lrz

TotalView Training for LRZ

Dean Stewart – Principal Sales Engineer Perforce

Bill Burns – Vice President, Software Engineering Perforce

Susanne Horn – SMB GmbH

perforce

© Perforce Software, Inc. All rights reserved.

Confidentiality Statement

The information contained in this document is strictly confidential, privileged, and only for the information of the intended recipient. The information contained in this document may not be otherwise used, disclosed, copied, altered, or distributed without the prior written consent of Perforce Software, Inc.

© Perforce Software, Inc. All rights reserved.

perforce

Agenda

© Perforce Software, Inc. All rights reserved.

perforce

Agenda

- Introduction to Perforce
- HPC Debugging with TotalView
- Multithreaded debugging (OpenMP and pthreads)
- Remote debugging
- Multiprocess debugging
- Understanding complex data structures (advanced C++ and data debugging)
- Debugging CUDA applications
- Mixed Language C/C++ and Python debugging
- Reverse debugging with ReplayEngine
- Memory debugging (memory leaks, buffer overflows, dangling pointers)
- Batch debugging with tvscript
- Debugging I/O bottlenecks
- Best practices for debugging with TotalView
- Q&A

© Perforce Software, Inc. All rights reserved.

perforce

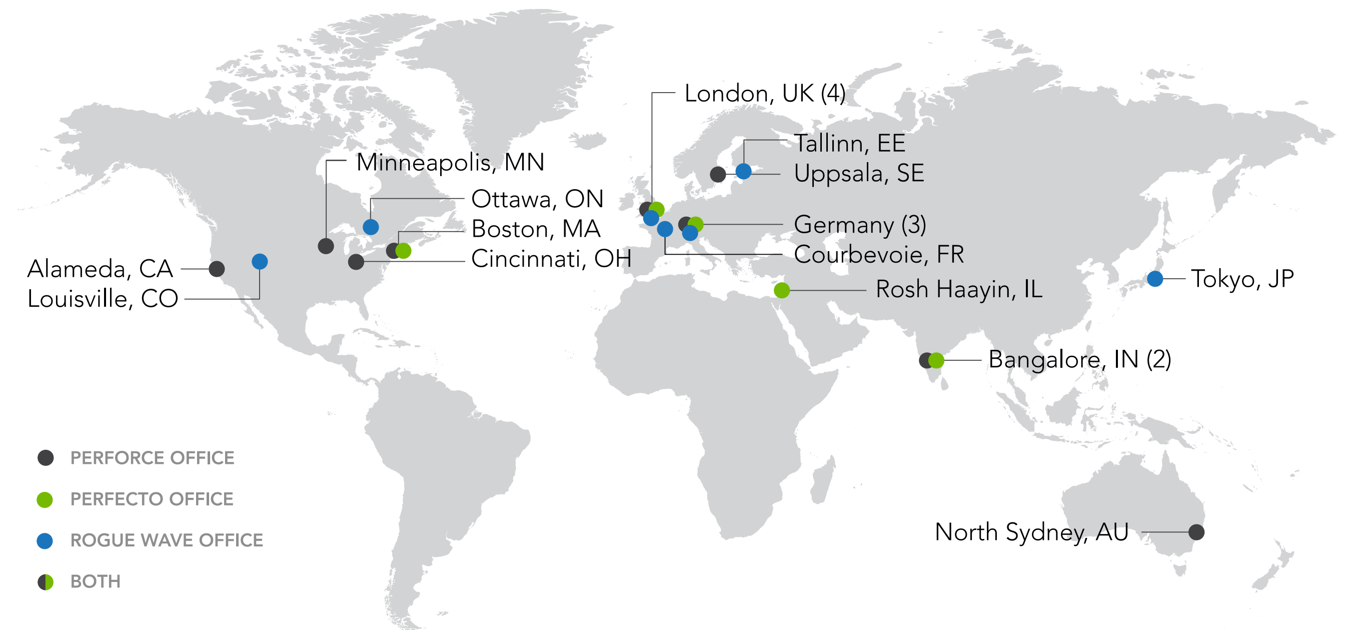
Introduction to Perforce

© Perforce Software, Inc. All rights reserved.

perforce

Global Footprint

- Customers in 80 countries
- 9,000 customers worldwide
- More than 250 of the Fortune 500
- Customers deploying multiple products
- 16 offices and 4 data centers which give us global reach
- 1,700 employees



