ATPESC 2018

Power Technologies and Techniques for Debugging HPC Applications

Bill Burns – Sr. Director of Product Development & Product Manager

RogueWave SOFTWARE

Innovate with Confidence

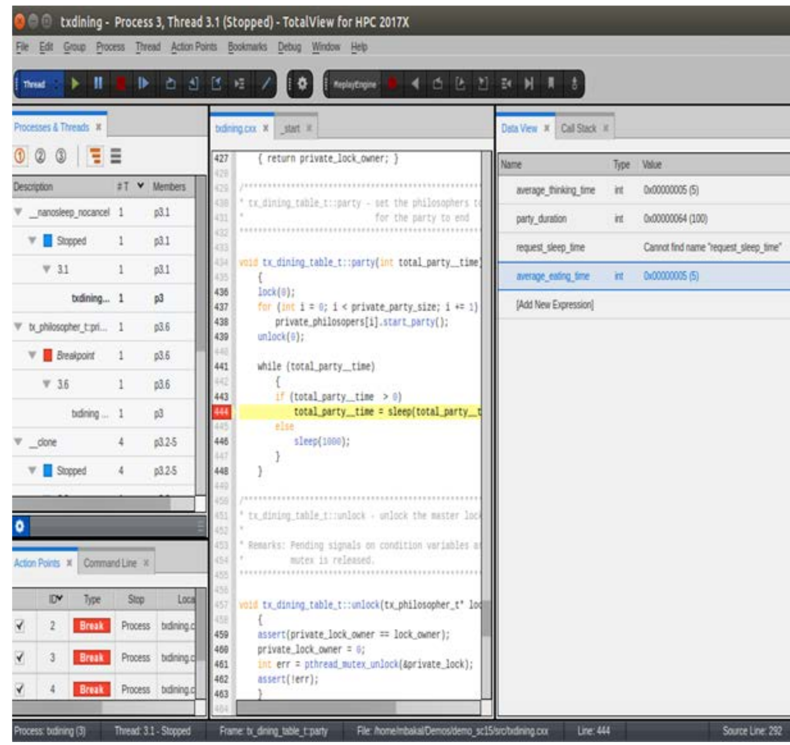
Agenda

- What is TotalView?
- Overview of TotalView features
- TotalView's new UI
- Python debugging
- Reverse debugging
- MPI and OpenMP parallel debugging
- Advanced C++ and Data debugging
- Memory debugging
- Unattended /Batch debugging
- Accelerator debugging
- Remote Display debugging
- Using TotalView
- TotalView resources and documentation
- Questions/Comments

TotalView Features

TotalView Features

- Comprehensive C, C++ and Fortran multi-core and multi-threaded analysis and debug environment
  - Thread specific breakpoints
  - Control individual thread execution
  - View thread specific stack and data
  - View complex data types easily
- CUDA and Xeon Phi debugging
- Integrated Reverse debugging
- Mixed Language - Python C/C++ debugging
- Track memory leaks in running applications
- Unattended debugging
- Linux, macOS and UNIX



- **More than just a tool to find bugs**
  - Understand complex code
  - Improve developer efficiency
  - Collaborate with team members
  - Improve code quality
  - Shorten development time

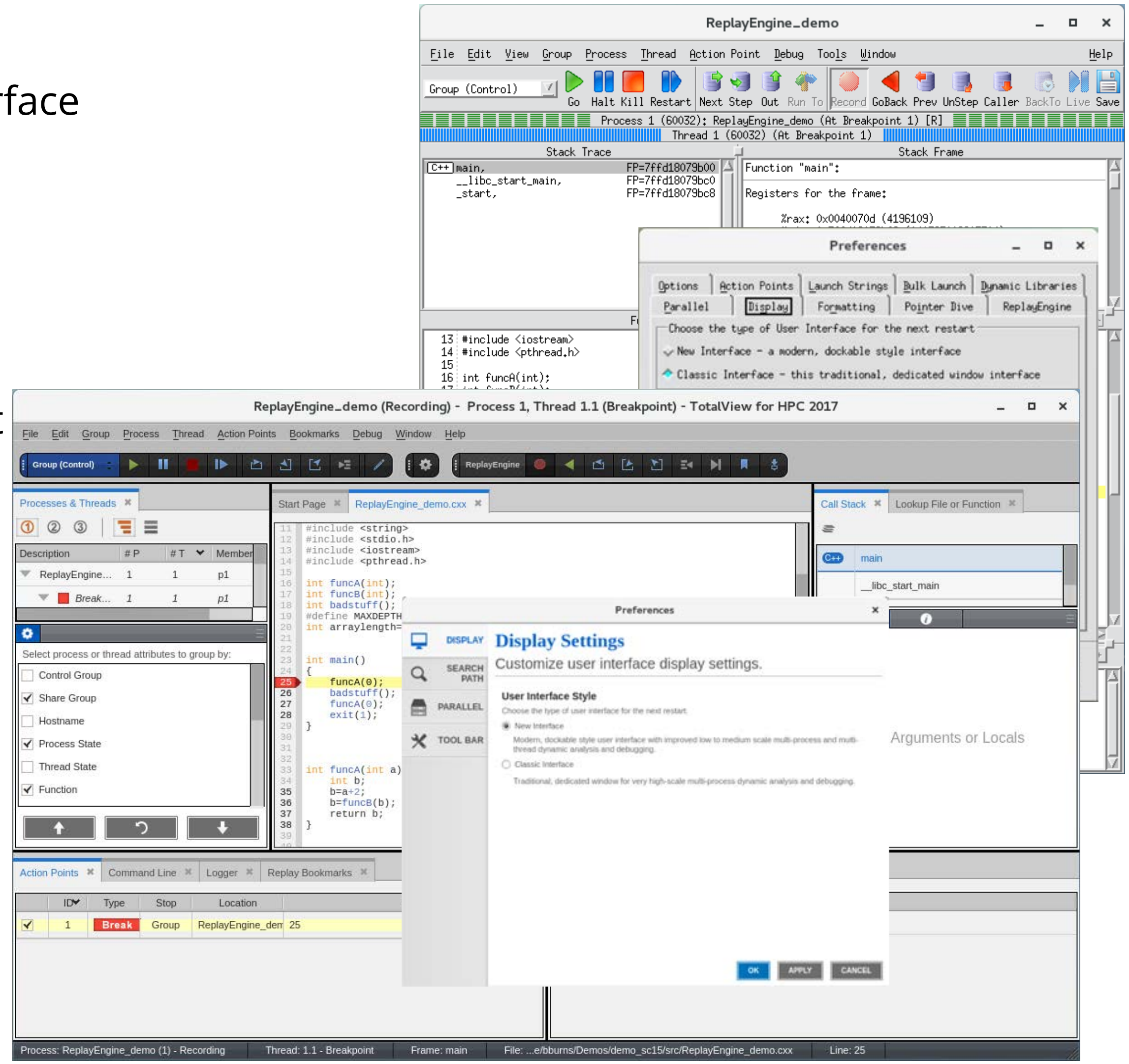
TotalView’s New UI

TotalView's New UI

- Provides a modern, dockable interface
- Easier to use, better workflows
- An architecture to grow
- To use:
  - Set UI preference
  - Or command line argument

    `totalview -newUI`

- New UI gaps:
  - Missing array slicing and striding, view across, data visualization
  - Memory debugging not integrated
  - No very high-scale support



