ATPESC 2017

TotalView: Debugging from Desktop to Supercomputer

Peter Thompson
Principal Software Support Engineer
August 8, 2017

RogueWave SOFTWARE
Innovate with Confidence

Some thoughts on debugging

- As soon as we started programming, we found out to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.
  - Maurice Wilkes
- Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.
  - Brian W. Kernigan
- Sometimes it pays to stay in bed on Monday, rather than spending the rest of the week debugging Monday's code.
  - Dan Saloman

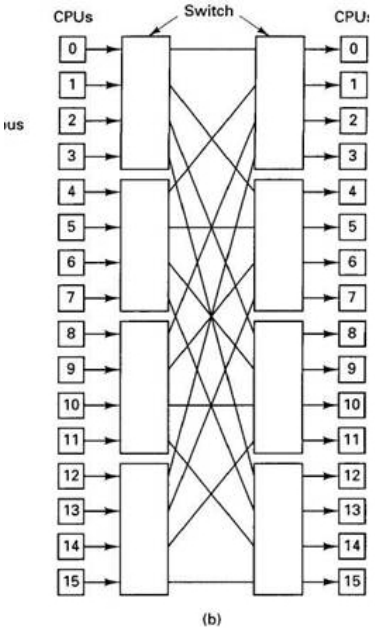
Rogue Wave's Debugging Tool

TotalView for HPC

- Source code debugger for C/C++/Fortran
  - Visibility into applications
  - Control over applications
- Scalability
- Usability
- Support for HPC platforms and languages

TotalView Overview

TotalView Origins



Mid-1980's Bolt, Berenak, and Newman (BBN) Butterfly Machine
An early 'Massively Parallel' computer

How do you debug a Butterfly?

- TotalView project was developed as a solution for this environment
  - Able to debug multiple processes and threads
  - Point and click interface
  - Multiple and Mixed Language Support
- Core development group has been there from the beginning and have been/are involved in defining MPI interfaces, DWARF, and lately OMPD (Open MP debugging interface)

Other capabilities added

- Support for most types of MPI
- Lightweight Memory Debugging
- Type transformations – STL and user containers
- Memscript and tvscript
- Reverse Debugging - only on Linux x86-64
- Remote Display Client
- GPU debugging
- Intel Xeon Phi – Including KNL
- Most popular platforms, Linux, Mac, Solaris, AIX… but not Windows
- ARM64
- Python Debugging support – currently in progress

Key TotalView Features

- Multi-process and Multi-thread debugging
- Interactive Memory Debugging
- Reverse Debugging
- Unattended Debugging
- Remote Display Client
- CUDA Debugging
- Xeon Phi Debugging

Serial, Parallel and Accelerated applications

Multi-process and Multi-thread Debugging

- Supports/Supported by most MPI flavors
  - Automatic process acquisition across nodes with lightweight debug servers in an MRNet tree configuration
  - Can attach to a running MPI job
- Support for OpenMP and pthreads
  - Ability to hold and control individual threads
- Mixed Multi-process and Multi-threaded programs
- Breakpoint control on the Group, process and thread level

TotalView's Memory Efficiency

- TotalView is lightweight in the back-end (server)
- Servers don't "steal" memory from the application
- Each server is a multi-process debugger agent
  - One server can debug thousands of processes
  - Not a conglomeration of single process debuggers
  - TotalView's architecture provides flexibility (e.g., P/SVR)
  - No artificial limits to accommodate the debugger (e.g., BG/Q ~~1 P/CN~~)
- Symbols are read, stored, and shared in the front-end (client)
- Example: LLNL APP ADB, 920 shlibs, Linux, 64 P, 4 CN, 16 P/CN, 1 SVR/CN

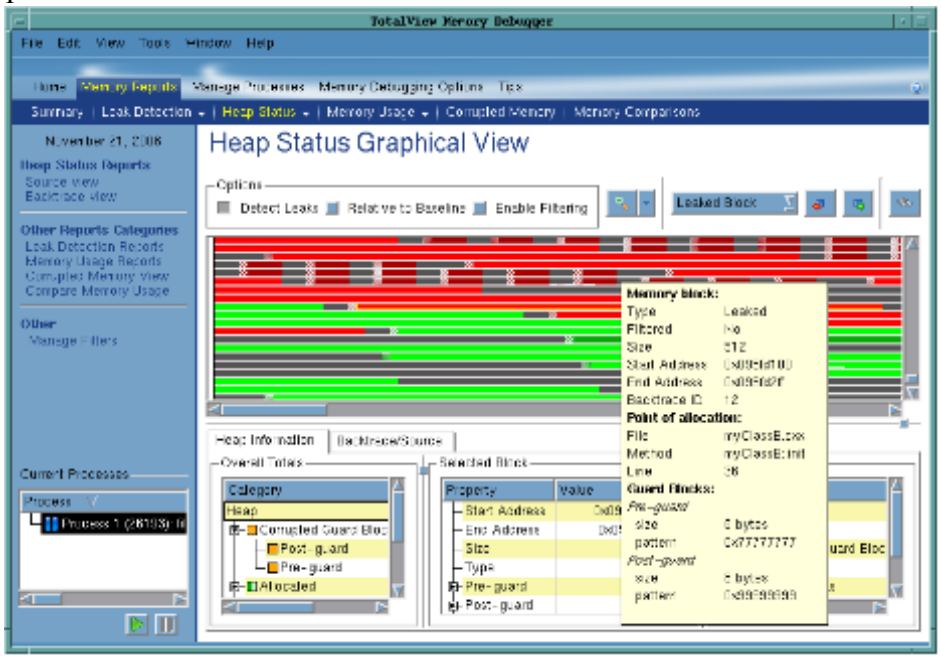
Process	VSZ (largest, MB)	RSS (largest, MB)
TV Client	4,469	3,998
MRNet CP	497	4
TV Server	304	53