© Perforce Software, Inc. All rights reserved.

perforce

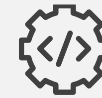
Perforce Portfolio

Digital Creation

- P4 Server
- P4 DAM
- P4 Plan
- ALM
- IPLM
- P4 ONE

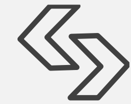
Developer Tools

- Akana
- OpenLogic
- Zend
- JRebel
- Gliffy
- Components
- TotalView



Testing Automation

- Perfecto
- BlazeMeter
- Klocwork
- Helix QAC



Platform Automation

- Puppet
- Delphix

© Perforce Software, Inc. All rights reserved.

perforce

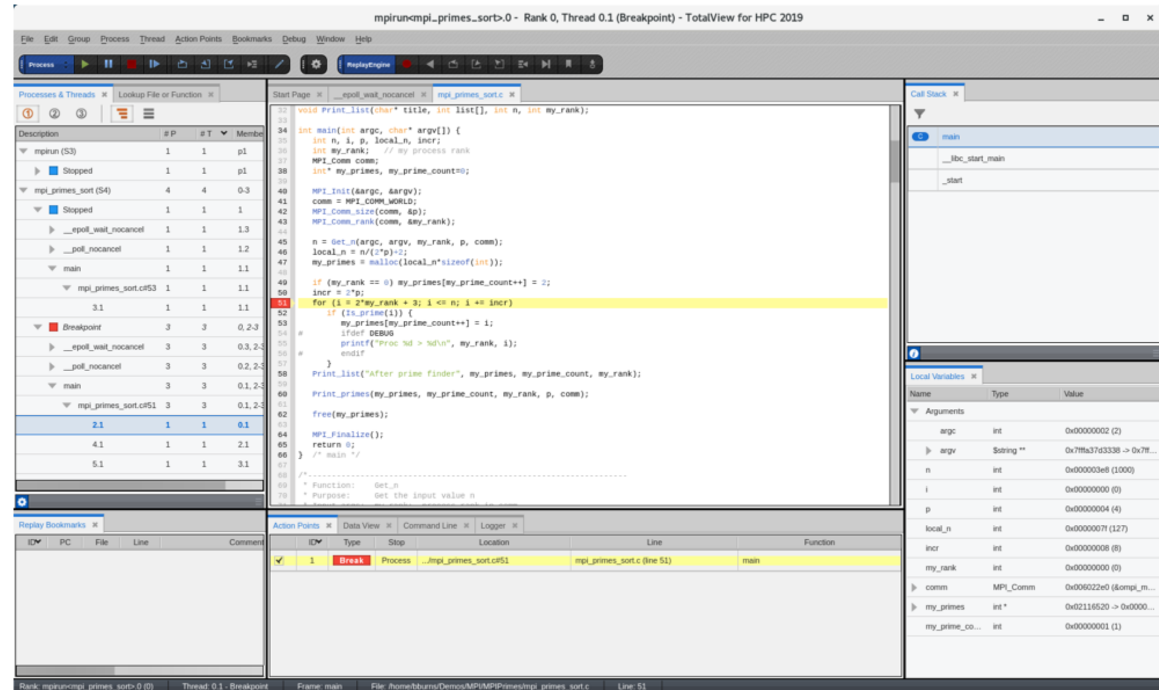
HPC Debugging with TotalView

© Perforce Software, Inc. All rights reserved.

perforce

HPC Debugging with TotalView

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



LANGUAGES

OPERATING SYSTEMS

APPLICATIONS

PLATFORMS

© Perforce Software, Inc. All rights reserved.

Multithreaded Debugging

© Perforce Software, Inc. All rights reserved.

perforce

Multithreaded Debugging with TotalView

- Built from the beginning to support debugging parallel programs
- Supports parallel technologies
  - Processes: fork/exec, MPI, etc
  - Threads: pthreads, TBB, OpenMP, GPU threads
- Provides parallel constructs across debugger features
  - Easily understand overall program state with an aggregate view of all processes and threads
  - Control program behavior with group, process, and thread breakpoints and stepping operations
  - Easily view program data across processes and threads
- Scalable to thousands of threads and processes
- Multi-GPU support