Python Debugging

Python Development Trends

- Increased usage of Python to build applications that call out to C++
- Provides access to
  - High-performance routines
  - Leverage existing algorithms and libraries
  - Utilize advanced multi-threaded capabilities
  - Computational steering
- Calling between languages easily enabled using technologies such as SWIG, ctypes, Cython, CFFI, et al
- Debugging mixed language applications is not easy

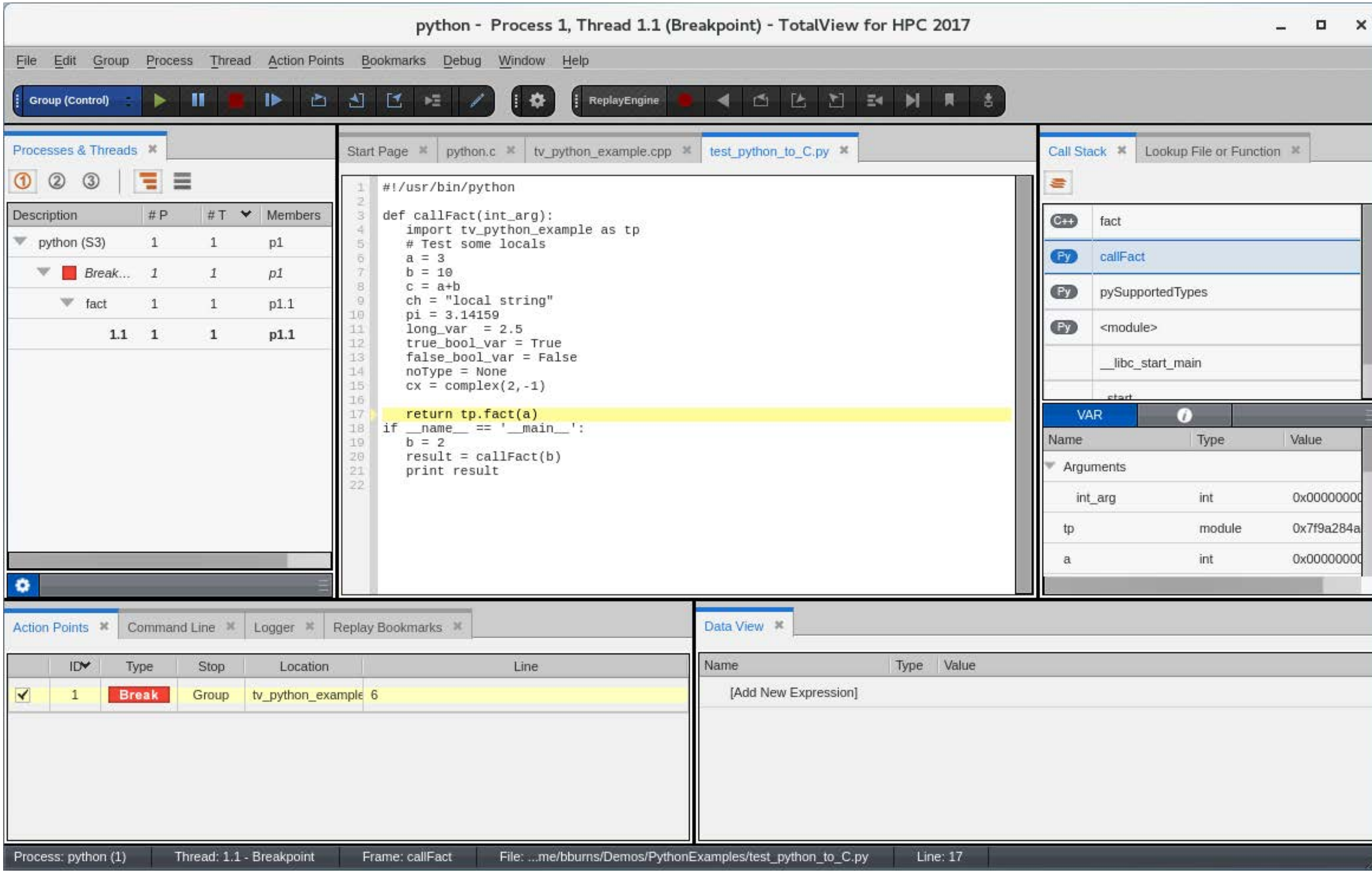
Python Debugging

- Debugging one language is difficult enough
- Understanding the flow of execution across language barriers is hard
- Examining and comparing data in both languages is challenging

- What TotalView provides:
  - Easy python debugging session setup
  - Fully integrated Python and C/C++ call stack
    - "Glue" layers between the languages removed
  - Easily examine and compare variables in Python and C++
  - Modest system requirements
  - Utilize reverse debugging and memory debugging
  - Python 2.7 (Python 3 coming soon)

- What TotalView does not provide (yet):
  - Setting breakpoints and stepping within Python code

Python Debugging Demo



Reverse Debugging

Reverse Debugging

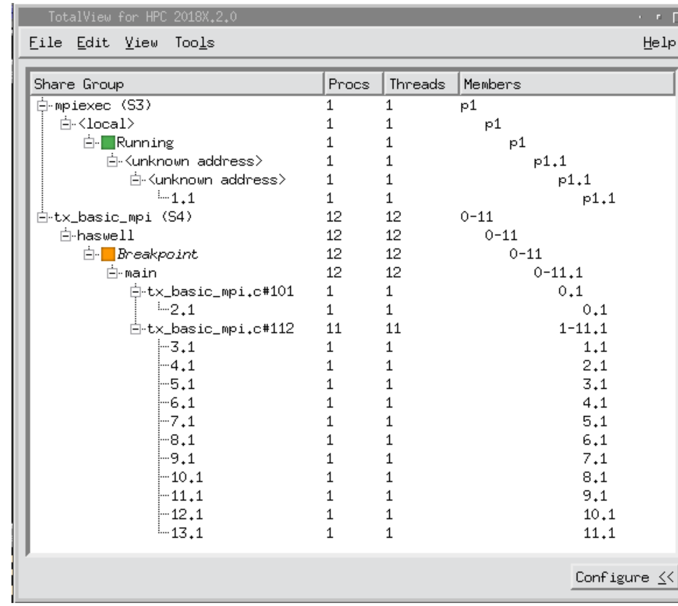
- Reverse debugging is an amazing feature that
  - Saves developers time by finding issues in one debugging session
  - Allows developers to quickly learn new or complex code
  - Enables collaboration and sharing of run sessions
- By
  - Capturing and deterministically reply execution
  - Enables stepping backwards and forward by function, line or instruction
  - Recording from the start of execution or on demand
  - Saving recording files for later analysis or collaboration
  - Linux x86/x86-64

