TotalView by Perforce

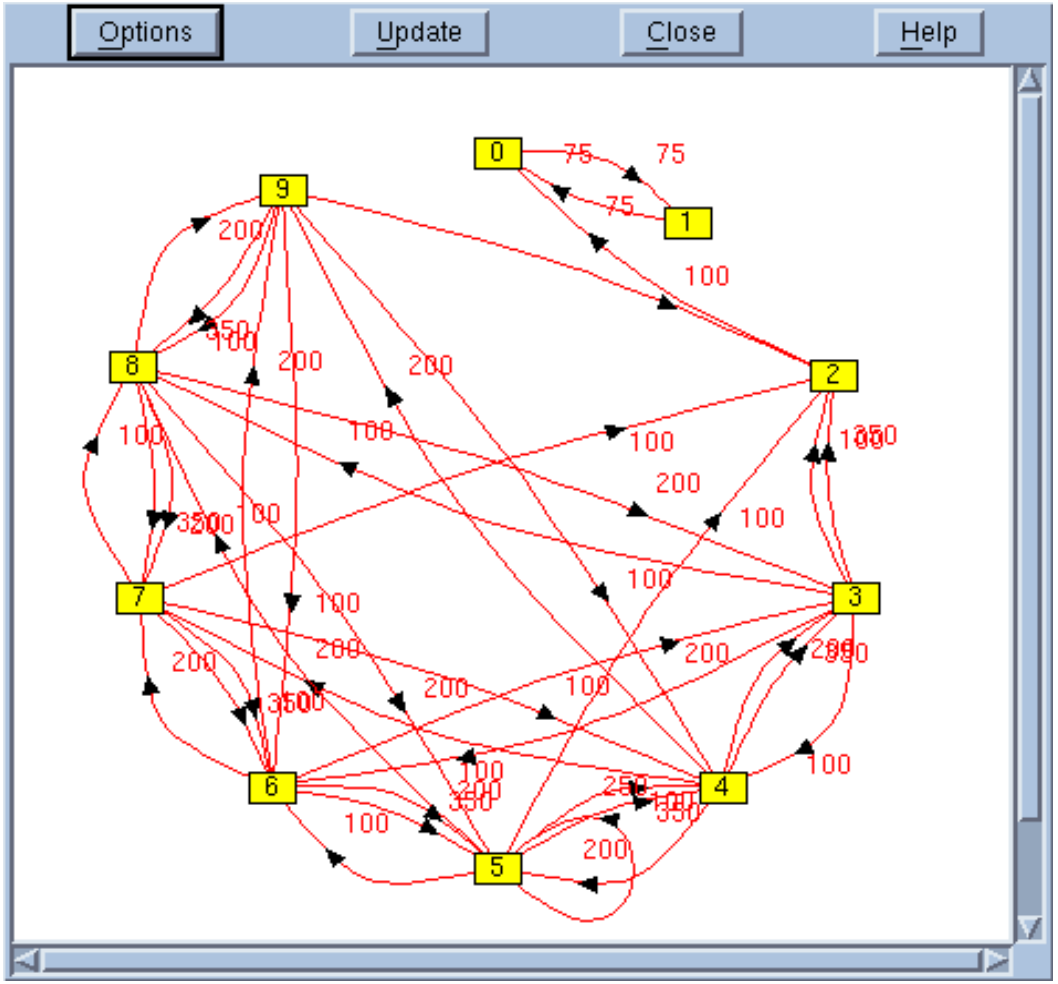
Techniques for Debugging HPC Applications

NIKOLAY PISKUN , DIRECTOR OF CONTINUING ENGINEERING, TOTALVIEW PRODUCTS

AUGUST 5 2020, ATRESC 2020

Agenda

- What is debugging and why TotalView?
- Introduction to TotalView by David Falkenstein
- Introduction to MPI debugging by Dean Stuart
- Reverse and Memory debugging
- GPU debugging
- Python/C++ debugging
- Reverse Connections by Dean Stuart
- Using TotalView on ANL
- TotalView resources and documentation
- Questions/Comments



What is Debugging and
Why do you need TotalView?