Processes & Threads

Description	# P	# T	Members
tx_fork_loop (S3)	4	4	p1-4
Breakpoint	*4*	*4*	*p1-4*
Breakpoint	*4*	*4*	*p2.1, p4.1, p3.2,...*
snore	4	4	p2.1, p4.1, p3.2,...
tx_fork_loop.cxx#682	4	4	p2.1, p4.1, p3.2,...
1.3	1	1	p1.3
2.1	1	1	p2.1
3.2	**1**	**1**	**p3.2**
4.1	1	1	p4.1
Stopped	4	8	p1.1, p3.1, p1-2....
__select_nocancel	2	3	p1-2.2, p2.3
<unknown line>	2	3	p1-2.2, p2.3
1.2	1	1	p1.2
2.2	1	1	p2.2
2.3	1	1	p2.3
snore	3	5	p1.1, p3.1, p4.2,...
tx_fork_loop.cxx#682	3	5	p1.1, p3.1, p4.2,...
1.1	1	1	p1.1
3.1	1	1	p3.1
3.3	1	1	p3.3

© Perforce Software, Inc. All rights reserved.

OpenMP/OMPD Support

- Extends and productizes existing TotalView OpenMP/OMPD debugging support
  - Focuses on supporting the Clang, AMD Clang/Flang, and HPE CCE C, C++, and Fortran compilers / OMPD libraries
- Single-step into and out of parallel regions automatically
- Display OpenMP runtime state of parallel/task regions, threads, control variables, and ICVs
- Enhanced stack displays using OMPD-provided parallel/task region information
  - Filter-out OpenMP runtime frame
  - Annotate parallel/task region frames with "#pragma omp parallel" / "#pragma omp task"
  - Insert "parent back-link" frames to focus on a parallel region's encountering thread/frame
- Support for OMPD interface defined by the OpenMP 5.0 - 5.2 APIs
- Note: OMPD-based features are CPU only
  - An OMPD library implementation for GPUs does not exist yet

© Perforce Software, Inc. All rights reserved.

