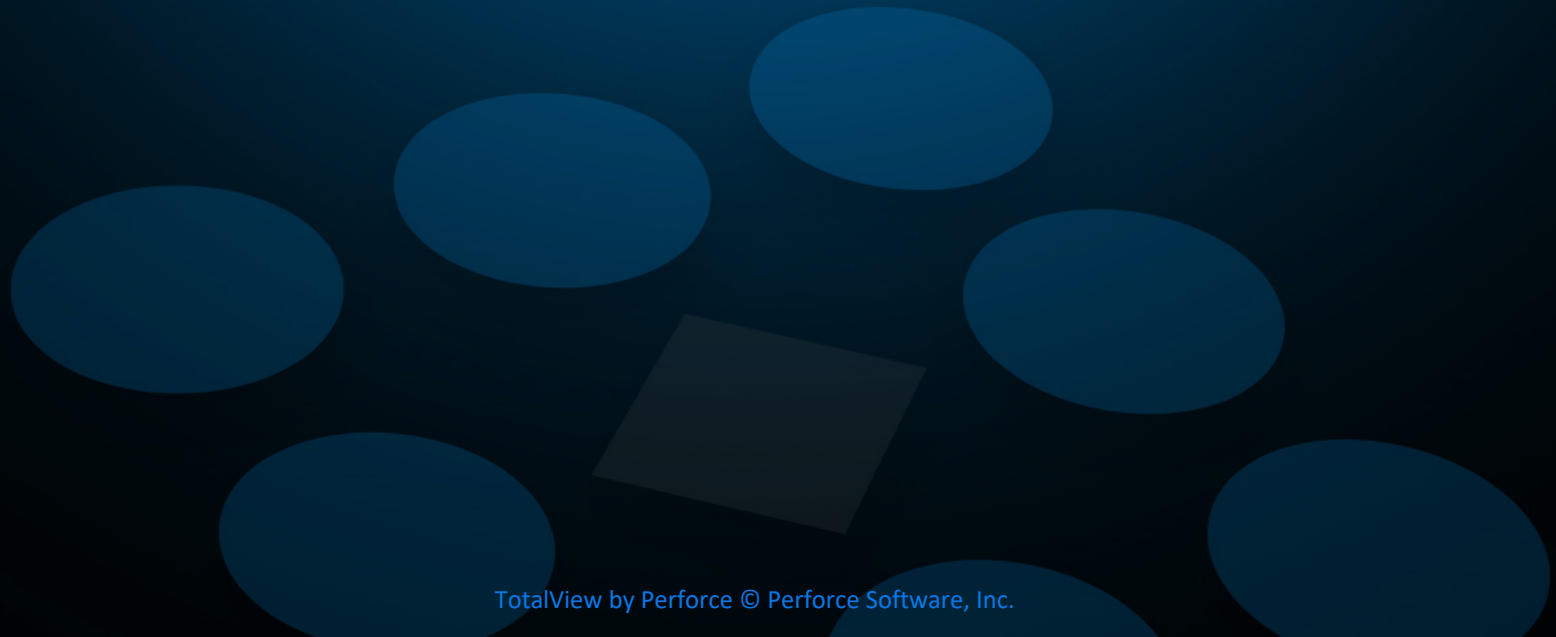
TotalView by Perforce

Advanced Techniques for Debugging Hybrid MPI/OpenMP Applications

BILL BURNS – SENIOR DIRECTOR OF PRODUCT DEVELOPMENT AND PRODUCT MANAGER



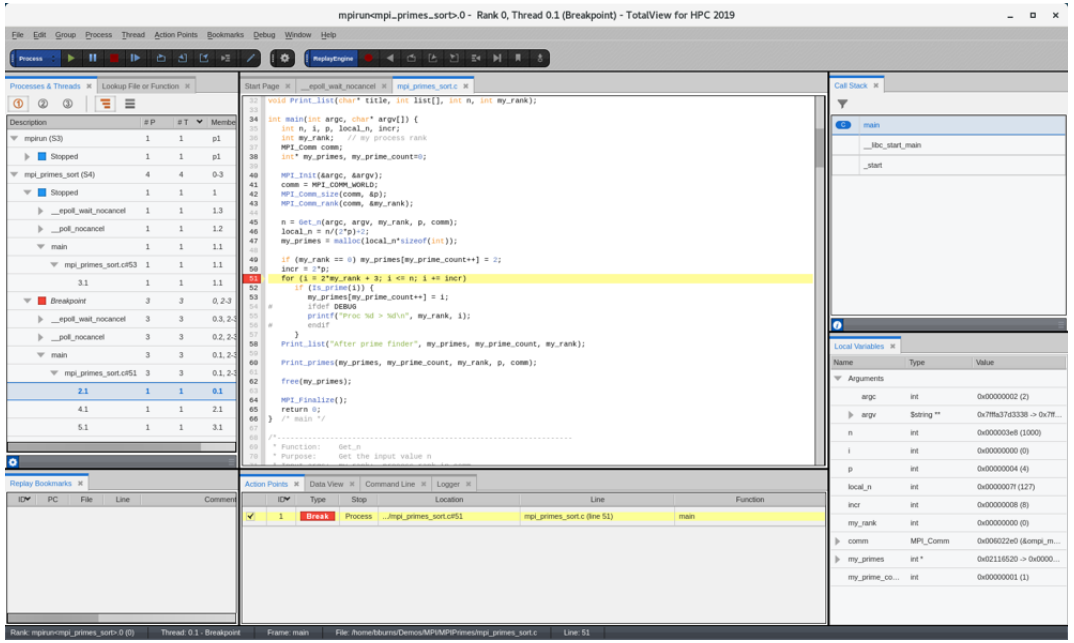
Agenda

1. Hybrid Debugging and Dynamic Analysis with TotalView
2. Advanced MPI and OpenMP Debugging
3. A Toolbox of Advanced Debugging Technologies

Hybrid Debugging and Dynamic Analysis with TotalView

Introduction to HPC Debugging With TotalView

- Comprehensive multi-process/thread dynamic analysis and debugging