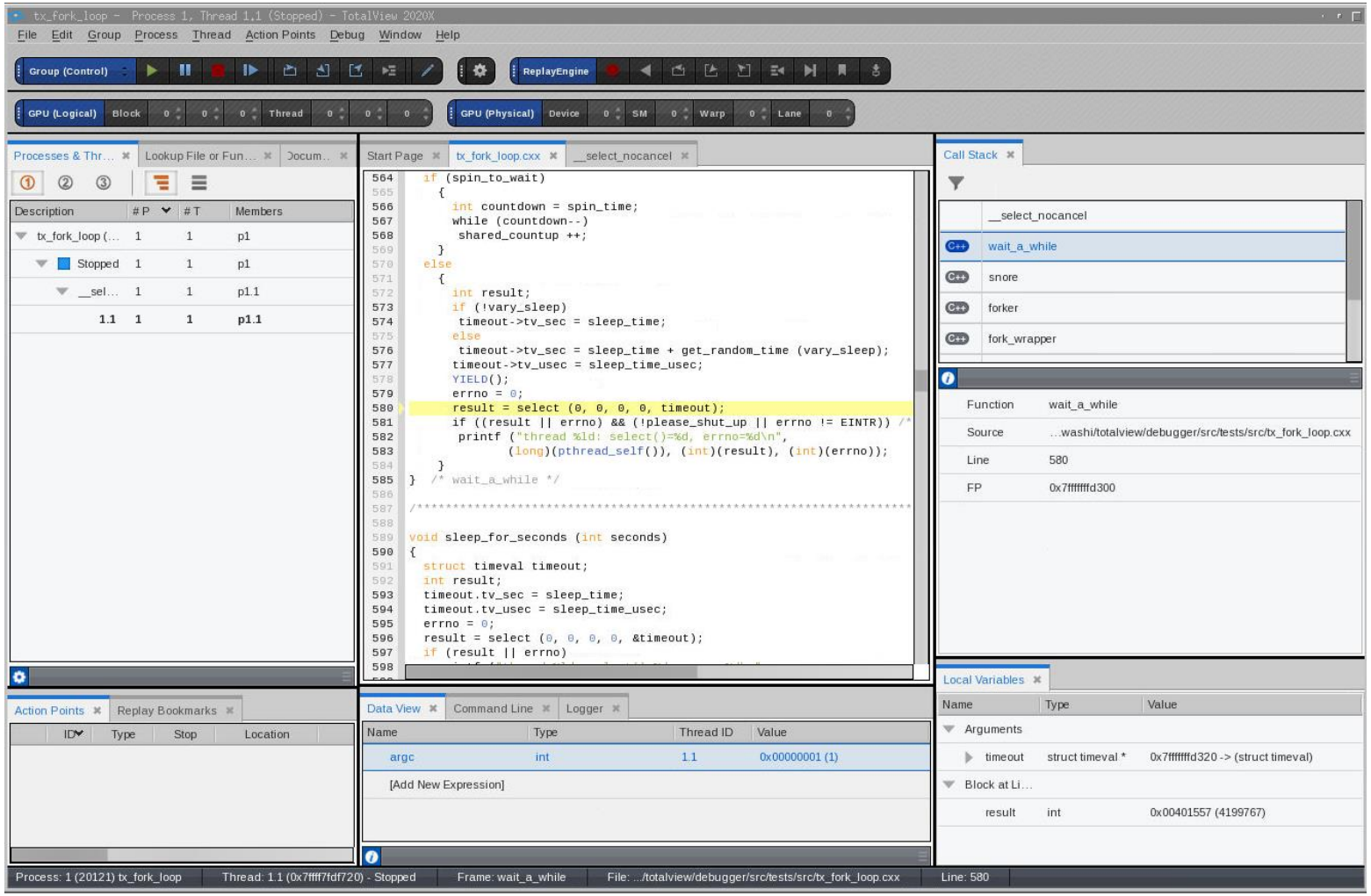
What is Debugging?

- Debugging is the process of finding and resolving defects or problems within a computer program or a system.
  - Algorithm correctness
  - Data correctness
  - Scaling/Porting correctness

TotalView for HPC and for All

- Leading debug environment for HPC users
  - Active development for 30+ years
  - Thread specific breakpoints
  - Control individual thread execution
  - View complex data types easily
  - From **MacBook** to **Top500** Supercomputers
- Track memory leaks in running applications
- Supports C/C++ and Fortran on Linux/Unix/Mac
- Support debugging mixed Python/C++
- Integrated Reverse debugging
- Batch non-interactive debugging.

- Allowing YOU to have
  - Predictable development schedules
  - Less time spent debugging



Introduction to TotalView User Interface

TotalView debugger enables you to do:

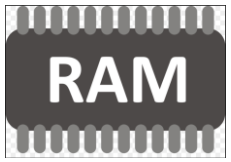
- **Interactive debugging**
  - Live control of an executing program

- **Remote debugging**

  - Debug a program running on another computer

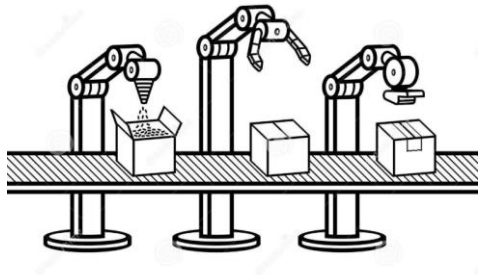
- **Post-mortem debugging (core files and reverse debugging)**
  - Debugging a program after it has crashed or exited

- **Memory debugging**



  - Find memory management problems (leaks, corruption …)
  - Comparing results between executions

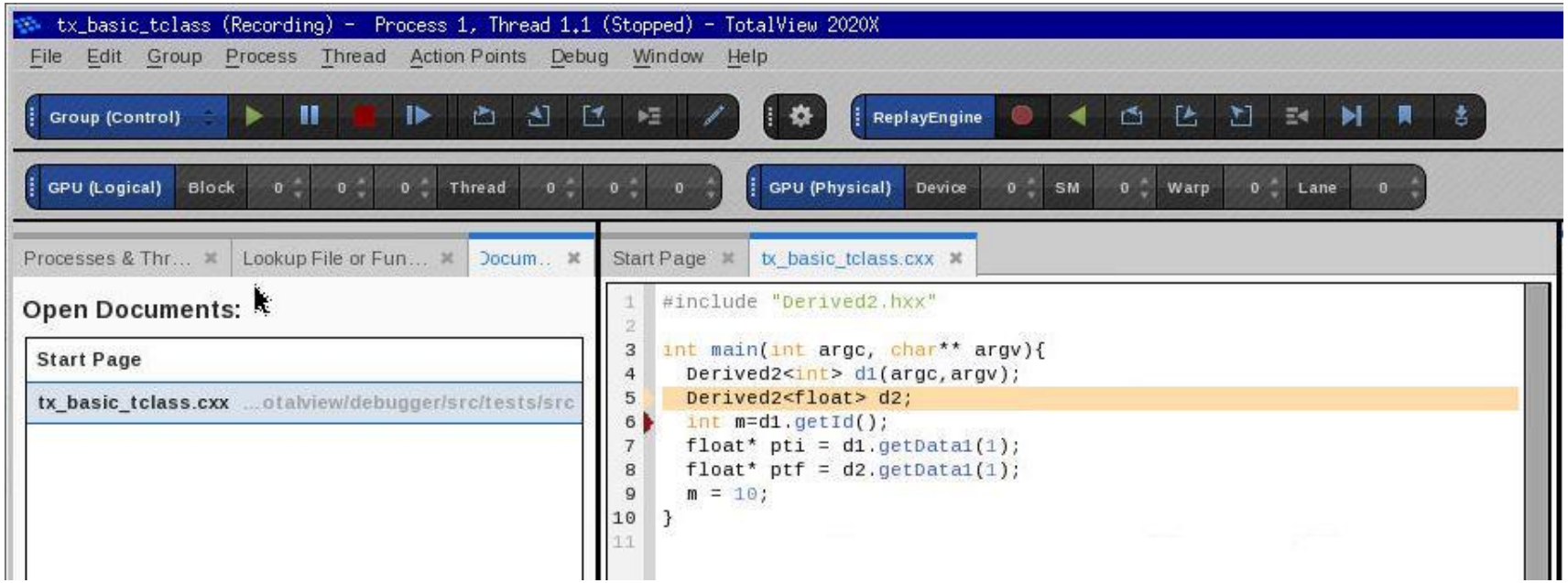
- **Batch debugging (tvscript, CI environments)**
  - Unattended debugging