perforce

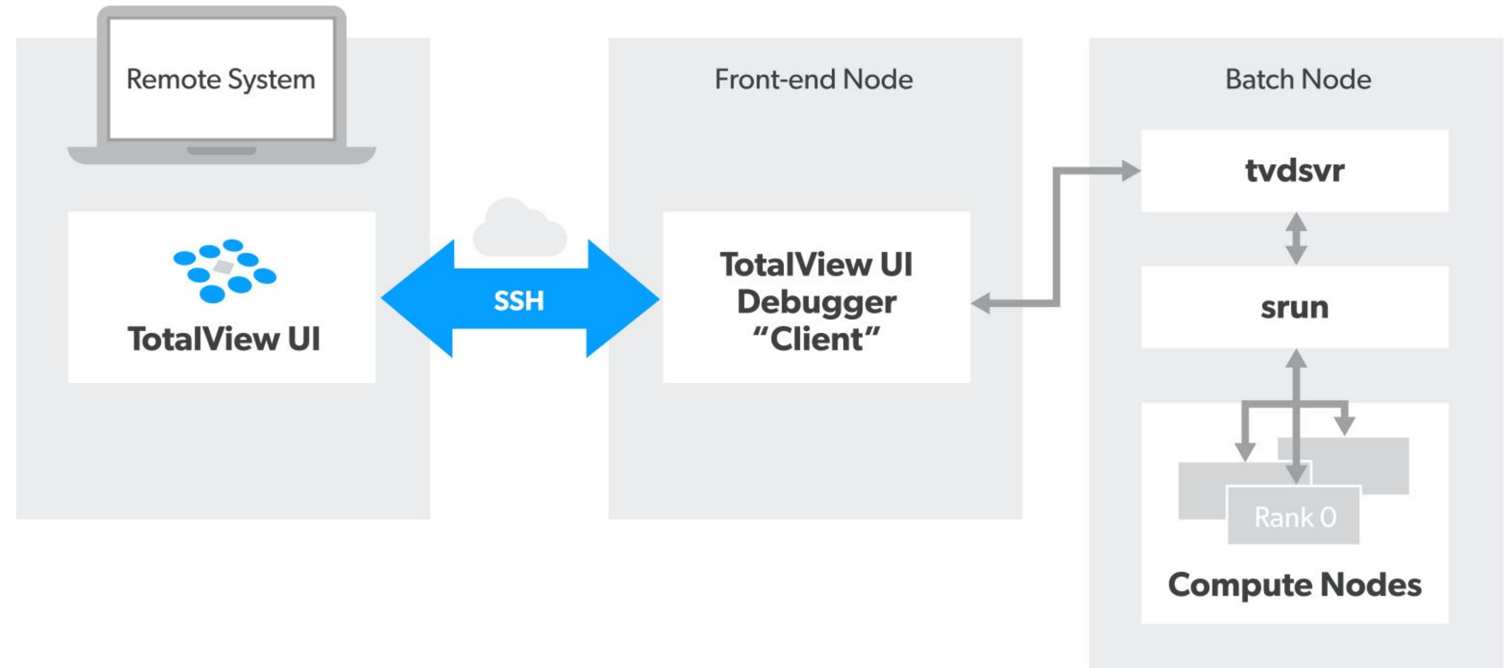
Remote Debugging

perforce

© Perforce Software, Inc. All rights reserved.

Remote Debugging with TotalView

- Combine the convenience of establishing a remote connection to a cluster and the ability to run the TotalView GUI locally
- Front-end GUI architecture does not need to match back-end target architecture (macOS front-end -> Linux back-end)
- Secure communications
- Convenient saved sessions
- Once connected, debug as normal with access to all TotalView features
- Windows, macOS, Linux Support



© Perforce Software, Inc. All rights reserved.

perforce

Multiprocess Debugging

© Perforce Software, Inc. All rights reserved.

perforce

Starting a Parallel Program Session from the UI

From New Parallel Session page select:

- Parallel Environment
- Launch Style
- Number of tasks
- Number of nodes
- Starter arguments

Click Start Session to save and launch

Session Editor

Parallel Session

Session Details

Session Name

[Enter or select a session name, e.g. myprogram with ReplayEngine]

Parallel Details

Parallel Environment REQUIRED

Slurm srun

Parallel Launch Style

Native

Use the MPI starter process to launch the job (recommended)

Bootstrap

Use the bootstrap server to indirectly launch the job (if Native fails)

Tasks (-n): [Enter the number of tasks]

Nodes (-N): [Enter the number of nodes]

Additional Starter Arguments

[Enter starter arguments as needed]

Program Details

File Name REQUIRED

[Enter program path and name, e.g. /home/smith/myprogr...

BROWSE...

Arguments

[Enter any program arguments. Ex. -option foo]

Debug Options

Python Debugging

Enable call stack filtering for Python

Memory Debugging

Enable memory debugging

Options

Program Environment

Working Directory

RESET LOAD SESSION CANCEL

© Perforce Software, Inc. All rights reserved.

perforce

Starting a Parallel Program Session from the Command Line

General Command Line: `totalview --args <starter> -n ## <partition> <program>`

MPI	Startup Command
Linux under SLURM	`totalview --args srun -n 16 -p pdebug <program>`
Open MPI / MPICH / Intel MPI	`totalview --args mpirun -np 16 <program>`

SLURM

- Use salloc or sbatch to grab an allocation
- Use srun to start the job
  ```
  totalview -args srun -N 4 -n 16 <program> <program args>
  ```
- sbatch tv.sbatch to submit a batch job
  - The batch script may need to define a DISPLAY variable to forward TotalView's display
- Use of tvconnect can also simplify a parallel debugging session launch

© Perforce Software, Inc. All rights reserved.

perforce

TotalView Reverse Connections

The Problem:

- Establishing an interactive debugging session in a cluster environment can be difficult
  - Timing issues when submitting through a job manager and when the job runs
  - The organization of modern HPC systems often makes it difficult to deploy tools such as TotalView
  - The compute nodes in a cluster may not have access to any X libraries or X forwarding
  - Launching a GUI on a compute node may not be possible