- Debug Hybrid MPI/OpenMP applications
- Advanced C, C++ and Fortran support
- CUDA debugging support
- Integrated reverse debugging
- Mixed language C/C++ and Python debugging
- Memory leak detection
- Batch/unattended debugging

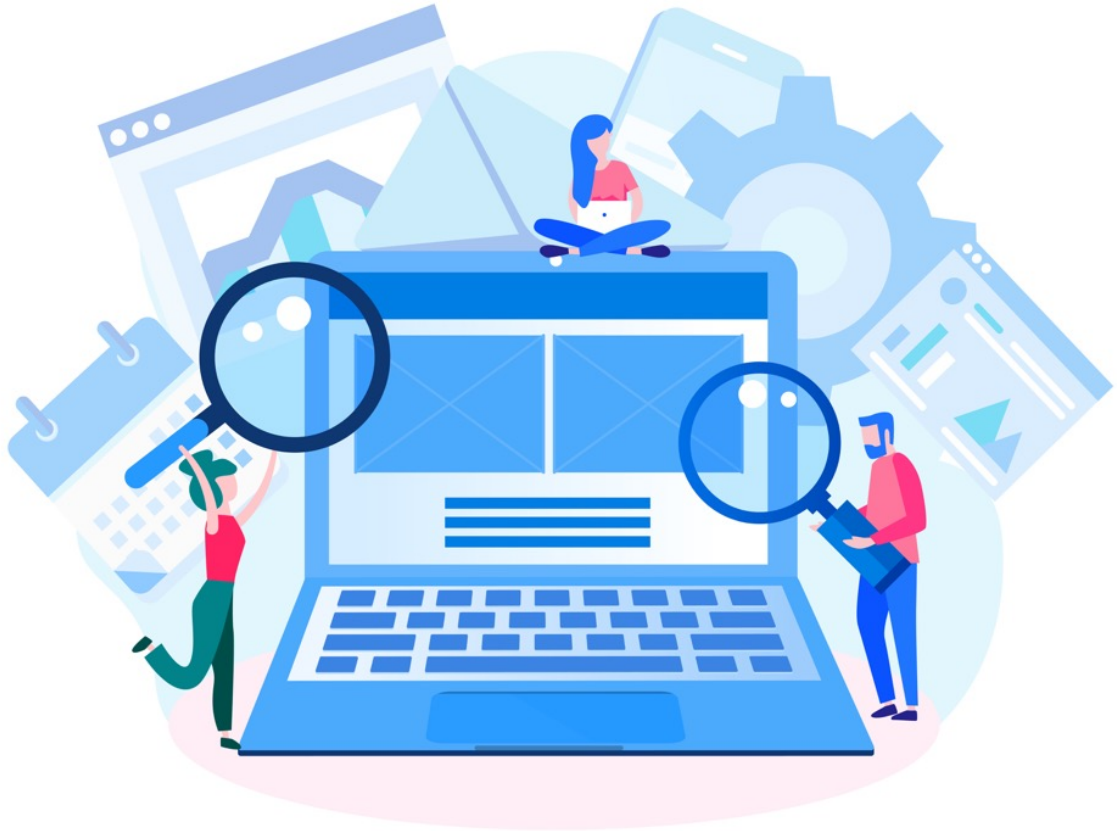


Supported Technologies…

LANGUAGES	OPERATING SYSTEMS	APPLICATIONS	PLATFORMS

More Than Just a Tool to Find Bugs

- Understand complex code
- Improve developer efficiency
- Collaborate with team members
- Improve code quality
- Shorten development time