Memory Debugging

How do you find buffer overflows or memory leaks?

Runtime Memory Analysis : Eliminate Memory Errors

- Detects memory leaks *before* they are a problem
- Explore heap memory usage

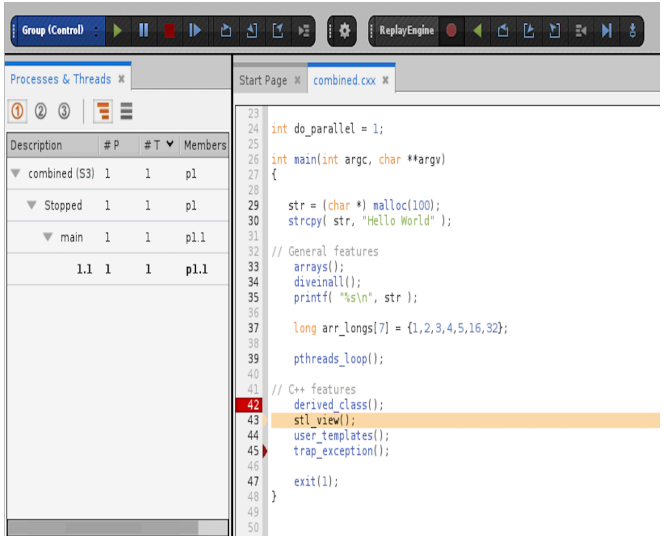


Features

- Detects
  - Malloc API misuse
  - Memory leaks
  - Buffer overflows
- Low runtime overhead
- Easy to use
  - Works with vendor libraries
  - No recompilation
  - No instrumentation
  - Link against HIA for MPI

Reverse debugging

- How do you isolate an intermittent failure?
  - Without TotalView,
    - Set a breakpoint in code
    - Realize you ran past the problem
    - Re-load
    - Set breakpoint earlier
    - Hope it fails
    - Keep repeating
  - With TotalView
    - Start recording
    - Set a breakpoint
    - See failure
    - Run backwards/forwards in context of failing execution



  - Reverse Debugging
    - Re-creates the context when going backwards
    - Focus down to a specific problem area easily
    - Saves days in recreating a failure

Unattended Debugging

Memscript and Tvscript

- 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

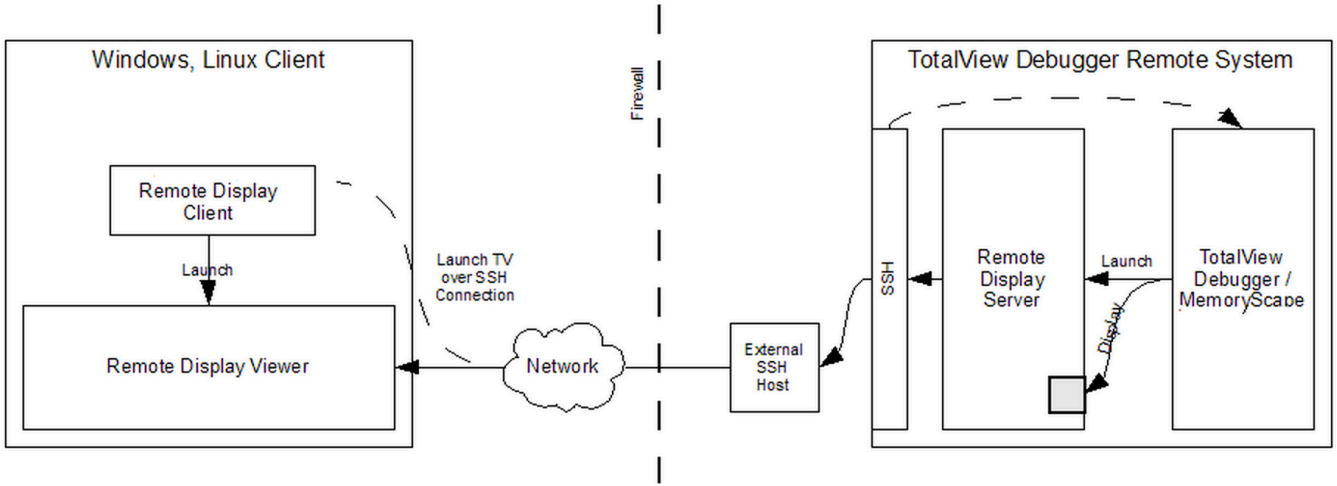
```
Memscrip -event_action \ "alloc_null=list_allocations,any_event=check_guard_blocks” \
-guard_blocks -maxruntime "00:30:00” -display_specifiers \ "noshow_pc,noshow_block_address,show_image”\
myProgram -a myProgramArg1
```