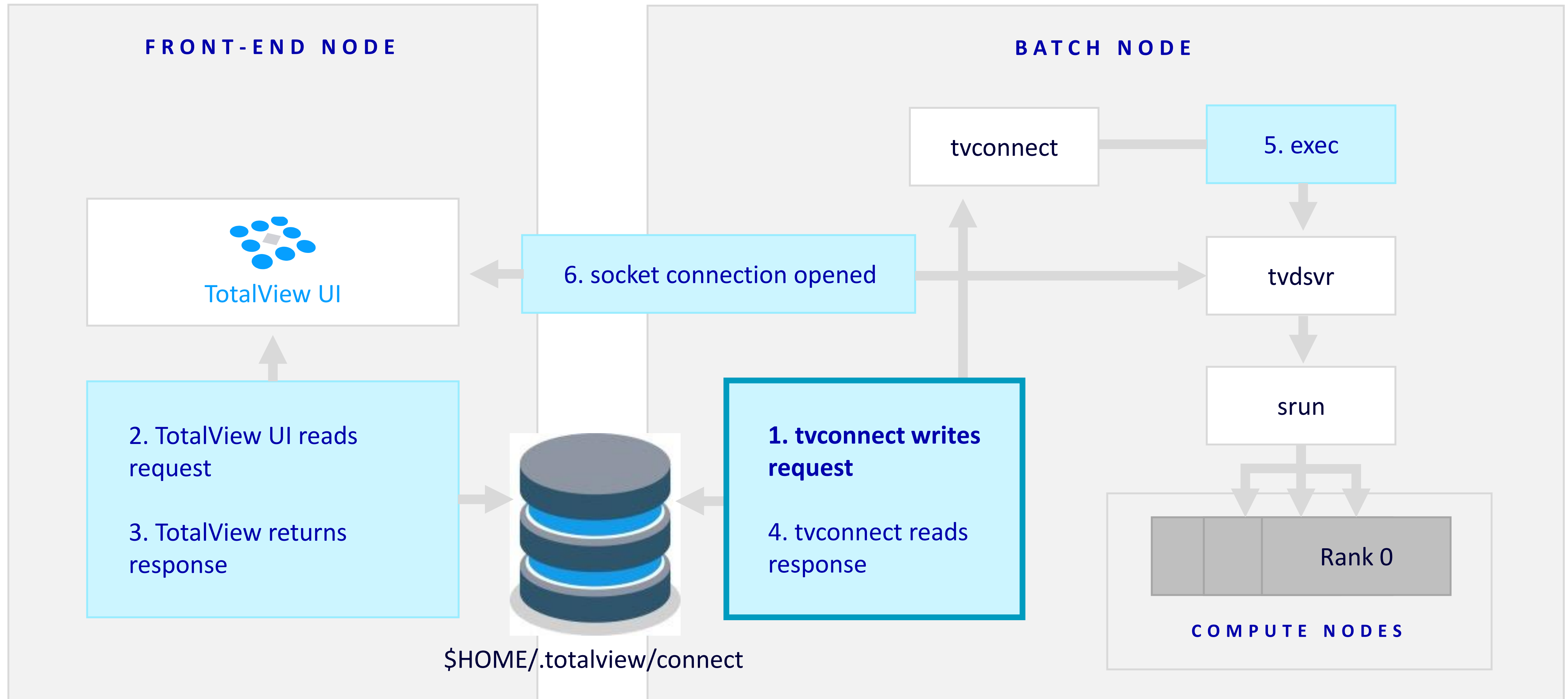
The Solution:

- Disconnect starting debugger UI from the backend job launch and debug session acquisition
- TotalView Reverse Connect workflow enables developers to start the TotalView UI on a front-end node and, when a job is run in the cluster, a remote TotalView reverse connect agent connects it back to the waiting UI

© Perforce Software, Inc. All rights reserved.