Introduction to MPI debugging

Replay Engine

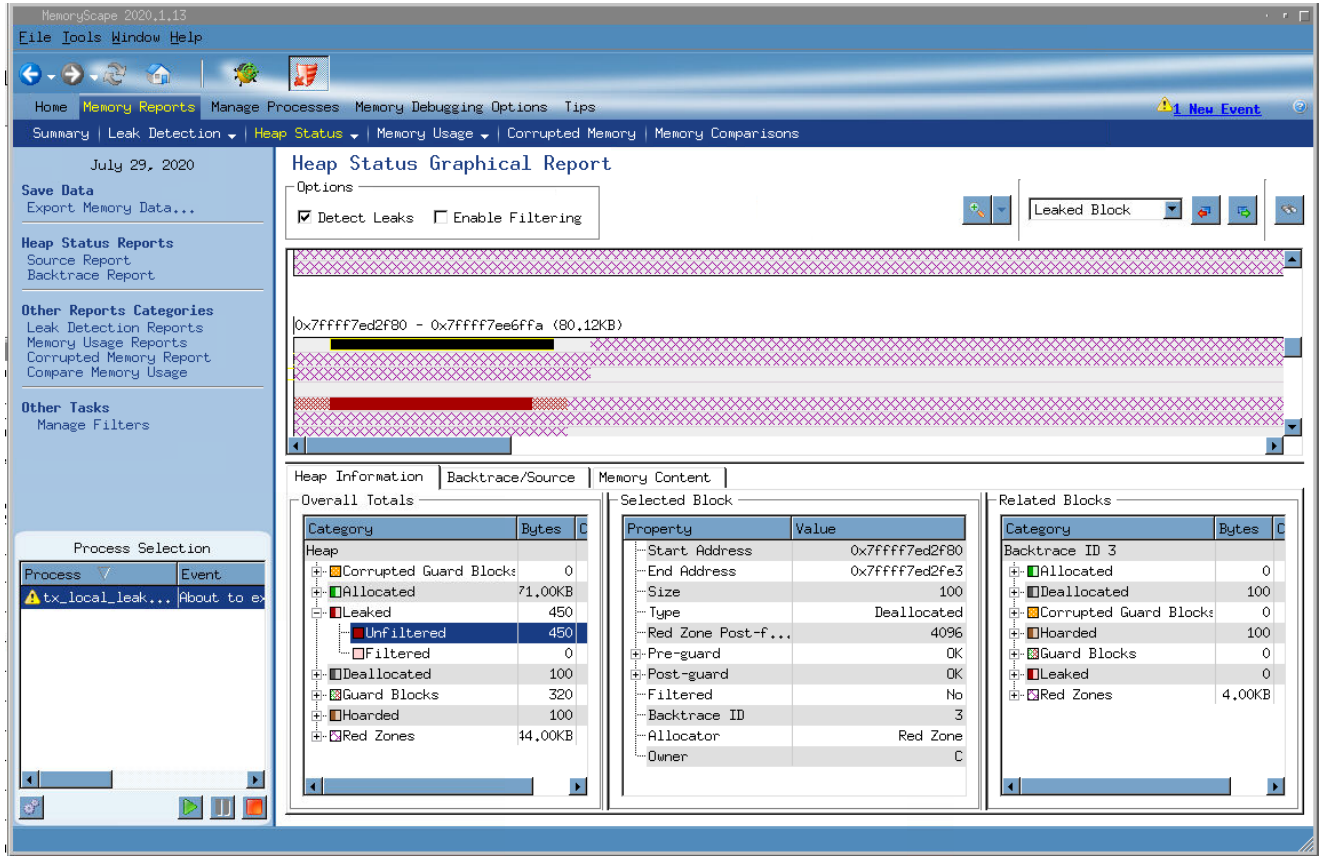
- **Captures execution history**
  - Records all external input to program
  - Records internal sources of non-determinism
- **Replays execution history**
  - Examine any part of the execution history
  - Step back as easily as forward
  - Jump to points of interest
- **An add-on product to TotalView**
  - Support for
    - Linux/x86
    - Linux x86- 64