Advanced MPI and OpenMP Debugging

Parallel Programming Models – Hybrid Model

- A variety of parallel programming models exist to extract maximum performance out of compute resources.
- Message passing models are used to maximize parallelism across compute nodes – MPI technology.
- Thread models, a type of shared memory programming, is used to maximize parallelism across cores within a compute node – OpenMP technology.
- A hybrid programming model combines the parallelism provided by the message passing model (MPI) with the thread model (OpenMP).
- Hybrid model also applicable to a CPU-GPU (Graphics Processing Unit) programming.

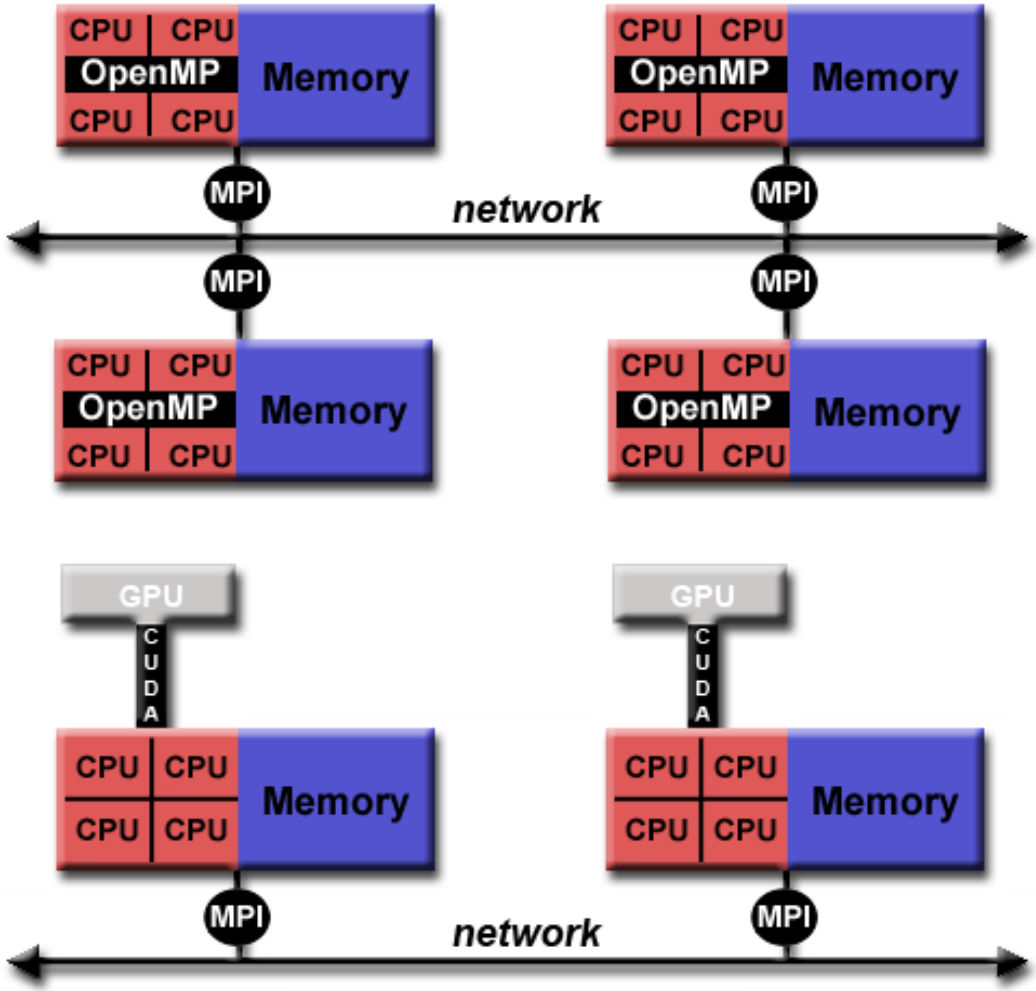
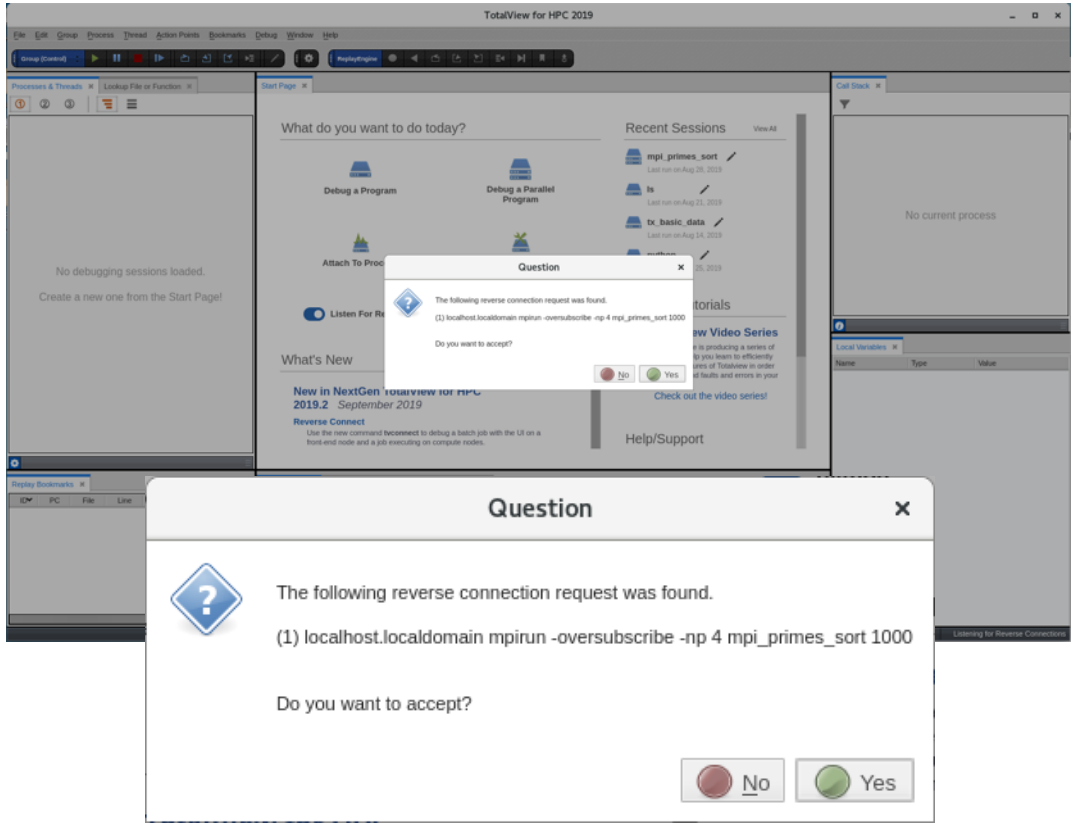