perforce

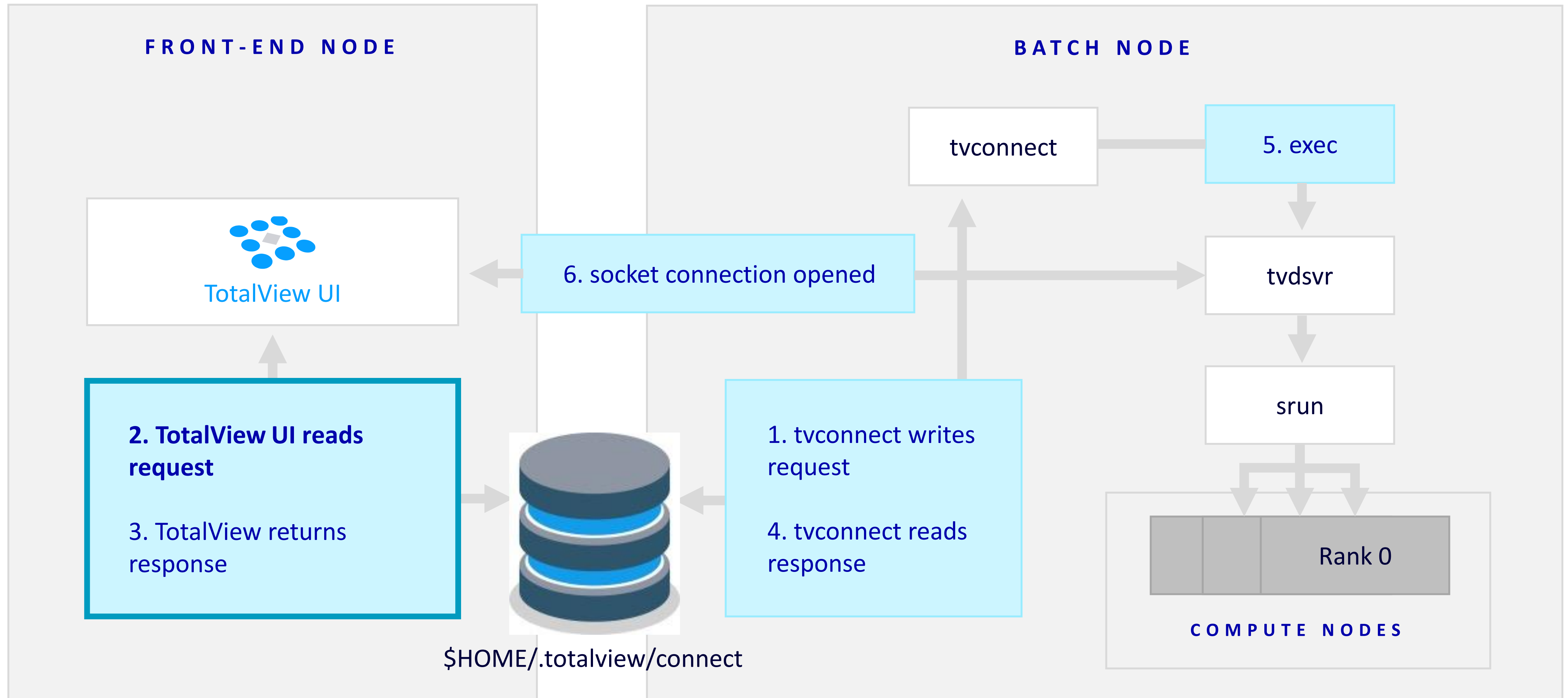
Reverse Connection Flow



© Perforce Software, Inc. All rights reserved.