Memory Debugging

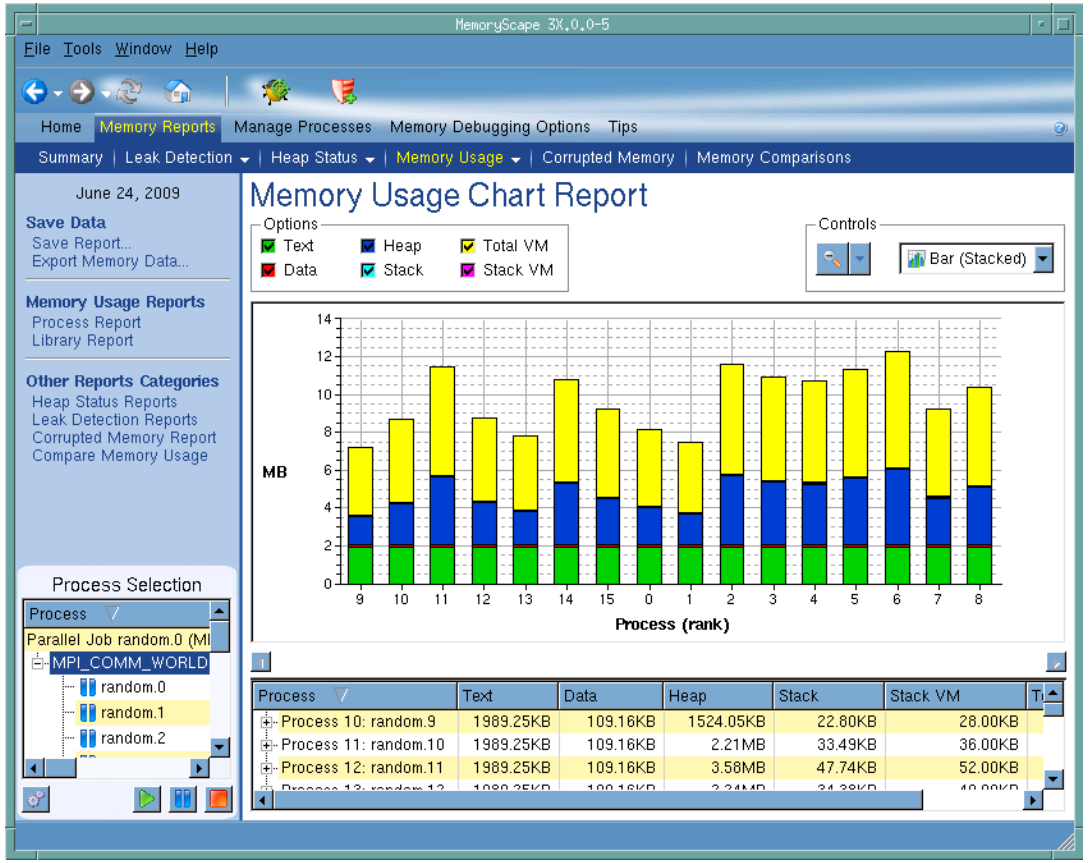
Memory Debugging

- 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



Strategies for Parallel Memory Debugging

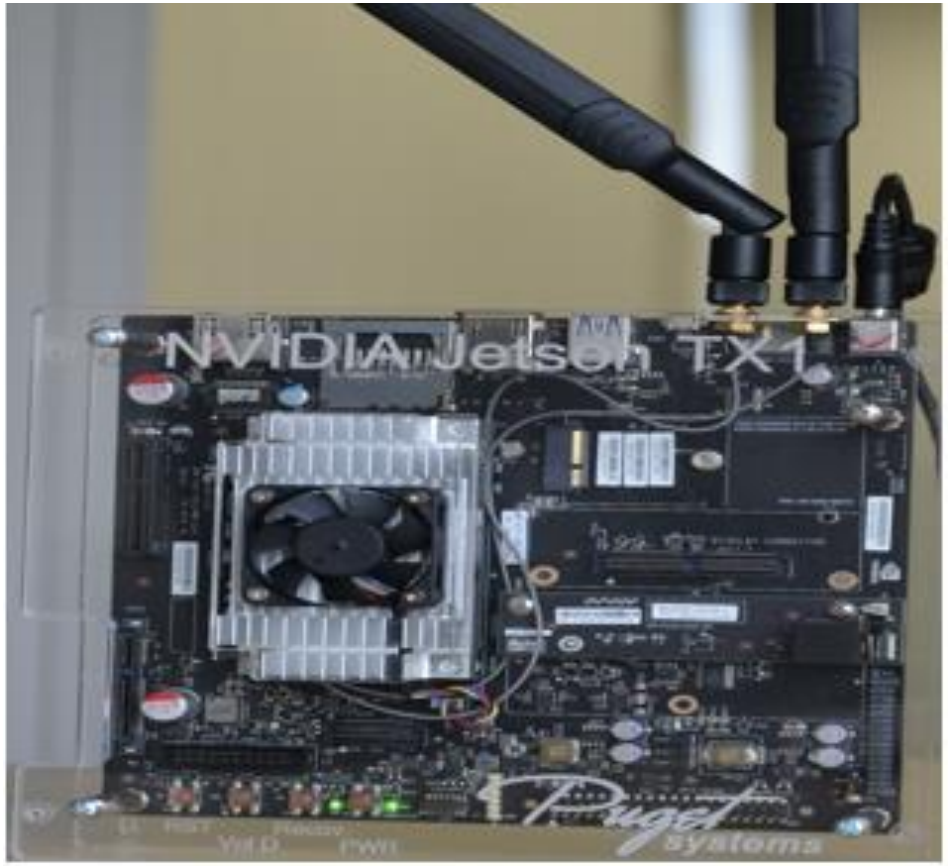
- Run the application and see if memory events are detected
- View memory usage across the MPI job
  - Compare memory footprint of the processes
    - Are there any outliers? Are they expected?
- Gather heap information in all processes of the MPI job
  - Select and examine individually
    - Look at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



GPU Debugging

GPU debugging with TotalView

- NVIDIA CUDA support
  - Multiple platforms : X86-64,PowerLE, ARM64
  - Multiple cards and SDKs
- 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