- **Demo!**

MPI and OpenMP Parallel Debugging

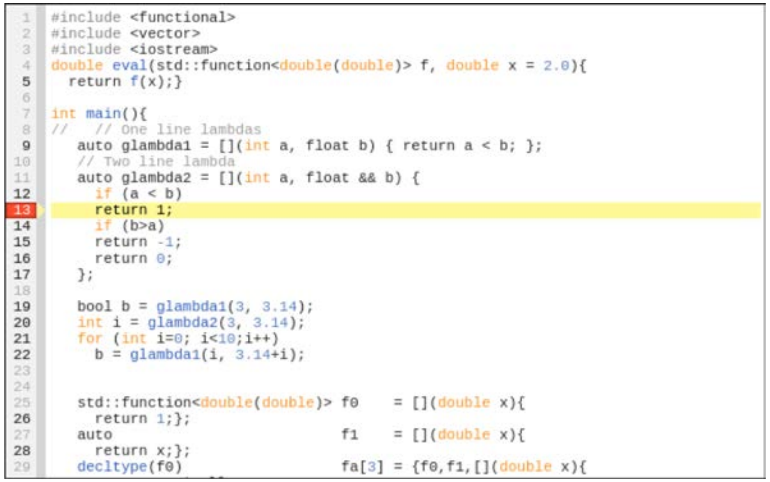
Parallel Debugging

- TotalView provides the power to
  - Simultaneously debug many MPI processes and threads in a single debugging session
  - Help locate deadlocks and race conditions
  - Examine message queues between MPI processes
  - Debug all or a subset of the processes in your job
  - Understand complex parallel applications
- By
  - Providing control of entire groups processes, individual processes or even down to individual threads within a process
  - Enabling thread level breakpoints and barrier controls
  - Showing aggregated process and thread state display

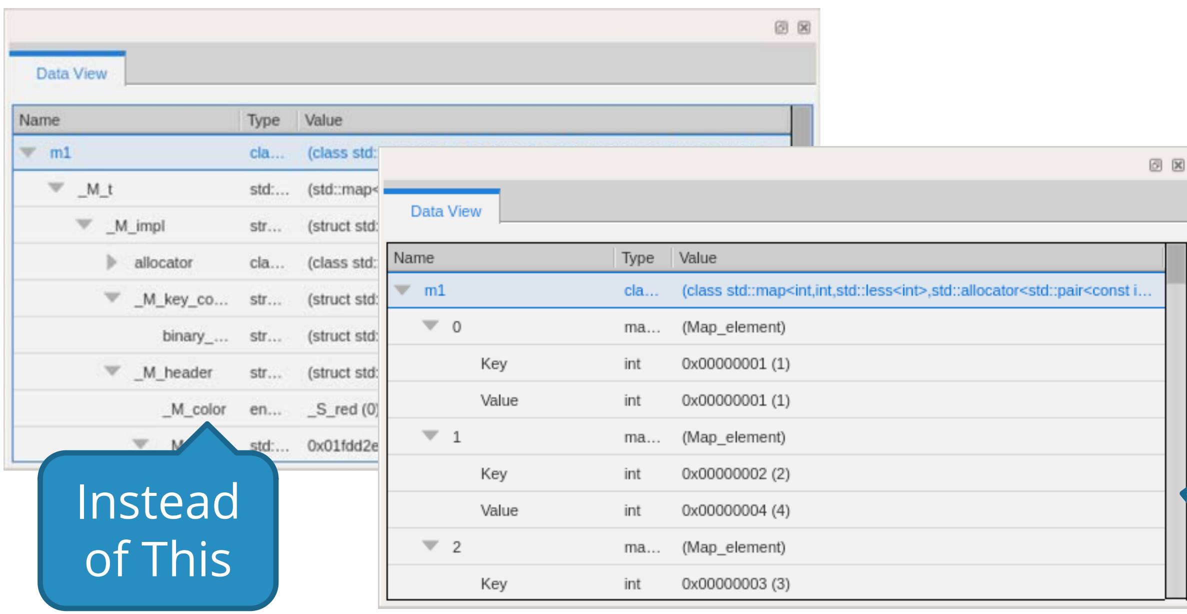


Advanced C++ and Data Debugging

Advanced C++ and Data Debugging



- TotalView supports debugging the latest C++11/14 features including:
  - lambdas, transformations for smart pointers, auto types, R-Value references, range-based loops, strongly-typed enums, initializer lists, user defined literals

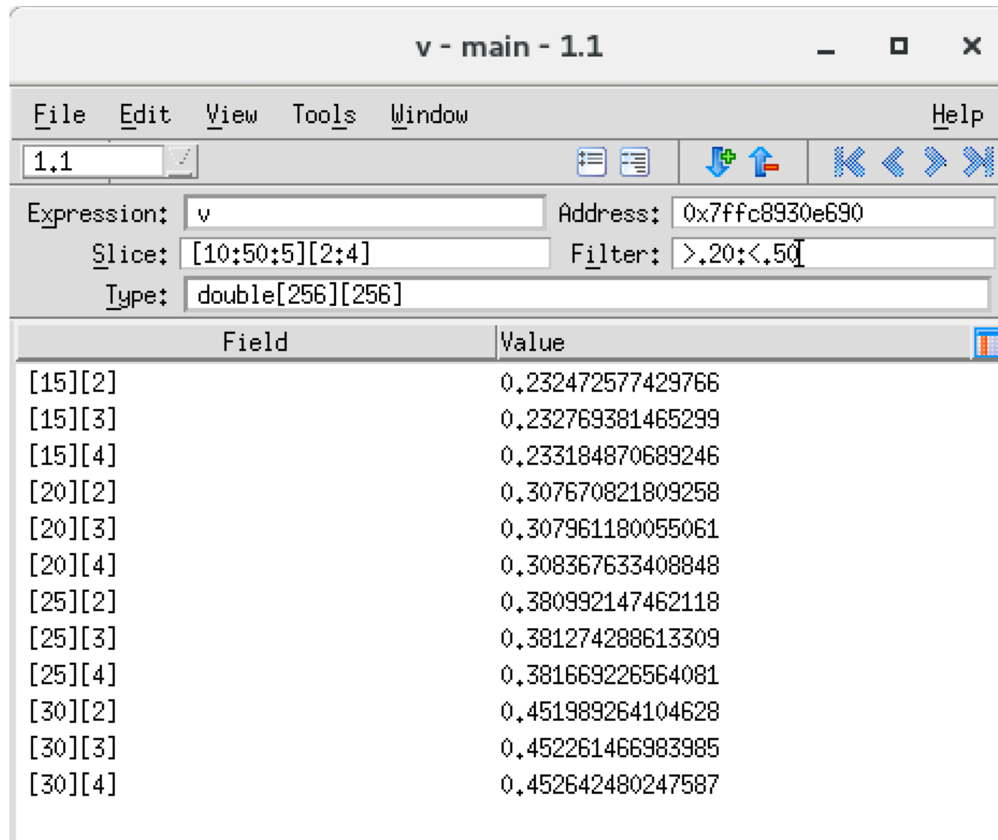


- TotalView transforms many of the C++ and STL containers such as:
  - array, forward_list, tuple, map, set, vector and others.

See This!