- Memscript data can be saved in html, memory debug file, text heap status file

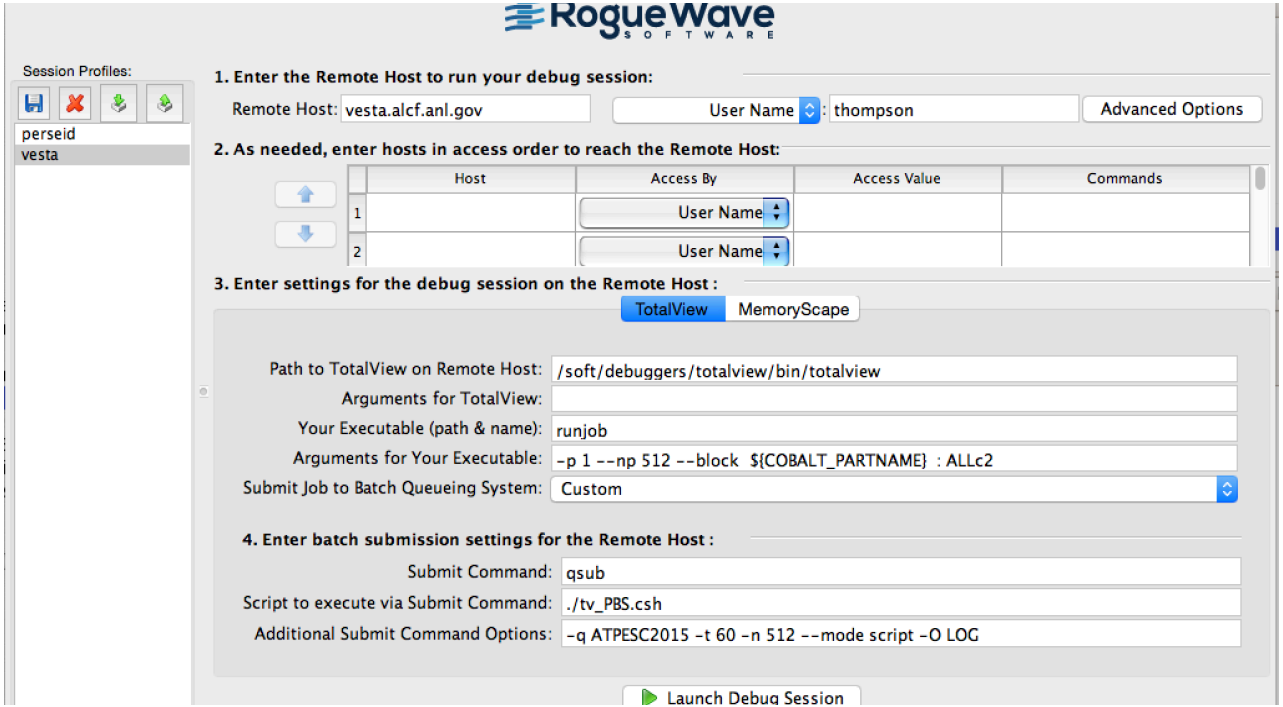
Remote Display Client (RDC)