Image from U.S. Department of Energy by Lawrence Livermore National Laboratory

Debugging Hybrid Models – MPI Debugging

- Start a hybrid debugging session using TotalView Reverse Connect.
- Reverse Connect enables the debugger to be submitted to a cluster and connected to the GUI once run.
- Enables running TotalView UI on the front-end node and remotely debug jobs executing on the compute nodes.
- Very easy to utilize, simply prefix job launch or application start with “tvconnect” command.

```
#!/bin/bash
#SBATCH -J hybrid_fib
…
#SBATCH -n 2
#SBATCH -c 4
#SBATCH --mem-per-cpu=4000
export OMP_NUM_THREADS=4
tvconnect srun -n 2 --cpus-per-task=4 --mpi=pmix ./hybrid_fib
```



Debugging Hybrid Models – MPI Debugging Session

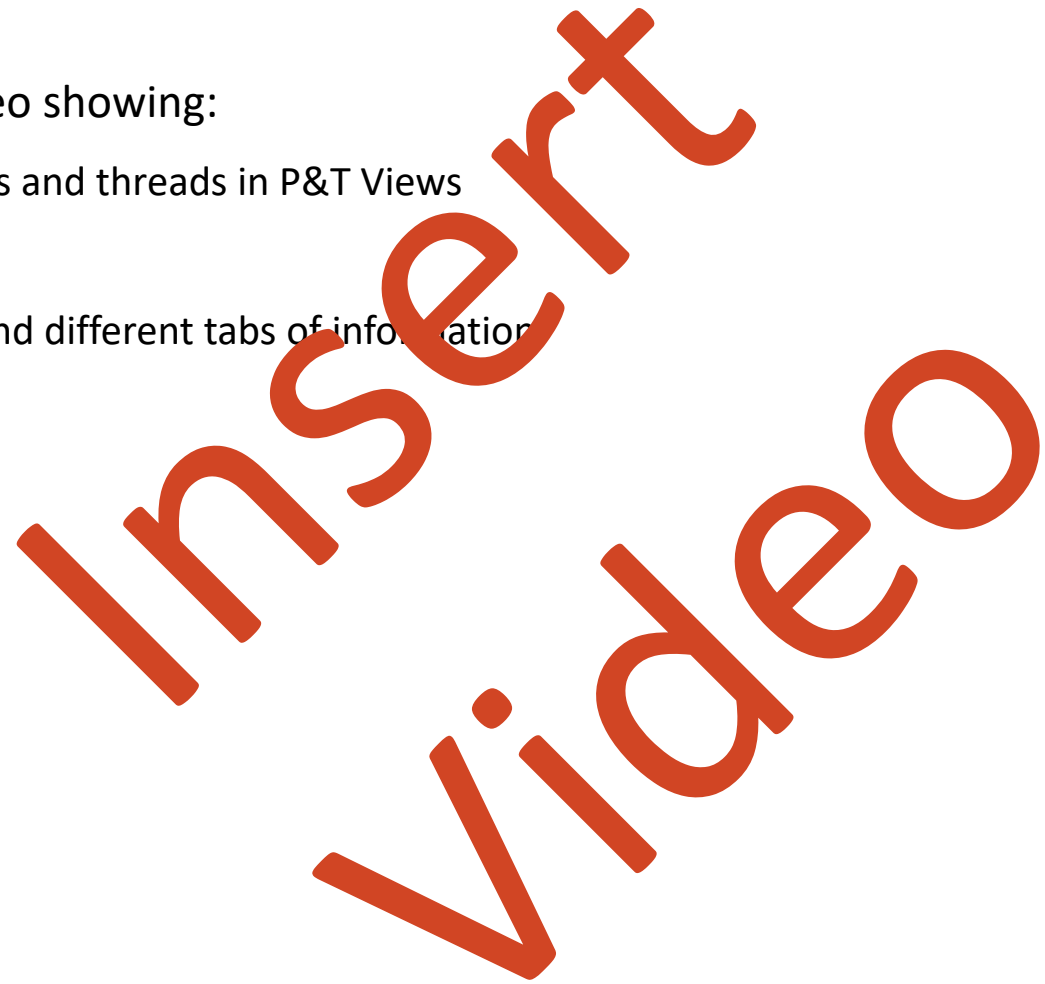
- TotalView demo video showing:
  - Explain Fibonacci test program
  - Start of setting up debugging session
  - Setting a breakpoint
  - Launching
  - Exploring navigation of multiple ranks
  - Transition to OpenMP Debugging

Debugging Hybrid Models – OpenMP Debugging

- OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran.
  - Compiler directives, library routines and environment variables influence run-time behavior.
- Latest OpenMP Debugging API (OMPD v5.0) provides an innovative new interface allowing tools such as debuggers to extract execution state of an OpenMP runtime library.
  - TotalView team had direct involvement in defining the OMDP v5.0 specification.
- Extract threads, parallel and task regions, internal control variables, thread relationships and runtime call-stack boundaries.
- OMPD v5.0 features integrated directly into TotalView's OMP debugging capabilities.
- OMPD only supported in latest Clang compilers, not yet released as a part of the runtime library.

Debugging Hybrid Models – MPI/OpenMP Debugging Session

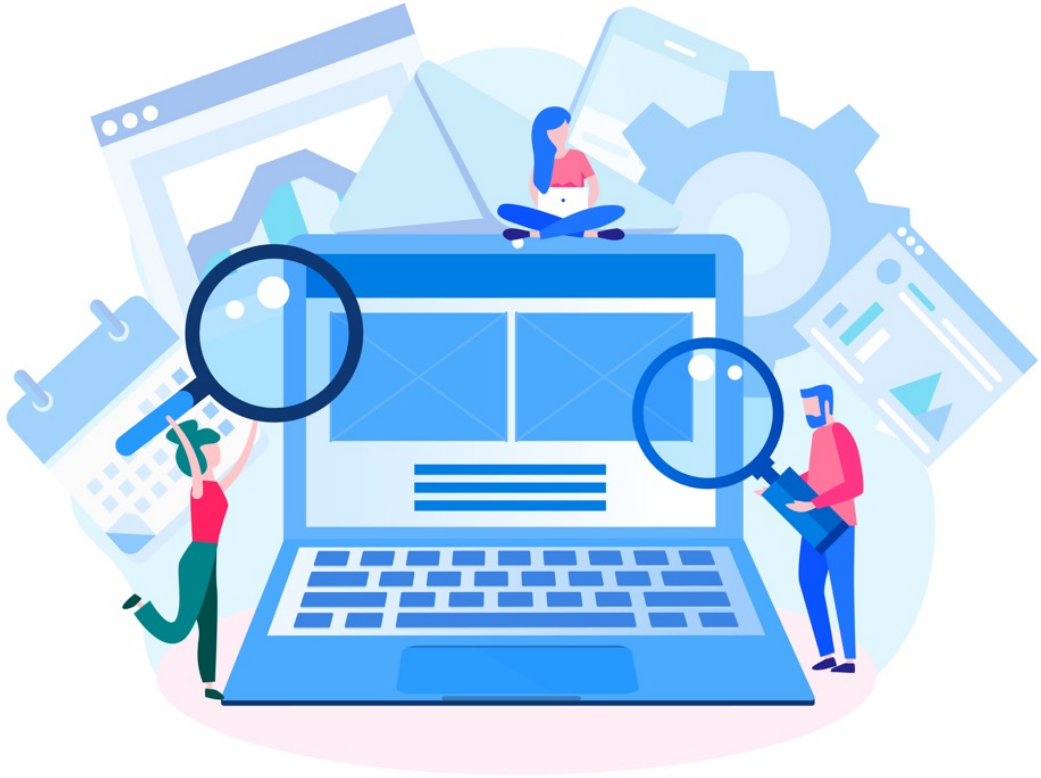
- Continue TotalView demo video showing:
  - Show the layout of processes and threads in P&T Views
  - Show Call Stack and backlink
  - Show new OpenMP Views and different tabs of information