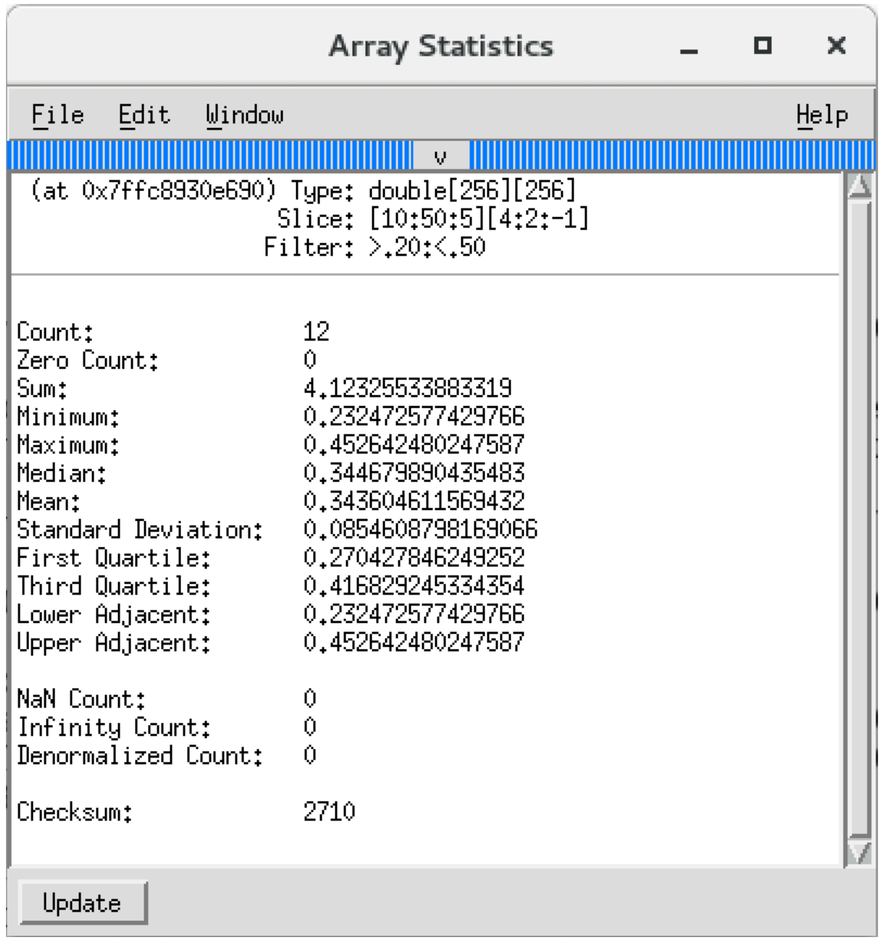
Array Slicing, Striding and Filtering

- Slicing – reduce display to a portion of the array
  - [lower_bound:upper_bound]
  - [5:10]
- Striding – Skip over elements
  - [::stride]
  - [::5], [5:10:-1]
- Filtering
  - Comparison: ==, !=, <, <=, >, >=
  - Range of values: **[>]** *low-value* **:** **[<]** *high-value*
  - IEEE values: $nan, $inf, $denorm