- Push X11 bits and events across wide networks can be painful. The RDC can help

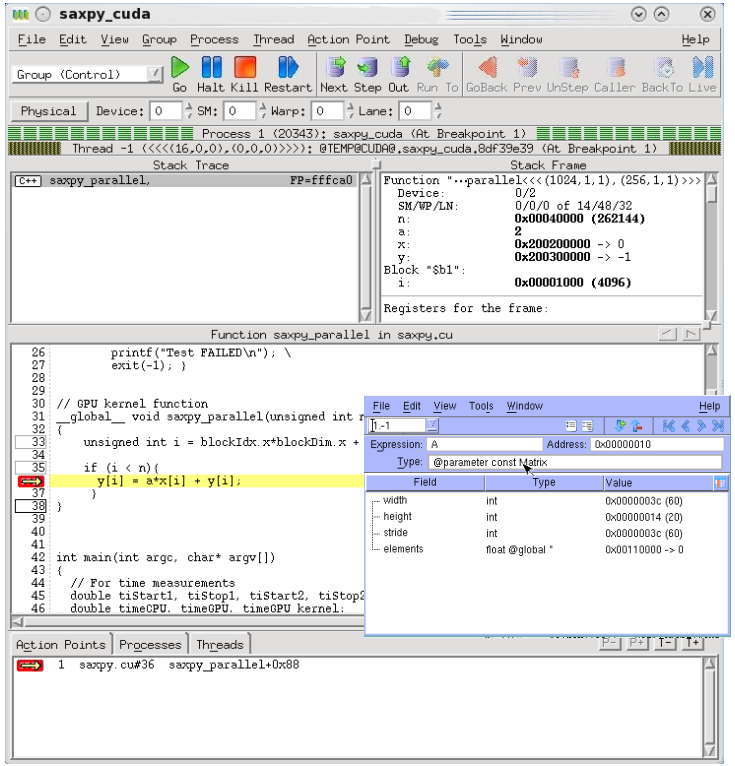
Figure 17 – Remote Display Components



The RDC setup



TotalView for the NVIDIA ® GPU Accelerator



- NVIDIA CUDA 6.5, 7.0, 7.5, 8.0 – (testing 9.0)
- 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

TotalView for the Intel® Xeon Phi™ coprocessor

Supports All Major Intel Xeon Phi Coprocessor Configurations

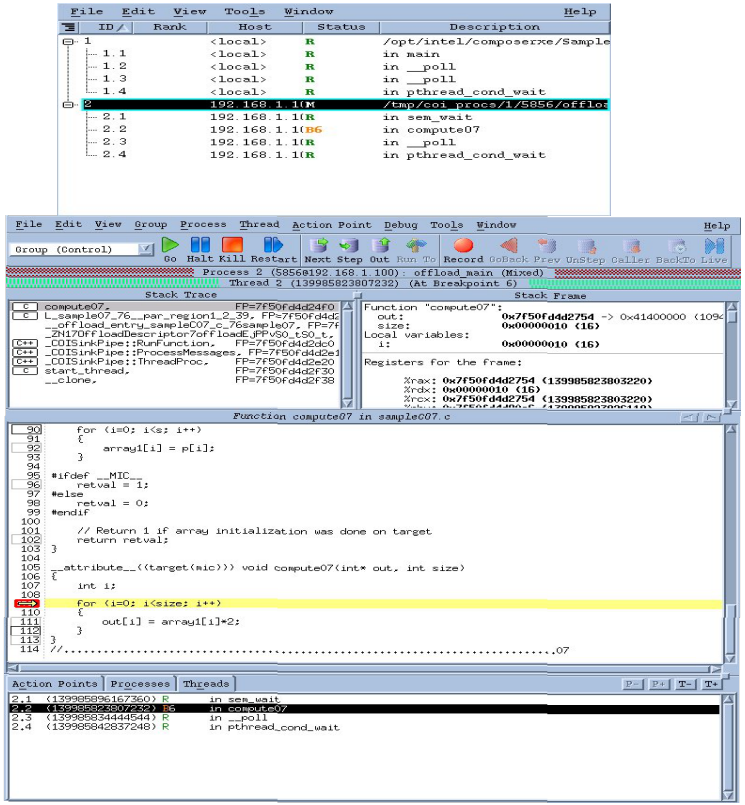
- Native Mode
  - With or without MPI
- Offload Directives
  - Incremental adoption, similar to GPU
- Symmetric Mode
  - Host and Coprocessor
- **Multi-device, Multi-node**
- Clusters
- KNL Support – Just works like a normal node
  - AVX2 support being added

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



Knights Landing Memory

- KNL has on-board high bandwidth memory (MCDRAM) which can be accessed much faster than going out to main memory.
  - Cache
  - Explicitly managed for placement of frequently accessed data
- MemoryScape will be able to track allocations made both the standard heap and the on-chip HBM
- Optimization may include making sure that the right data structures are available to the processor in HBM
  - MemoryScape can show you data structure usage and placement
- KNL machines online - right here! Let's test this…

