64	clone			
64	main			
5	cell_mpi (cell_mpi.f90:69)			
1	cell_mpi (cell_mpi.f90:70)			
58	cell_mpi (cell_mpi.f90:82)			



Not Covered

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