Extending Debugging Capabilities:
How to Debug (AI) Mixed Python/C++ Code

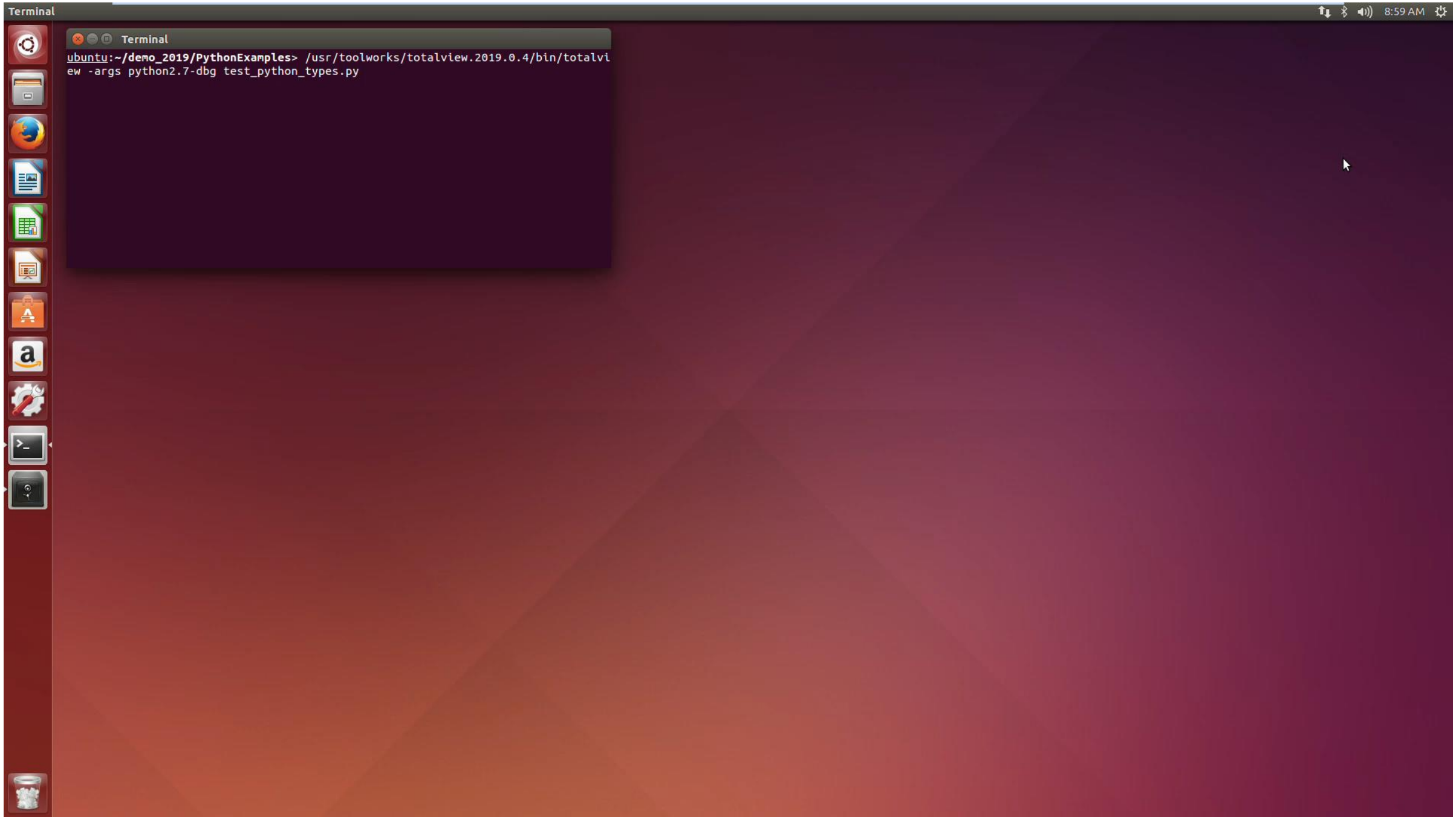
Python debugging with TotalView

- 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++
  - Utilize reverse debugging and memory debugging

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

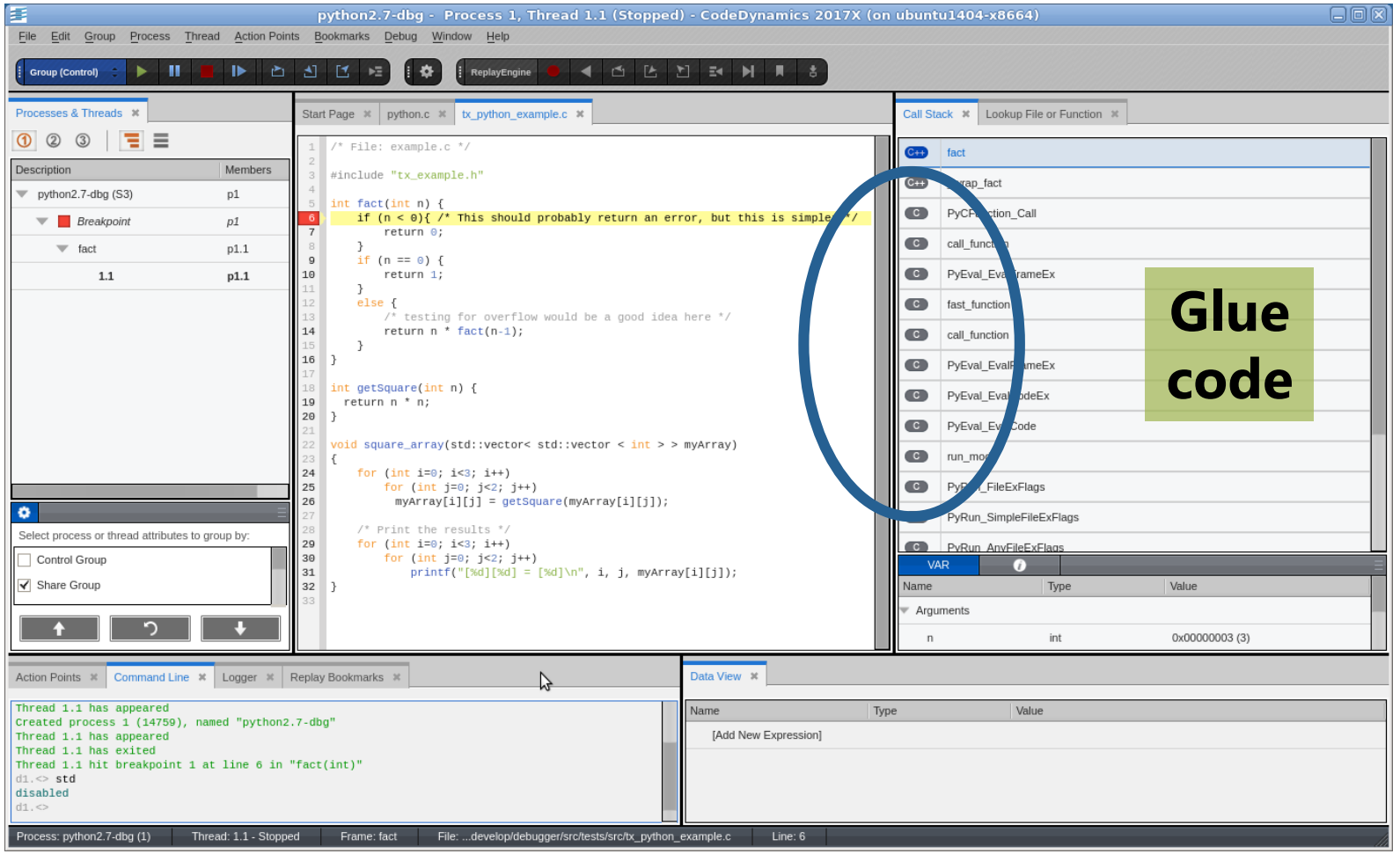
```
#!/usr/bin/python

def callFact():
  import tv_python_example as tp
  a = 3
  b = 10
  c = a+b
  ch = “local string”
   ……
  return tp.fact(a)
if __name__ == '__main__’:
  b = 2
  result = callFact()
  print result
```