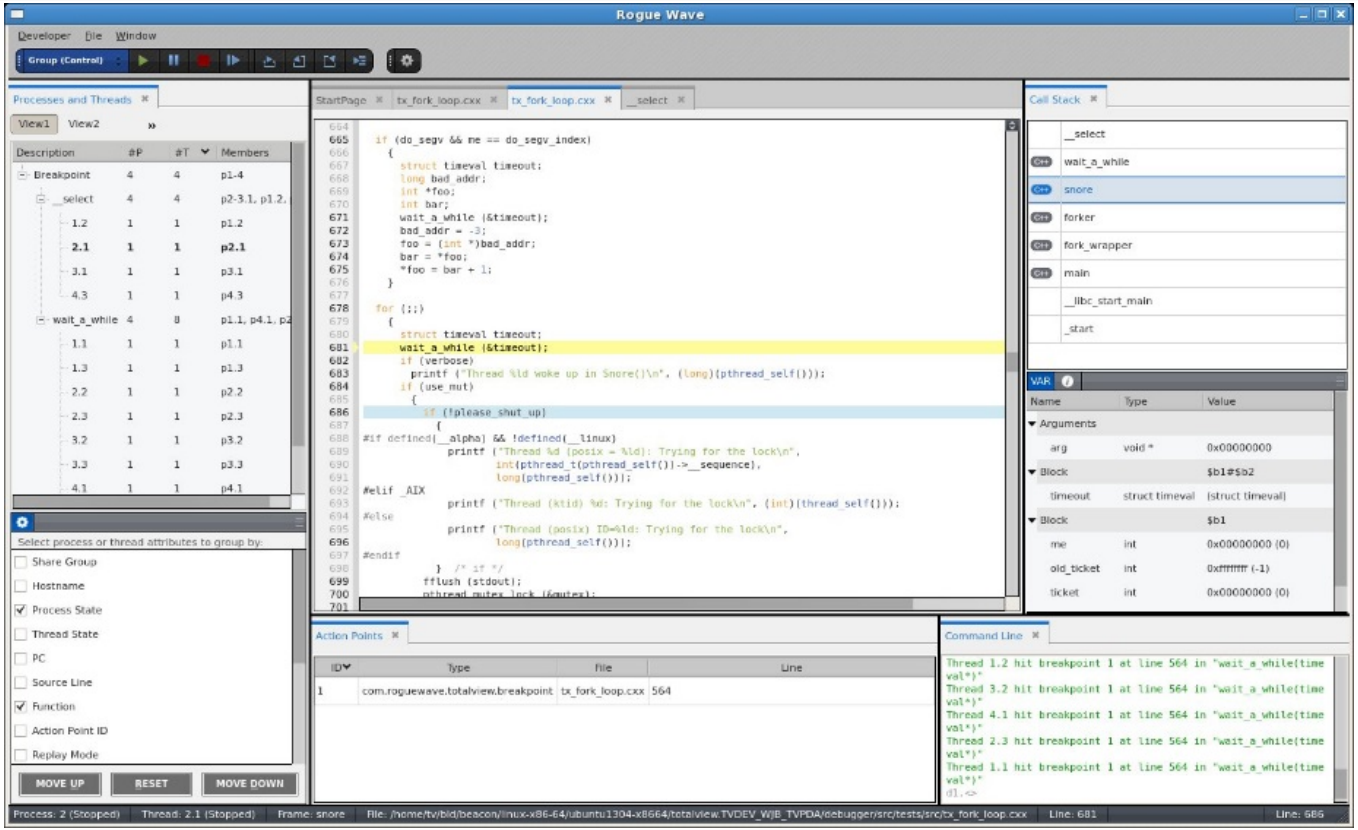
TotalView – Next Generation

What's New?

Linux OpenPower (LE) support with GPU

- Support for OpenPower (Linux power LE)
  - All major functionality
  - Support for CUDA Debugging on GPU Accelerators
- Currently working with IBM and Lawrence Livermore to support the CORAL systems (Power 8 nodes with 4 Nvidia PASCAL cards)

New UI Framework – aka CodeDynamics



Python Support

- Recently added to add in debugging mixed language programs
  - Still in development stages, but a good start