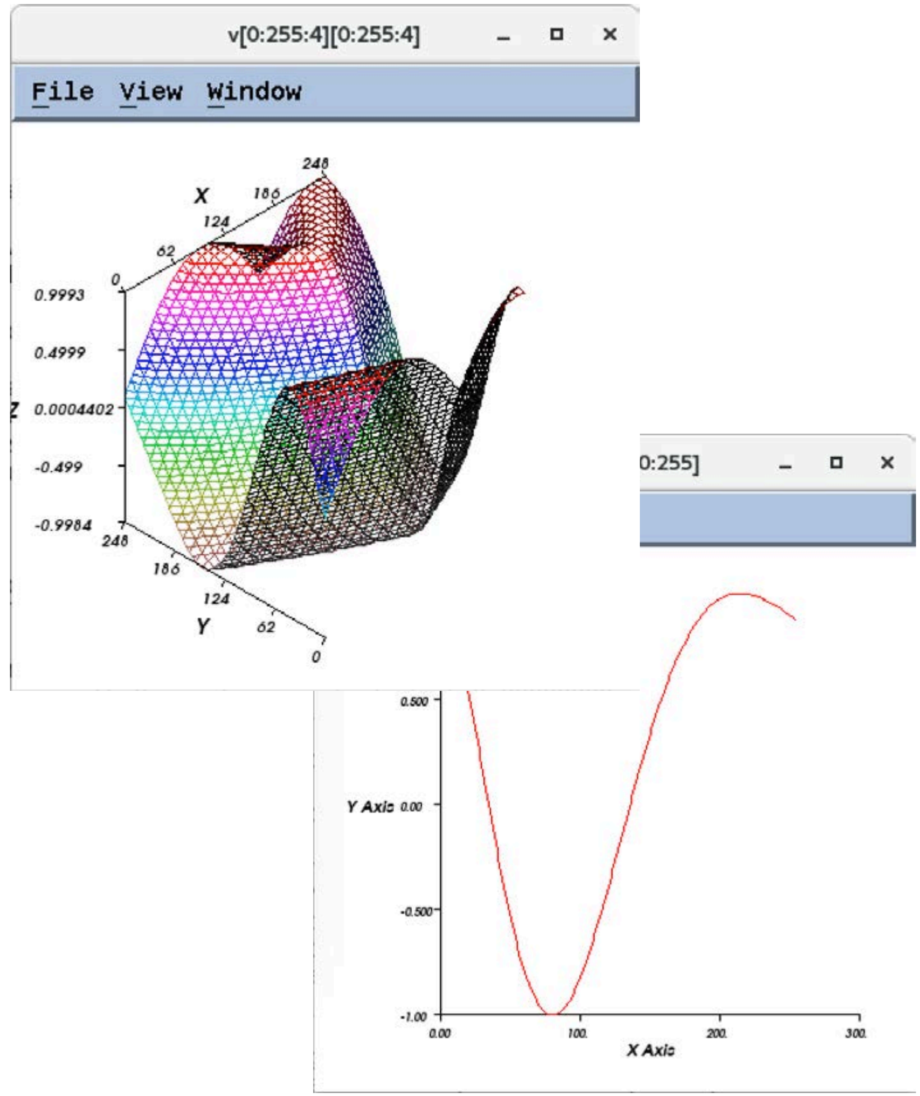
Array Statistics

- Easily display a set of statistics for the filtered portion of your array



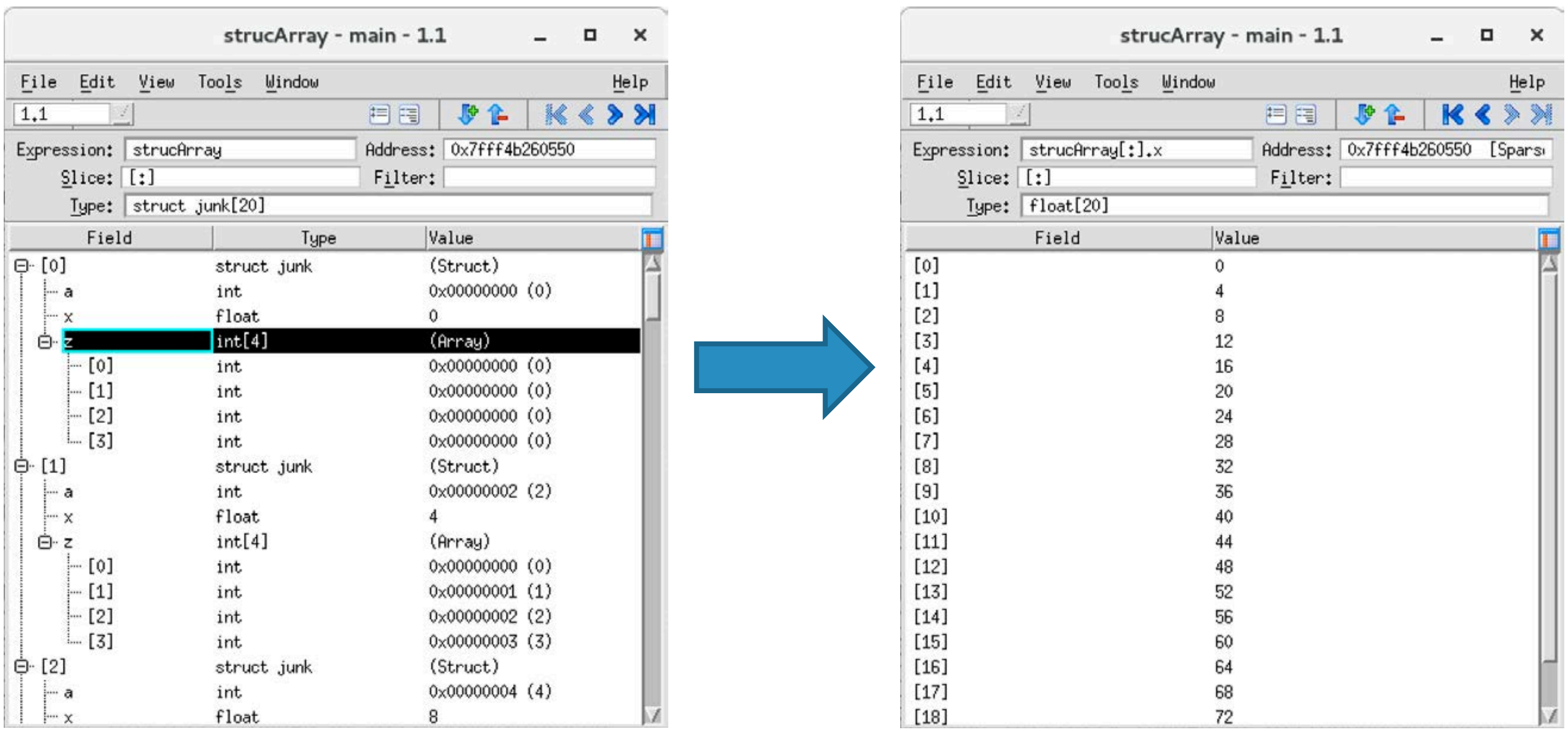
Visualizing Array Data

- Visualizer creates graphic images of your program's array data.
- Visualize one or two dimensional arrays
- View data manually through the Window > Visualize command on the Data Window
- Visualize data programmatically using the $visualize function



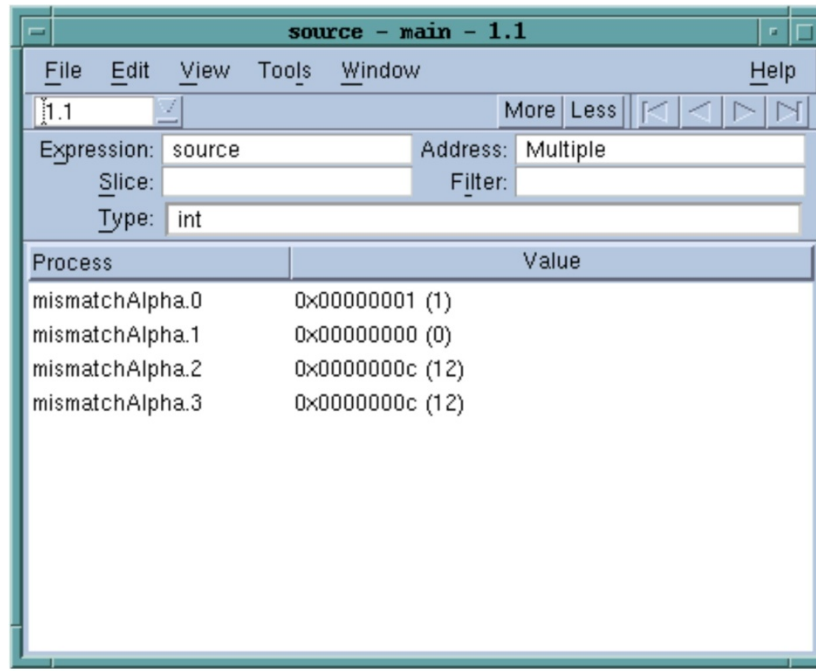
Dive in All

- Dive in All
  - Use Dive in All to easily see each member of a data structure from an array of structures



Looking at Variables Across Processes

- TotalView allows you to look at the value of a variable in all MPI processes
  - Right Click on the variable
  - Select the View > View Across
- TotalView creates an array indexed by process
- You can filter and visualize
- Use for viewing distributed arrays as well.
- You can also View Across Threads



Memory Debugging

Memory Debugging

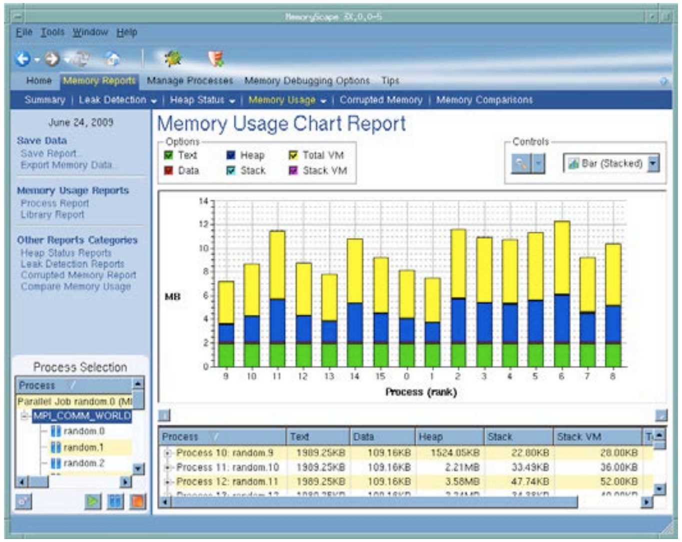
- TotalView's memory debugging technology allows you to
  - Easily find memory leaks and other memory errors
  - Detect malloc/free new/delete API misuse
  - Dangling pointer detection
  - Detect buffer overruns
  - Paint memory blocks on allocation and deallocation
- Memory debugging results can be easily shared as HTML reports or raw memory debugging files.
- Compare memory results between runs to verify elimination of leaks
- Supports parallel applications
- Low overhead and does not require recompilation or instrumentation

TotalView Heap Interposition Agent (HIA)

- Advantages of TotalView HIA Technology
  - Use it with your existing builds
    - No Source Code or Binary Instrumentation
  - Programs run nearly full speed
    - Low performance overhead
  - Low memory overhead
    - Efficient memory usage
  - Support wide range of platforms and compilers

Strategies for Parallel Memory Debugging

- Run the application and see if memory events are detected
- View memory usage across the MPI job
  - Compare memory footprint of the processes
    - Are there any outliers? Are they expected?
- Gather heap information in all processes of the MPI job
  - Select and examine individually
    - Look at the allocation pattern. Does it make sense?
    - Look for leaks
  - Compare with the 'diff' mechanism
    - Are there any major differences? Are they expected?