perforce

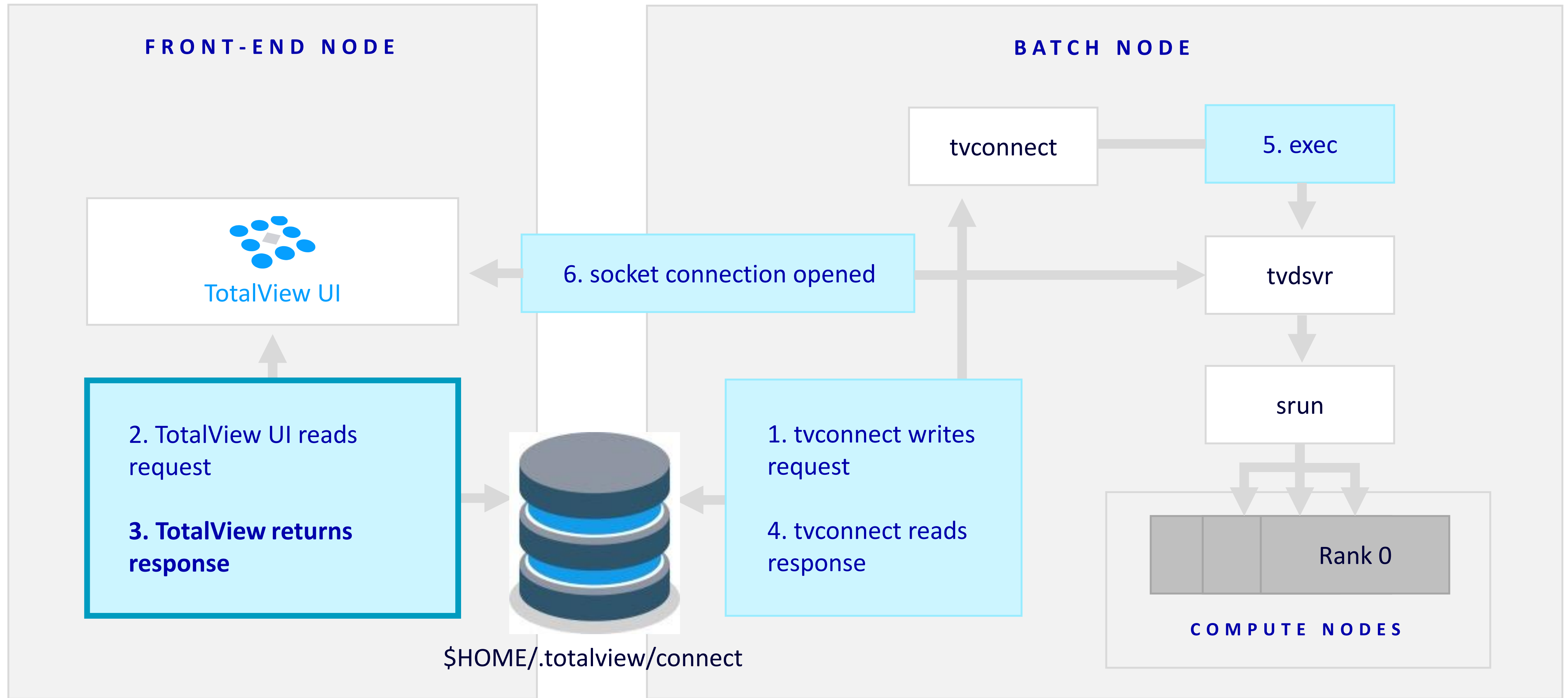
Reverse Connection Flow



© Perforce Software, Inc. All rights reserved.