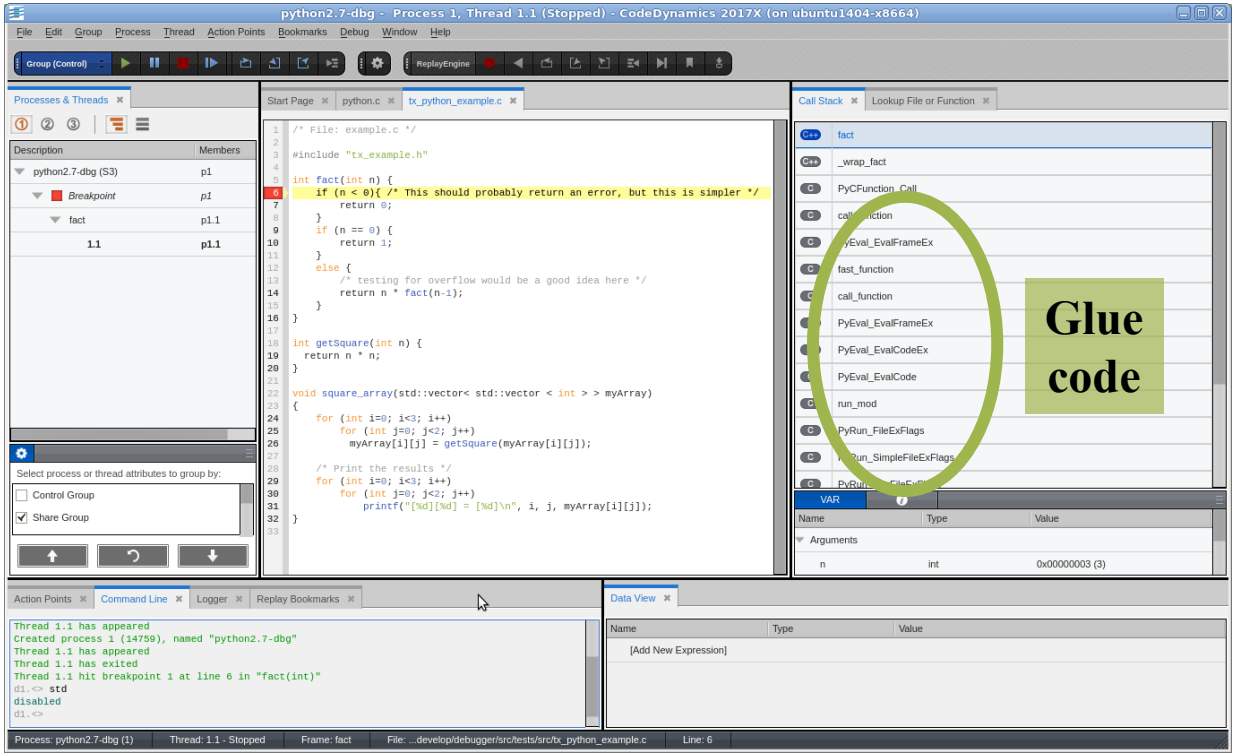
Calling C/C++ from Python

- Legacy libraries are written in C/C++ and Fortran
  - Run faster
  - Rewriting doesn't make sense
- Luckily there are many ways to call between the languages

Python C/C++ glue technology	Description
ctypes	A foreign function library for Python.
Cython	A superset of the Python language that additionally supports calling C functions and declaring C types on variables and class attributes.
SWIG	A software development tool that connects programs written in C and C++ with a variety of high-level programming languages including Python.
CFFI	Foreign Function Interface for Python calling C code.
PyQt/PySide and SIP	SIP is a tool that makes it easy to create Python bindings for C and C++ libraries.
Boost.Python	A C++ library which enables seamless interoperability between C++ and the Python programming language.