Unattended/Batch Debugging

Unattended Debugging

- tvscript provides for unattended, straightforward TotalView batch debugging
  - As an alternative to interactive debugging
  - Usable whenever jobs need to be submitted or batched
  - Provides a tool more powerful and flexible than Printf-style debugging
  - Can be used to automate test/verify environments
- Complete documentation in Chapter 4 of the TotalView Reference Guide:
  - http://docs.roguewave.com/totalview/current/html/index.html#page/Reference_Guide%2FBatchDebuggingUsingScripts.html

Think of tvscript as “Printf on steroids”!

Unattended debugging

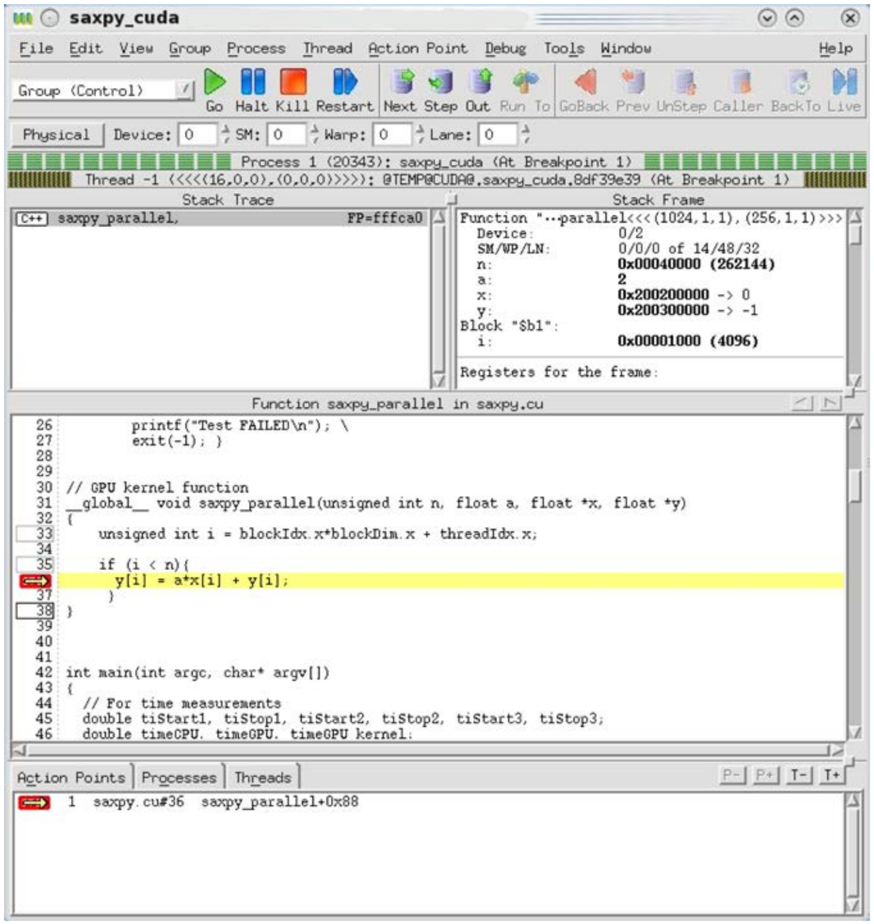
- Command line invocation to run TotalView and MemoryScape unattended
- tvscript can be used to set breakpoints, take actions at those breakpoints and have the results logged to a file. It can also do memory debugging
  - ```
    tvscript –create_actionpoint “method1=>display_backtrace show_arguments” \
          -create_actionpoint “method.c#342=>print x” myprog –a dataset 1
    ```
- memscript can be used to run memory debugging on processes and display data when a memory event takes place. Exit is ALWAYS an event
  - ```
    memscript -event_action
    "alloc_null=list_allocations,any_event=check_guard_blocks” \
     -guard_blocks  -maxruntime "00:30:00”  \
     -display_specifiers "noshow_pc,noshow_block_address,show_image” \
     myProgram -a myProgramArg1
    ```
- Memscript data can be saved in html, memory debug file, text heap status file

Accelerator Debugging

Accelerator Debugging

- High-level debugging support for
  - CUDA
    - Up-to SDK 9.2 and Volta
  - Xeon Phi
    - Native and offloads
  - OpenACC
    - PGI, Cray compilers

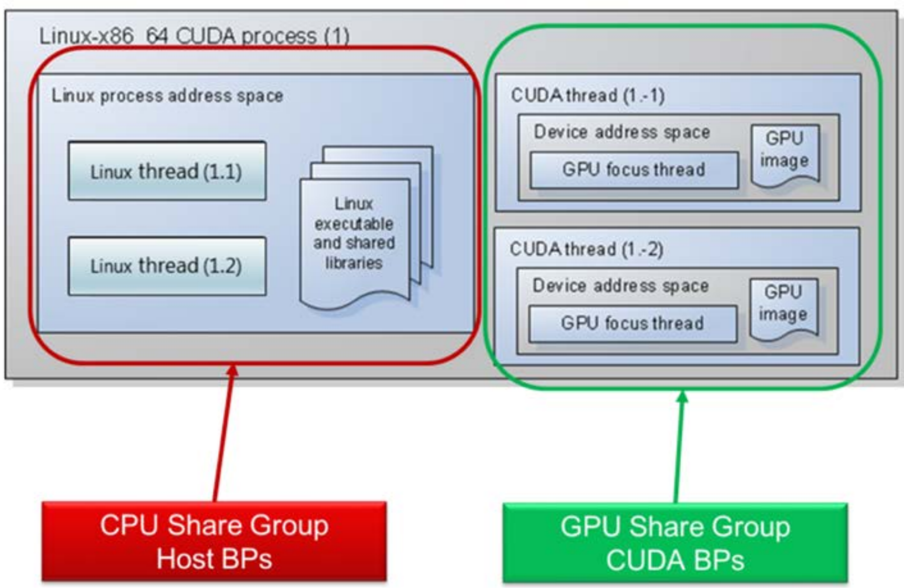
CUDA GPU Debugging with TotalView

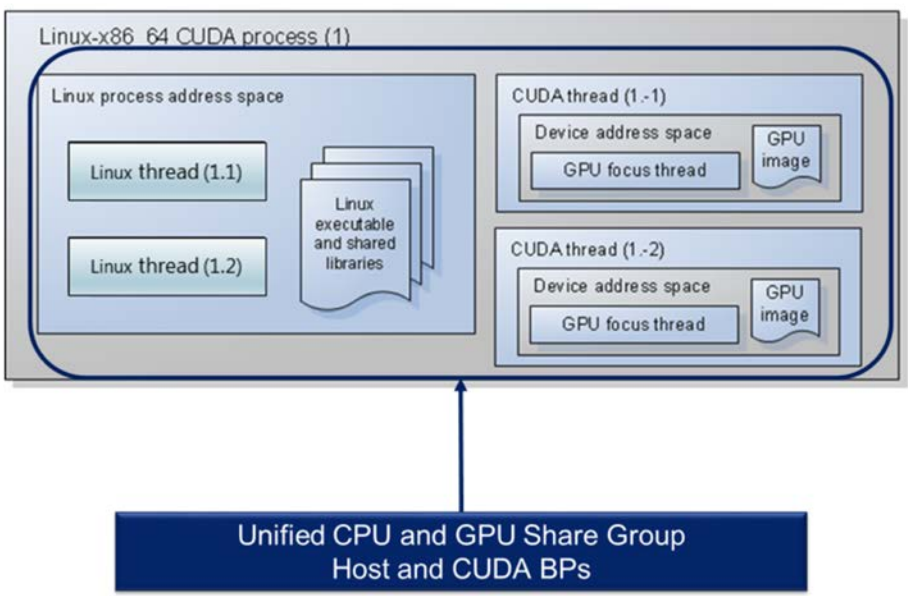


- NVIDIA CUDA 7.5, 8.0, 9.2
- Features and capabilities include
  - Support for dynamic parallelism
  - Support for MPI based clusters and multi-card configurations
  - Flexible Display and Navigation on the CUDA device
    - Physical (device, SM, Warp, Lane)
    - Logical (Grid, Block) tuples
  - CUDA device window reveals what is running where
  - Support for CUDA Core debugging
  - Leverages CUDA memcheck
  - Support for OpenACC