Terminal
8:59 AM
Terminal
ubuntu:~/demo_2019/PythonExamples> /usr/toolworks/totalview.2019.0.4/bin/totalvi
ew -args python2.7-dbg test_python_types.py

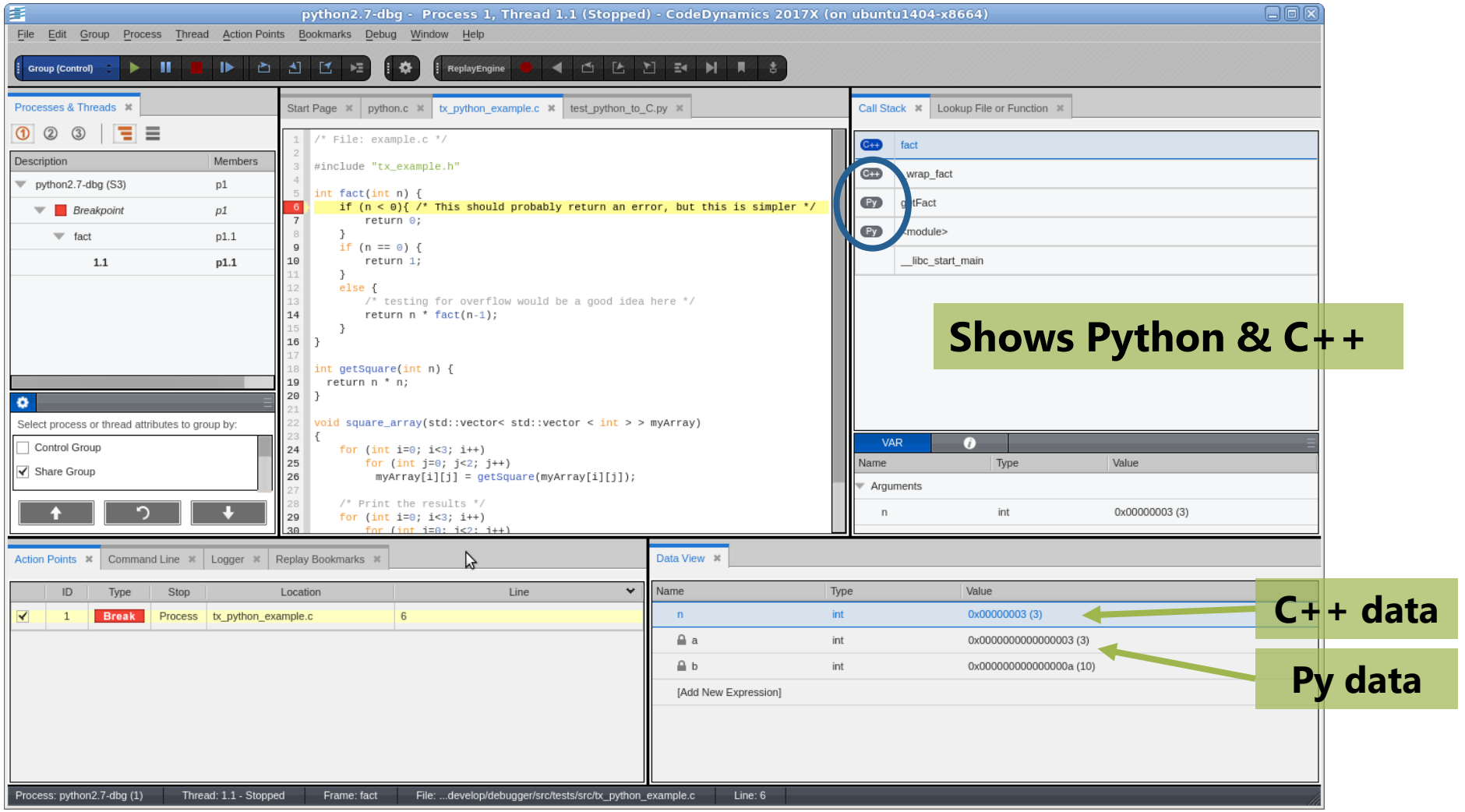
Python without special debugger support

No viewing of Python data and code



Showing C code with mixed data

- Glue code filtered out
- Python data and code available for viewing



Reverse Connections

Remote Display Client (RDC)