Debugging Hybrid Models – MPI/OpenMP/GPU Techniques

Hybrid GPU Debugging with TotalView

- NVIDIA CUDA 11, 10, 9 support
- Multiple platforms: X86-64, ARM64, PowerLE
- Multiple cards: from Jetson Xavier to Turing (Ampere testing)
- Debugging target regions offloaded to NVIDIA GPUs
- Features and capabilities include
  - Support for dynamic parallelism
  - Support for MPI based clusters and multi-card configurations
  - Flexible Physical/Logical Display and Navigation on the CUDA device
- Support for CUDA Core debugging
- Leverages CUDA memcheck
- Support for OpenACC



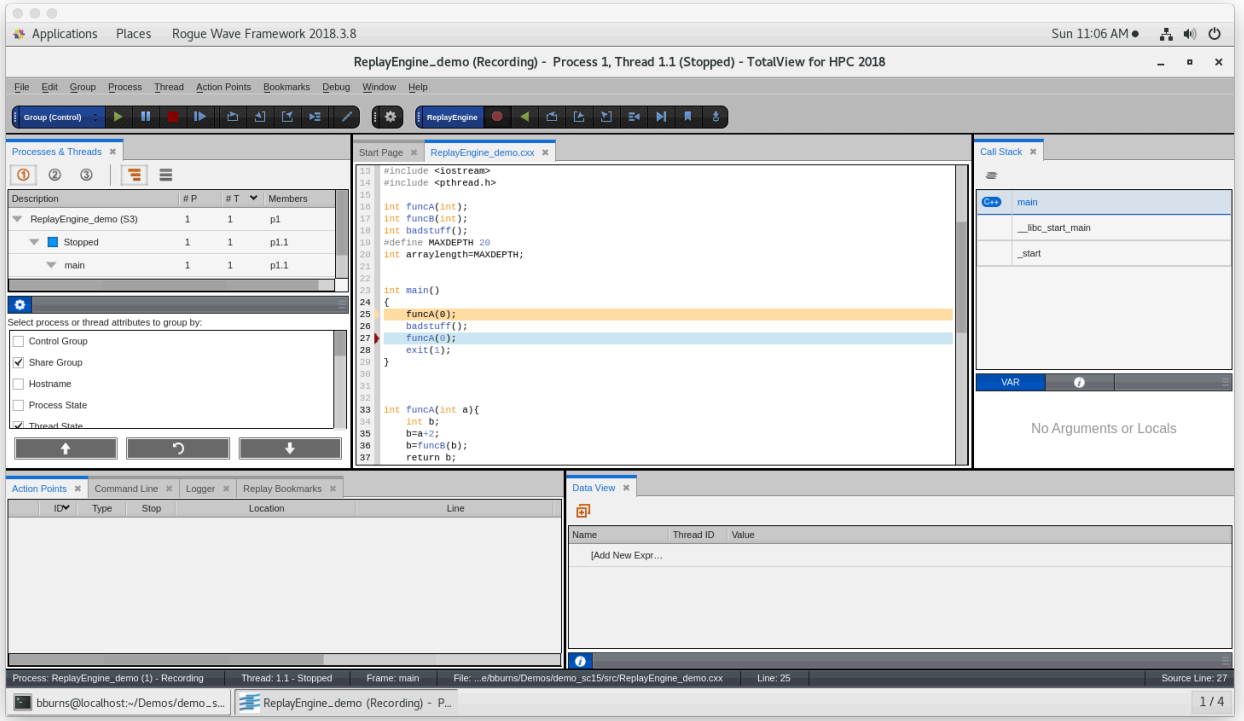
A Toolbox of Advanced Debugging Technologies