CUDA Debugging Model Improvements

- Coming in TotalView 2018.2 in August
- Set breakpoints in GPU kernel code before it is launched on the GPU
- Unification of source code locations across images (GPU kernel code and shared libraries)
- Improves and streamlines debugging CUDA applications





Intel Xeon Phi Debugging

Supports All Major Intel Xeon Phi Coprocessor Configurations

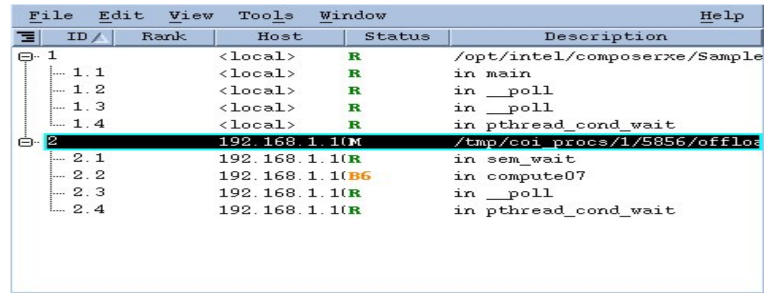
- Native Mode
  - With or without MPI
- Offload Directives
  - Incremental adoption, similar to GPU
- Symmetric Mode
  - Host and Coprocessor
- **Multi-device, Multi-node**
- Clusters
- KNL Support – Just works like a normal node
- Supports AVX2

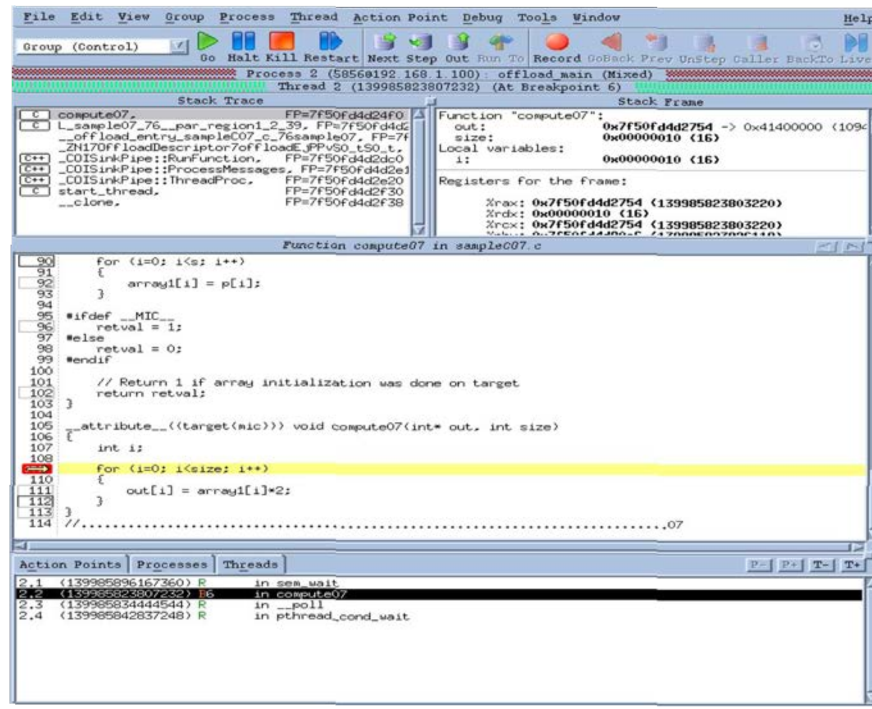
User Interface

- MPI Debugging Features
  - Process Control, View Across, Shared Breakpoints
- Heterogeneous Debugging
  - Debug Both Xeon and Intel Xeon Phi Processes

Memory Debugging

- Both native and symmetric mode





Remote Display Debugging

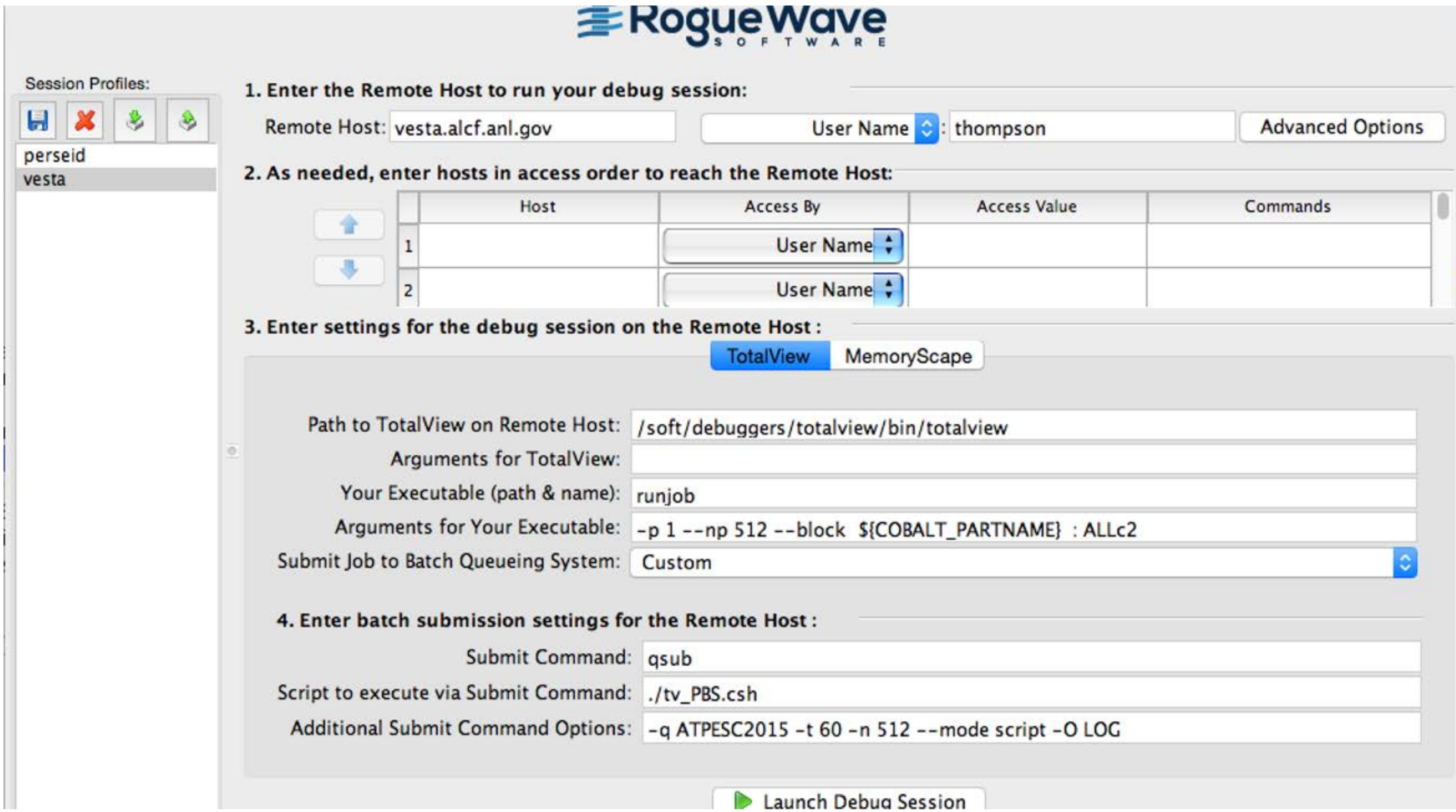
Remote Display Client (RDC)