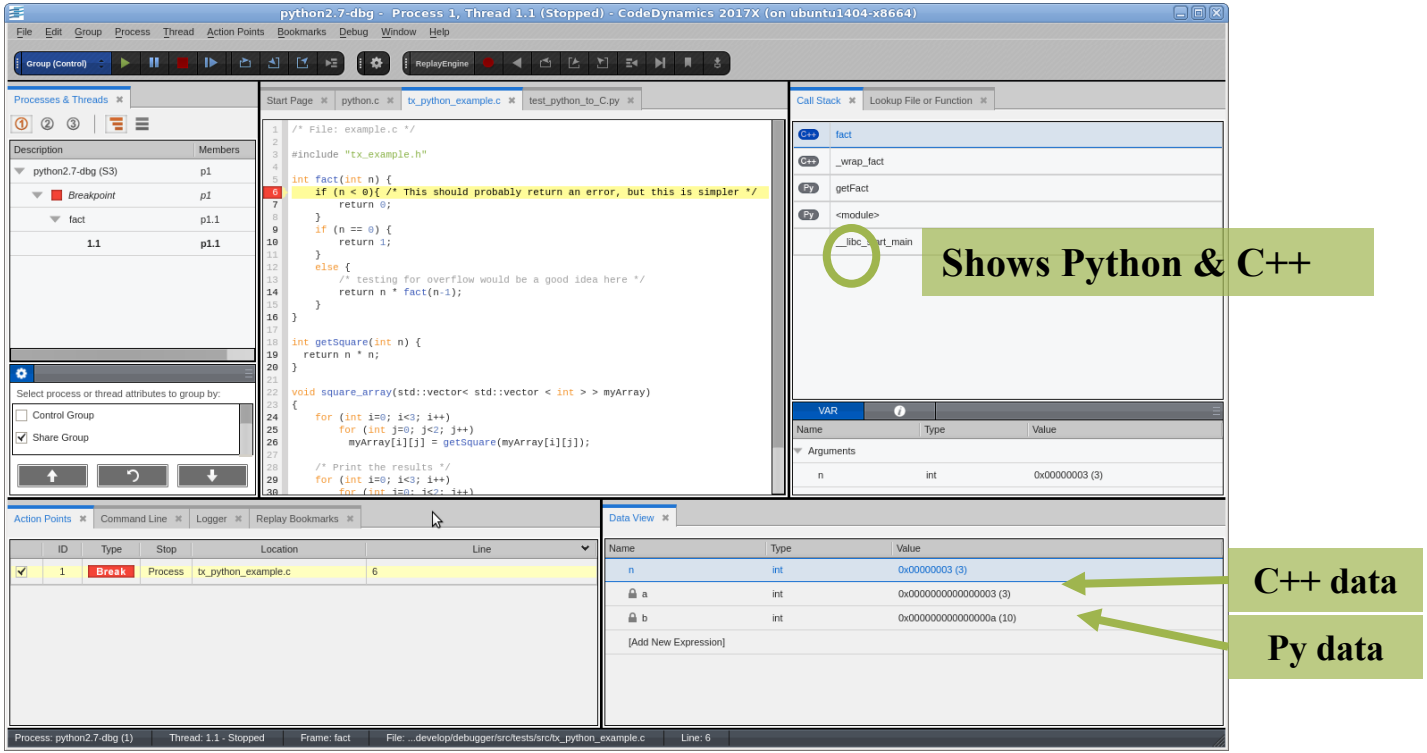
Python without Filtering

No viewing of Python data & code



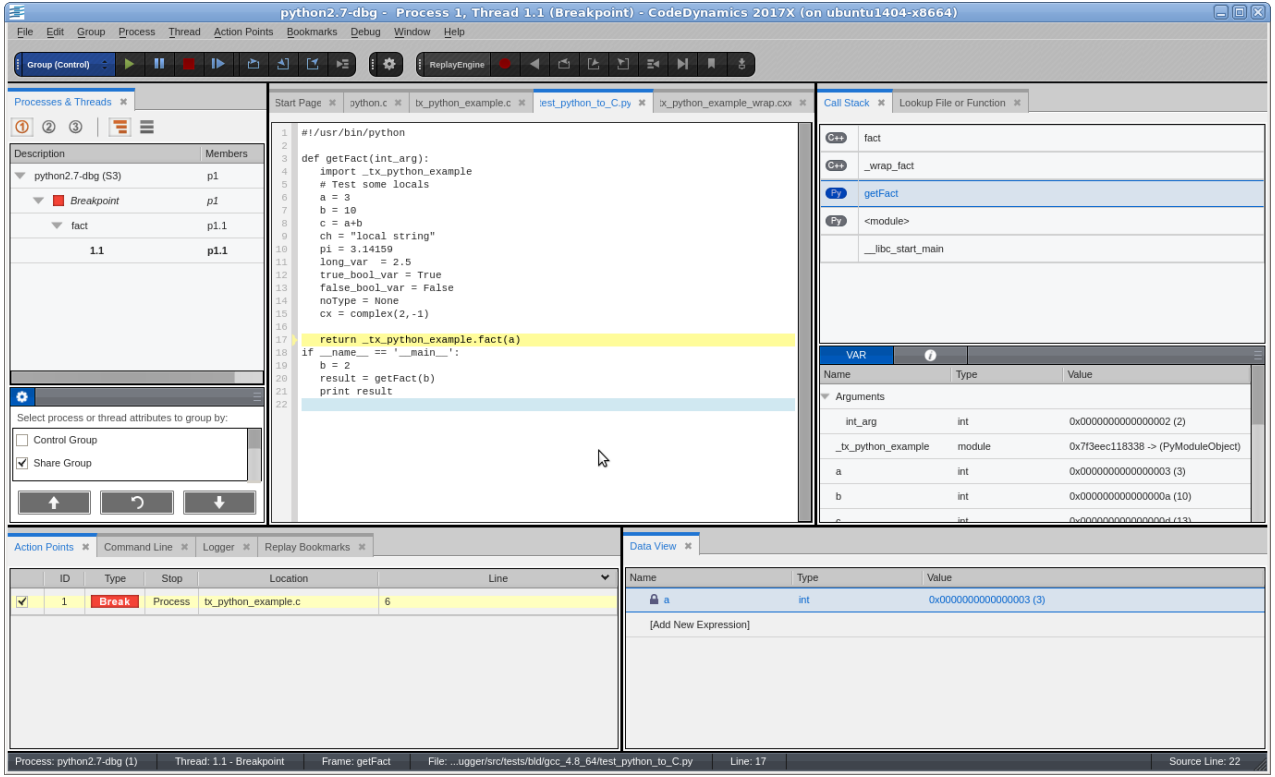
Showing C code with mixed data

Glue code filtered out - Python data available for viewing



Python with filtering

Python code available - Program counter shows calling location



Debug Fission – Split Dwarf Support

Debug Information takes up a lot of Space

- Line and symbol information generally represented in DWARF format
  - Allows us to show the source code and locate variables
  - The larger and more complex the code, the more data is needed to represent it. This can grow to GB's in size
- DebugFission SplitDwarf, gdb_index, dwz methods of dealing with this are now all supported.

Using TotalView

Using TotalView

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.