Remote Debugging Sessions

- Debugging on a remote HPC cluster can be a challenge
  - Setting up the secure remote connection
  - Launching/connecting to the target application
  - Interactive debugging UI
- TotalView Remote Debugging Options
  - TotalView Remote Display Client
    - Conveniently setup a remote VNC connection
  - TotalView Reverse Connect
    - Disconnect launching the core debugger within the cluster from the UI on a front-end node
  - TotalView Remote UI
    - Run the TotalView UI on a remote client and connect to the remote TotalView debugger

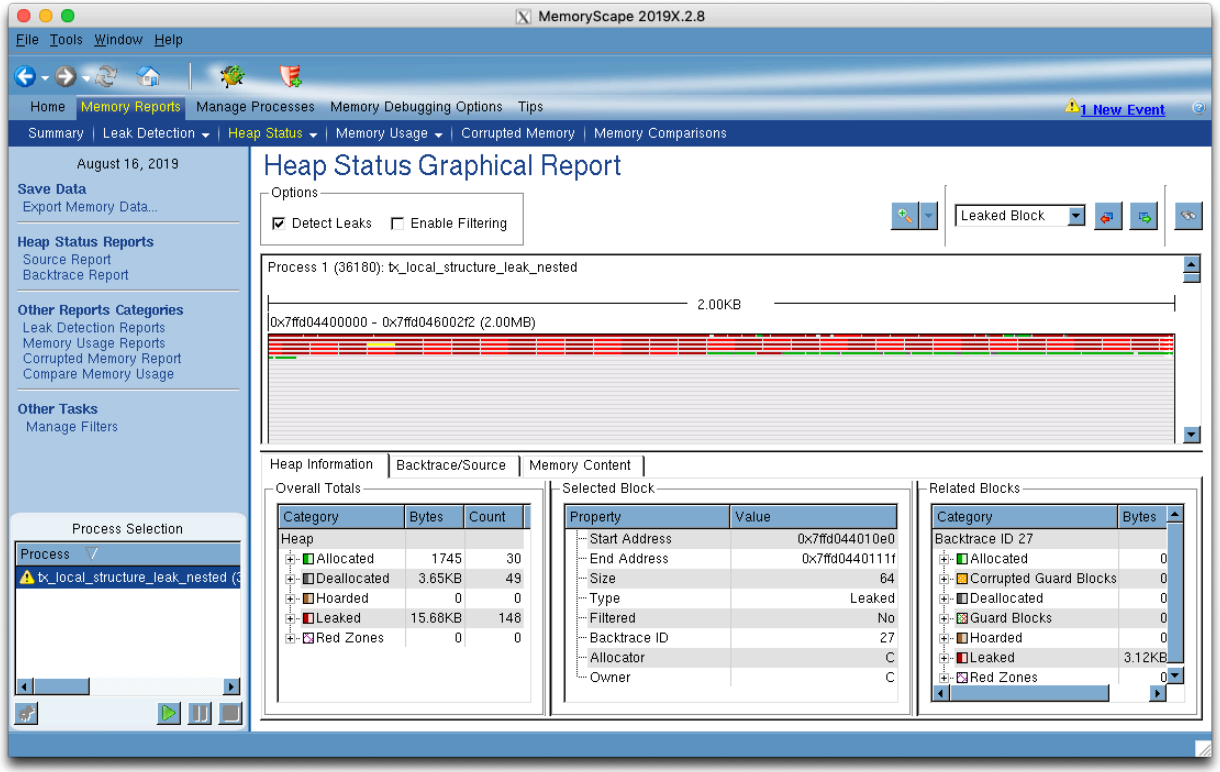
Reverse Debugging

- 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
  - Linux x86/x86-64
- By
  - Capturing and deterministically replay 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