- Offers users the ability to easily set up and operate a TotalView debug session that is running on another system
- Consists of two components
  - Client – runs on local machine
  - Server – runs on any system supported by TotalView and “invisibly” manages the secure connection between host and client
- Free to install on as many clients as needed
- Remote Display Client is available for:
  - Linux x86, x86-64
  - Windows
  - Mac OS X

Remote Display Client (RDC)

- User must provide information necessary to connect to remote host
- Connection info can be saved for reuse
- Information required includes:
    - User name, public key file, other ssh information
    - Directory where TotalView/MemoryScape is located
    - Path and name of executable to be debugged
    - If using indirect connection with host jump, each host
        - Host name
        - Access type (User name, public key, other ssh information)
        - Access value
- Client also allows for batch submission via PBS Pro or LoadLeveler

Remote Display Client



Using TotalView
for Parallel Debugging

Starting a MPI job – method 1

For HPC we have two methods to start the debugger

The 'classic' method

- **`totalview –args mpiexec –np 512 ./myMPIprog myarg1 myarg2`**
- This will start up TotalView on the parallel starter (mpiexec, srun, runjob, etc) and when you hit 'Go' the job will start up and the processes will be automatically attached. At that point you will see your source and can set breakpoints.

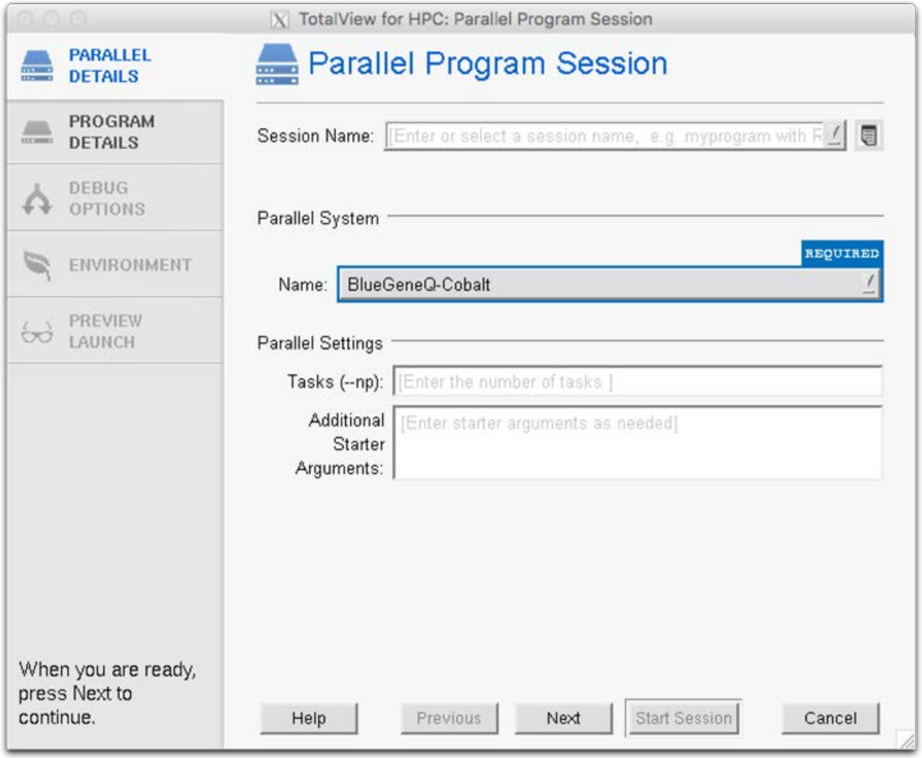
- Some points to consider…
  - You don't see your source at first, since we're 'debugging' the mpi starter
  - Some MPI's don't support the process acquistion method (most do, but might be stripped of symbols we need when packaging)
  - In general more scalable than the next method…

Starting a MPI job – method 2

The ‘indirect’ method

- Simply ‘totalview’ or ‘totalview myMPIprog’ and then you can choose a parallel system, number of tasks, nodes, and arguments to the program.

- With this method the program source is available immediately
- Less dependent on MPI starter symbols
- May not be as scalable as some ‘indirect’ methods launch a debug server per process



Using TotalView at Argonne

- TotalView available on Theta, Vesta, Mira, Cooley
  - Installed at:
    /soft/debuggers/totalview-2018-07-26/toolworks/totalview.2018.1.12/bin

- Memory Debugging on BG\Q and Cray should link against the agent, either static or dynamically
  - BG/Q:
    - -L<path> –Wl,@<path>/tvheap_bgqs.ld #static
    - -L<path> -ltvheap_64 –Wl,-rpath,<path> #dynamic
  - Cray:
    - -L<path> -ltvheap_cnl # static
    - -L<path> -ltvheap_cnl –Wl,-rpath,<path> #dynamic
  - <path> = Path to platform specific TV lib
    - export TVLIB=/soft/debuggers/totalview-2018-07-26/toolworks/totalview.2018.1.12/linux-x86-64/lib
      - Substitute linux-power on BlueGene

Job Control at Argonne

- TotalView can be run on simple serial programs on login nodes (though maybe not the preferred method)
- MPI jobs require an allocation, either an interactive session (qsub –I) or through a batch script that creates an interactive session.
- tvscript and memscript can be run totally in batch.

TotalView Resources and Documentation

TotalView Resources & Documentation

- TotalView documentation:
  - https://support.roguewave.com/documentation/tvdocs/en/current/
  - User Guides: Debugging, Memory Debugging and Reverse Debugging
  - Reference Guides: Using the CLI, Transformations, Running TotalView
- TotalView online HTML doc:
  - http://docs.roguewave.com/totalview/current/html/index.html
- Other Resources (Blogs, videos, white papers, etc):
  - https://www.roguewave.com/resources?tagid=18
- New UI resources:
  - Reference CodeDynamics Help
    https://www.roguewave.com/help-support/documentation/codedynamics
- New UI videos:
  - https://www.roguewave.com/products-services/codedynamics/videos
- Python Debugging blog:
  - http://blog.klocwork.com/dynamic-analysis/the-challenge-debugging-python-and-cc-applications/

Questions/Comments

Questions/Comments

- Any questions or comments?
  - Don’t hesitate to reach out to me directly with any problems or suggestions!
  - **Email:** bill.burns@roguewave.com

- **Thank you for your time today!**

RogueWave
SOFTWARE

Innovate with Confidence