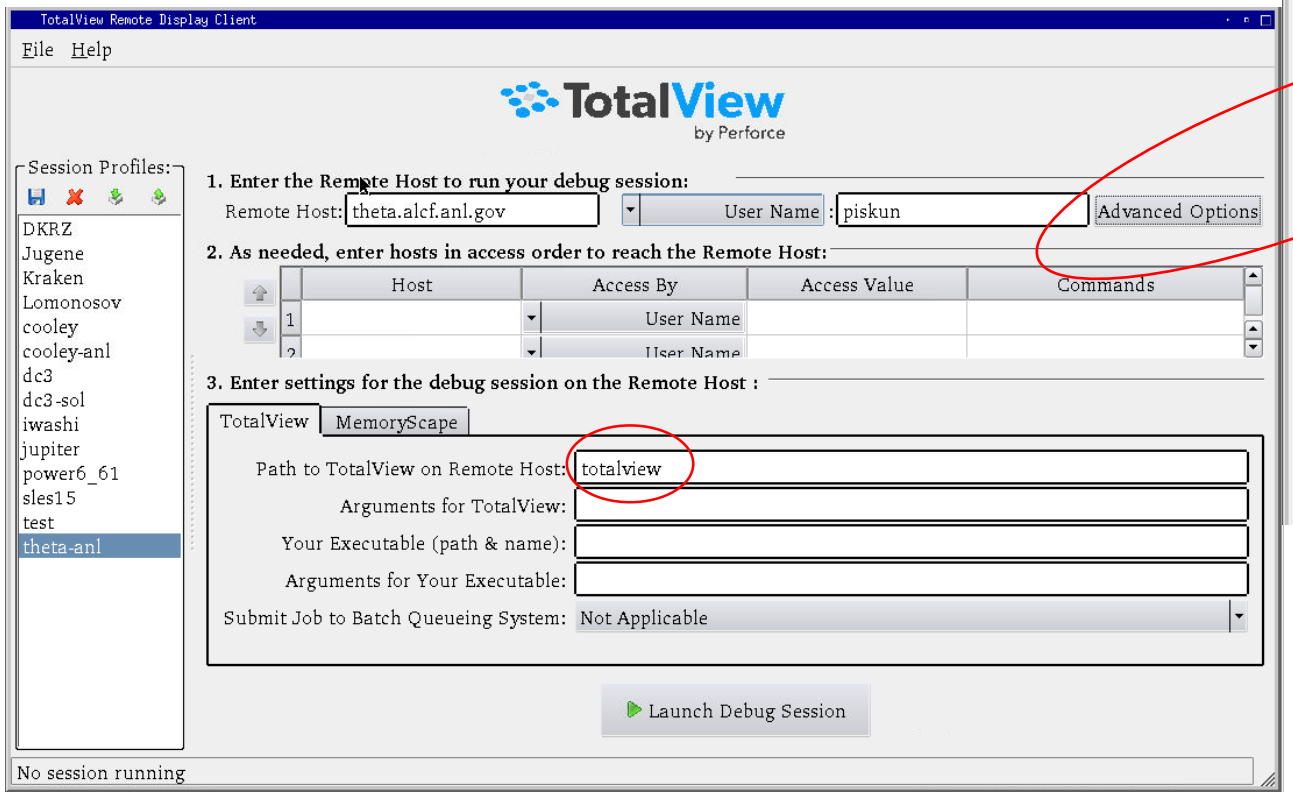
- Offers users the ability to easily set up and operate a TotalView debug session that is running on another system
- Consists of two components
  - Client – runs on local machine
  - Server – runs on any system supported by TotalView and “invisibly” manages the secure connection between host and client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

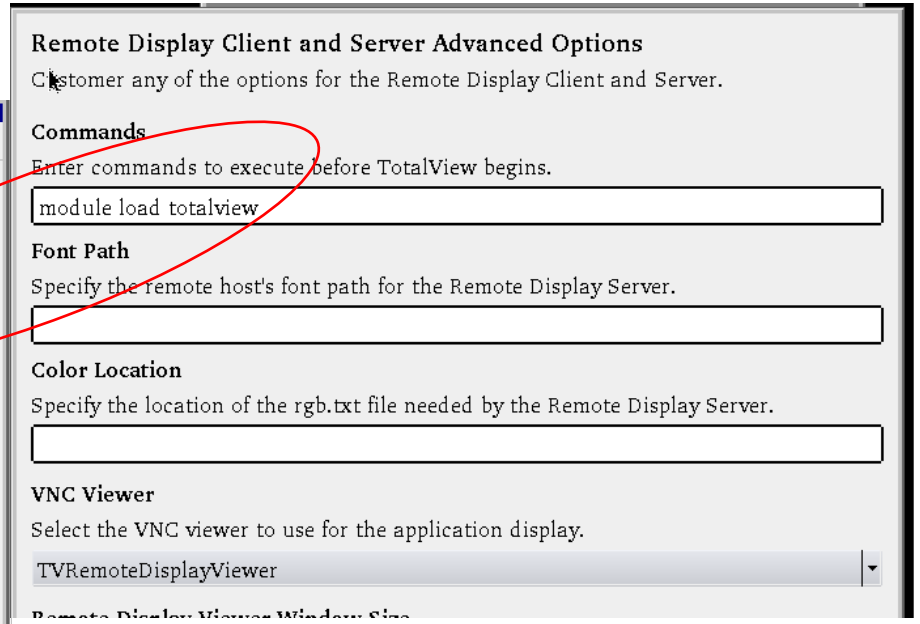

TotalView
by Perforce
Remote Display Client
TotalView by Perforce © Perforce Software, Inc.