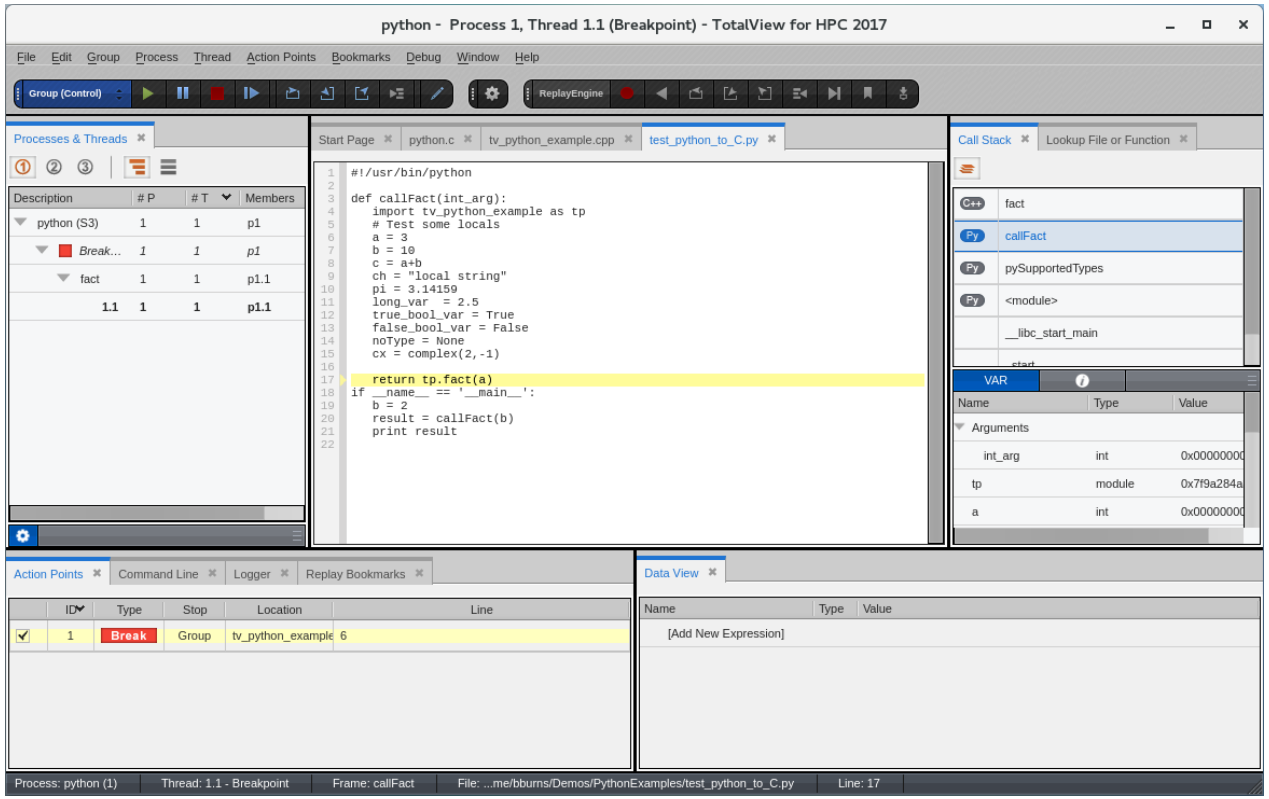
Memory Debugging

- TotalView's memory debugging technology allows you to
  - Easily find memory leaks and other memory errors
  - Detect malloc/free new/delete API misuse
  - Detect buffer overruns
  - Understand where memory is being used
  - Supports remote and MPI debugging
- 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
- Low overhead and does not require recompilation or instrumentation



Mixed Language 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



TotalView Advanced Debugging Capabilities

- Debug the latest C++ features including:
  - lambdas, transformations for smart pointers, auto types, R-Value references, range-based loops, strongly-typed enums, initializer lists, user defined literals
- Transform common STL styles into easy to debug data structures
- Advanced array debugging
- Advanced Data Debugging
  - Easily view a structure member across an array
  - Flexible ways to view and debug data in the UI
- Advanced parallel job debugging control
  - Control from group to individual thread levels
  - Group, process and thread level breakpoint control
  - Dynamically evaluate expressions at points in the code or when memory changes with evaluation points and watchpoints.
  - Synchronize threads and processes with barriers
- Fortran debugging support
- Unattended/Batch Debugging
  - Script driven debugging
  - Enables debugging withing DevOps environments

TotalView Resources and Documentation

TotalView Resources and Documentation

- TotalView website: https://totalview.io
- TotalView documentation:
  - https://docs.roguewave.com/en/totalview/current/
  - User Guides: Debugging, Memory Debugging and Reverse Debugging
  - Reference Guides: Using the CLI, Transformations, Running TotalView
- Blog: https://totalview.io/blog
- Video Tutorials: https://totalview.io/support/video-tutorials

Questions

Questions

- Any questions or comments?
  - Don’t hesitate to reach out to me directly with any questions or comments!
  - **Email:** bburns@perforce.com

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