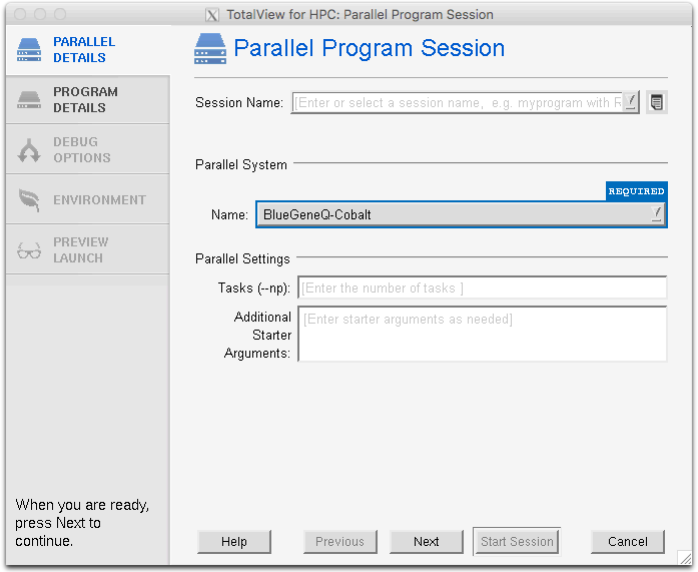
- Some points to consider…
  - You don’t see your source at first, since we’re ‘debugging’ the mpi starter
  - Some MPI’s don’t support the process acquistion method (most do, but might be stripped of symbols we need when packaging)
  - In general more scalable than the next method...

Starting TotalView

The ‘indirect’ method

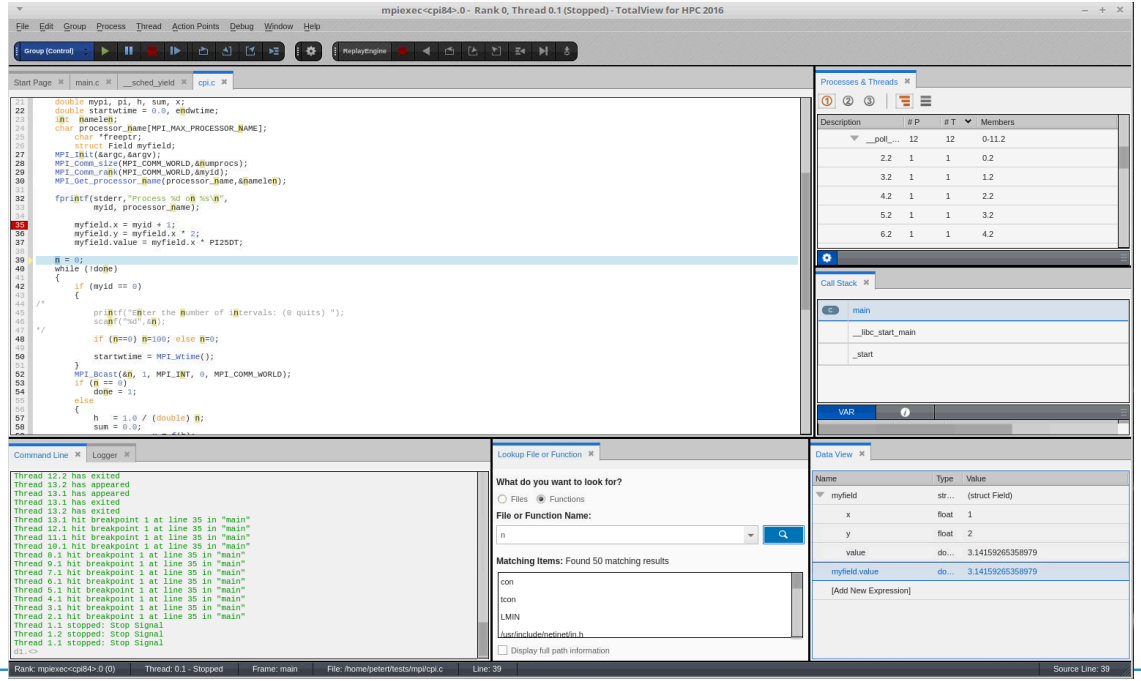
- Simply ‘totalview’ or ‘totalview myMPIprog’ and then you can choose a parallel system, number of tasks, nodes, and arguments to the program.

- 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



The New UI for HPC

- MPI debugging with the new UI requires starting in ‘classic’ mode with the –newUI argument
  - totalview –newUI –args mpiexec –np 4 ./cpi
- Python debugging support stack transform only in newUI



Using TotalView at Argonne

- Modules available on Theta, Vesta, Mira
  - module load totalview
- 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-2017-07-26/toolworks/totalview.2017.2.10/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.
- Examples will be provided (After I confirm they work!)

And that’s all…

- See me for demos of particular features or to try TotalView on your code

Our products and services

Tools

Klocwork On-the-fly static code analysis for app security

CodeDynamics Commercial dynamic analysis

OpenLogic Support Enterprise-grade SLA support

OpenLogic Audits Detailed open source license and security risk guidance

TotalView for HPC Scalable debugging

Zend Server Enterprise PHP app server

Zend Studio PHP IDE

Zend Guard PHP encoding and obfuscation

Libraries

SourcePro OS, database, network, and analysis abstraction for C++

Visualization Real-time data visualization at scale

PV-WAVE Visual data analysis

IMSL Numerical Libraries Scalable math and statistics algorithms

HydraExpress SOA/C++ modernization framework

HostAccess Terminal emulation for Windows

Stingray MFC GUI components

Rogue Wave SOFTWARE

Innovate with Confidence