Using TotalView for Parallel Debugging on ANL

RDC on Linux and Mac OS

- Download and install RDC from
  - /projects/ATRESC2020/piskun/RDC_installer.1.5.1-macos.dmg
  - /projects/ATRESC2020/piskun/RDC_installer.1.5.1-linux-x86-64.run





- In .ssh/config add:

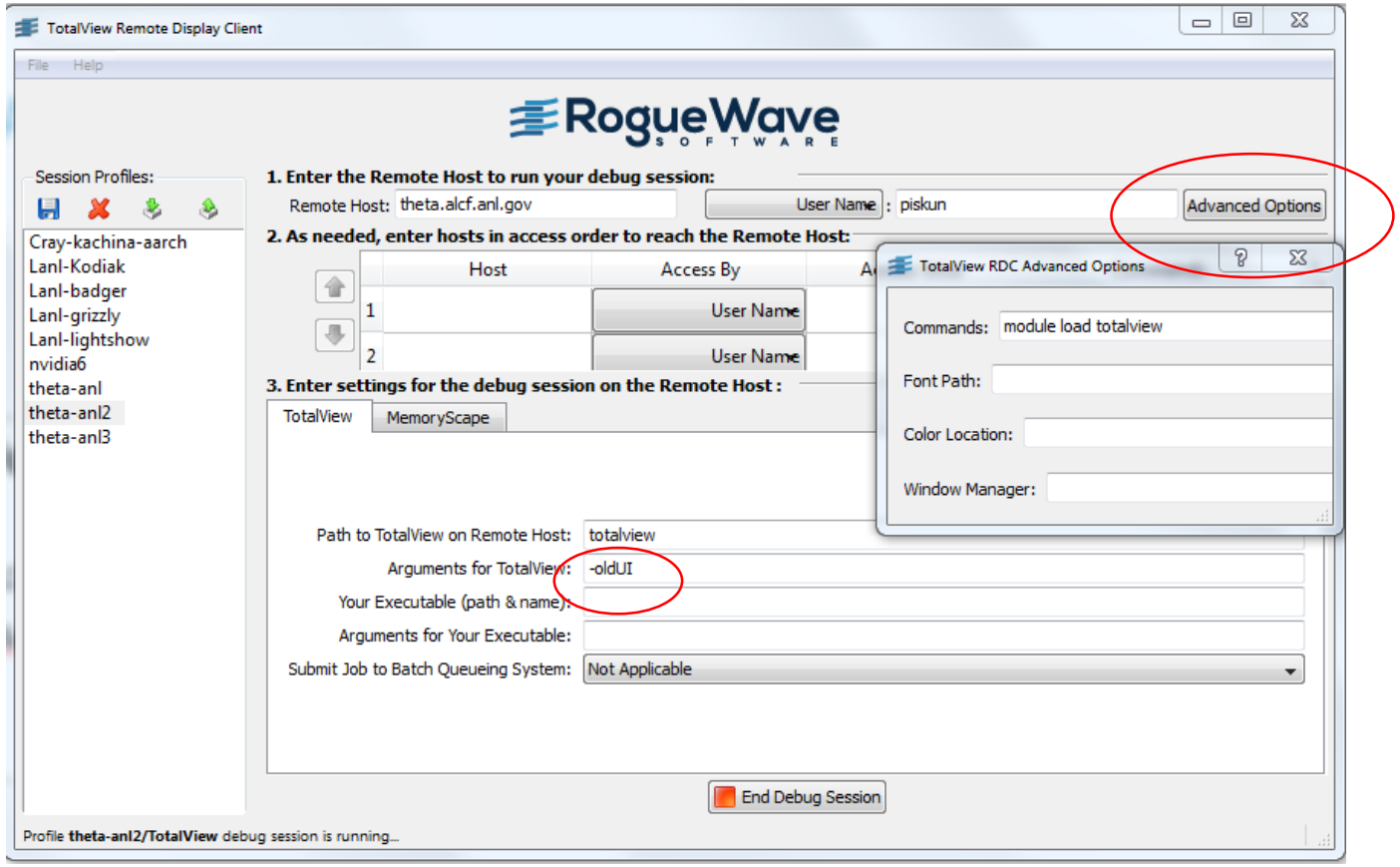
  Host *

  ForwardX11 yes

  StrictHostKeyChecking no

On Windows

- Download and install RDC from
  - /projects/ATRESC2020/piskun/RDC_installer.1.4.2-2.exe



- In .ssh/config add:

  Host *

  ForwardX11 yes

  StrictHostKeyChecking no

Hands-on labs

- /projects/ATRESC2020/piskun/labs/
- Lab 1 Debugger Basic
- Lab 2 Viewing, Examining, Watching and Editing Data
- Lab 3 Examining and Controlling a Parallel Application.
  - Use aprun instead of mpiexec and Cray-aprun as parallel system.
- Using remote connect (tvconnect)
  - Start totalview
  - Modify and submit tvconnect.job

TotalView is available on on Theta, Cooley

- Installed at: /soft/debuggers/totalview-2020-07-27/toolworks/totalview.2020X.2.3/bin/totalview
  - module load totalview

- Connect to Theta
  - Get allocation first
  - qsub -A ATPESC2020 –n <N> –q debug-flat-quad –I
  - module load totalview
  - totalview -args aprun –np <N> ./demoMpi_v2

- Connect to Cooley
  - On Cooley
    - Add –attr=nox11
    - Set DISPLAY by ssh to compute node.
  -

TotalView Resources & Documentation

- TotalView documentation:
  - http://totalview.io
  - User Guides: Debugging, Memory Debugging and Reverse Debugging
  - Reference Guides: Using the CLI, Transformations, Running TotalView
- TotalView online HTML doc:
- Other Resources (Blogs, videos, white papers, etc):
- New UI resources:
- New UI videos:
- Python Debugging blog:
  - http://blog.klocwork.com/dynamic-analysis/the-challenge-debugging-python-and-cc-applications/

Summary

- Use of modern debugger saves you time.

- TotalView can help you because:
  - It's **cross-platform** (the only debugger you ever need)
  - Allow you to debug accelerators (GPU) and CPU in **one session**
  - Allow you to debug **multiple languages** (C++/Python/Fortran)

TotalView by Perforce

THANK YOU