perforce

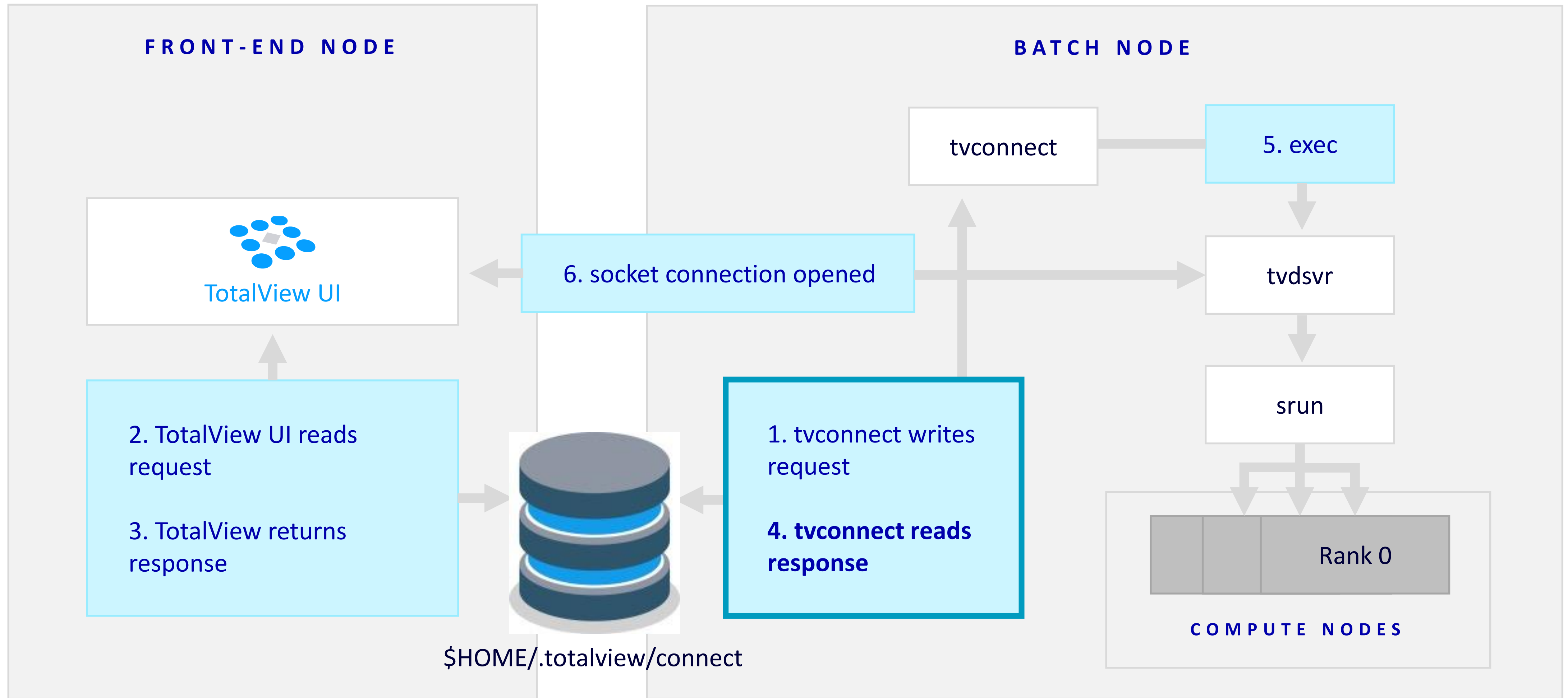
Reverse Connection Flow



© Perforce Software, Inc. All rights reserved.

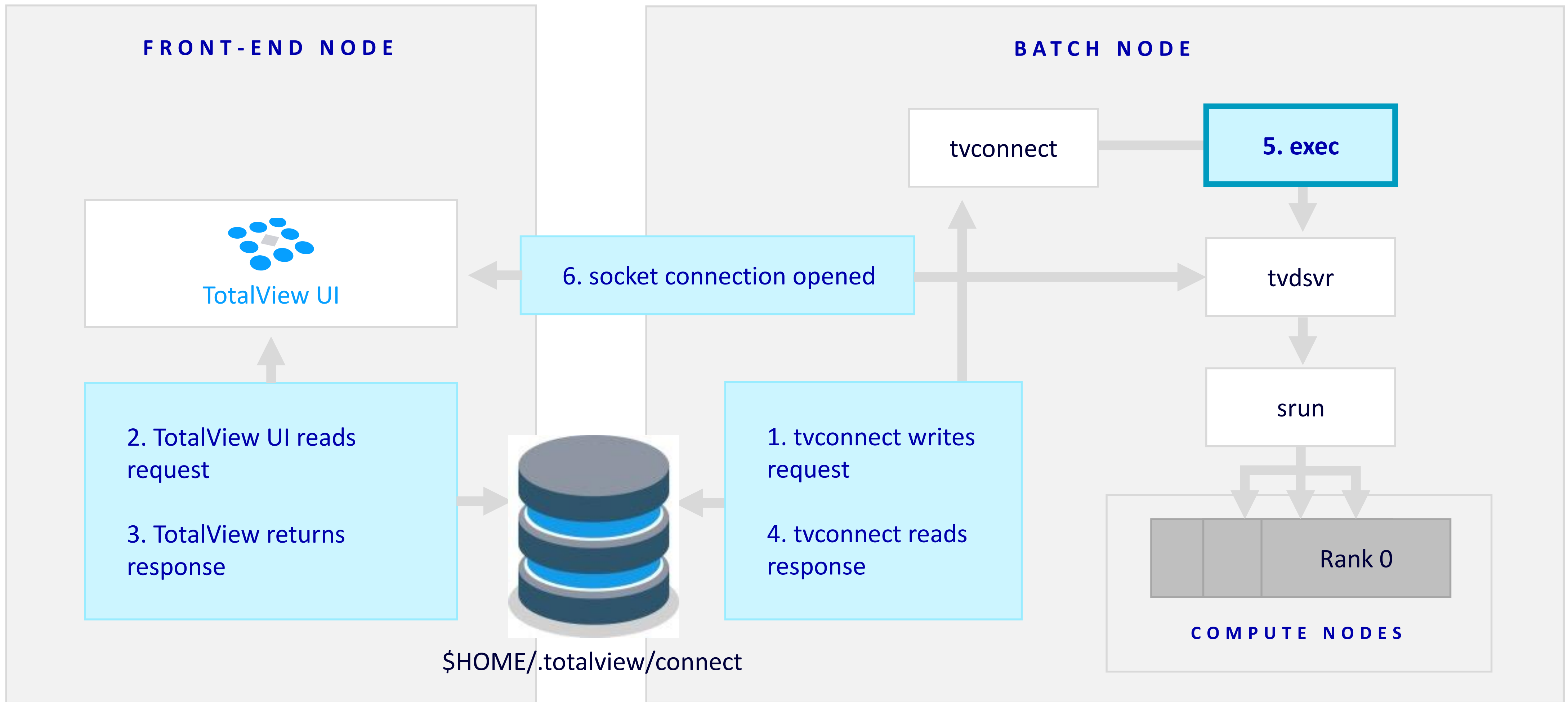
perforce

Reverse Connection Flow



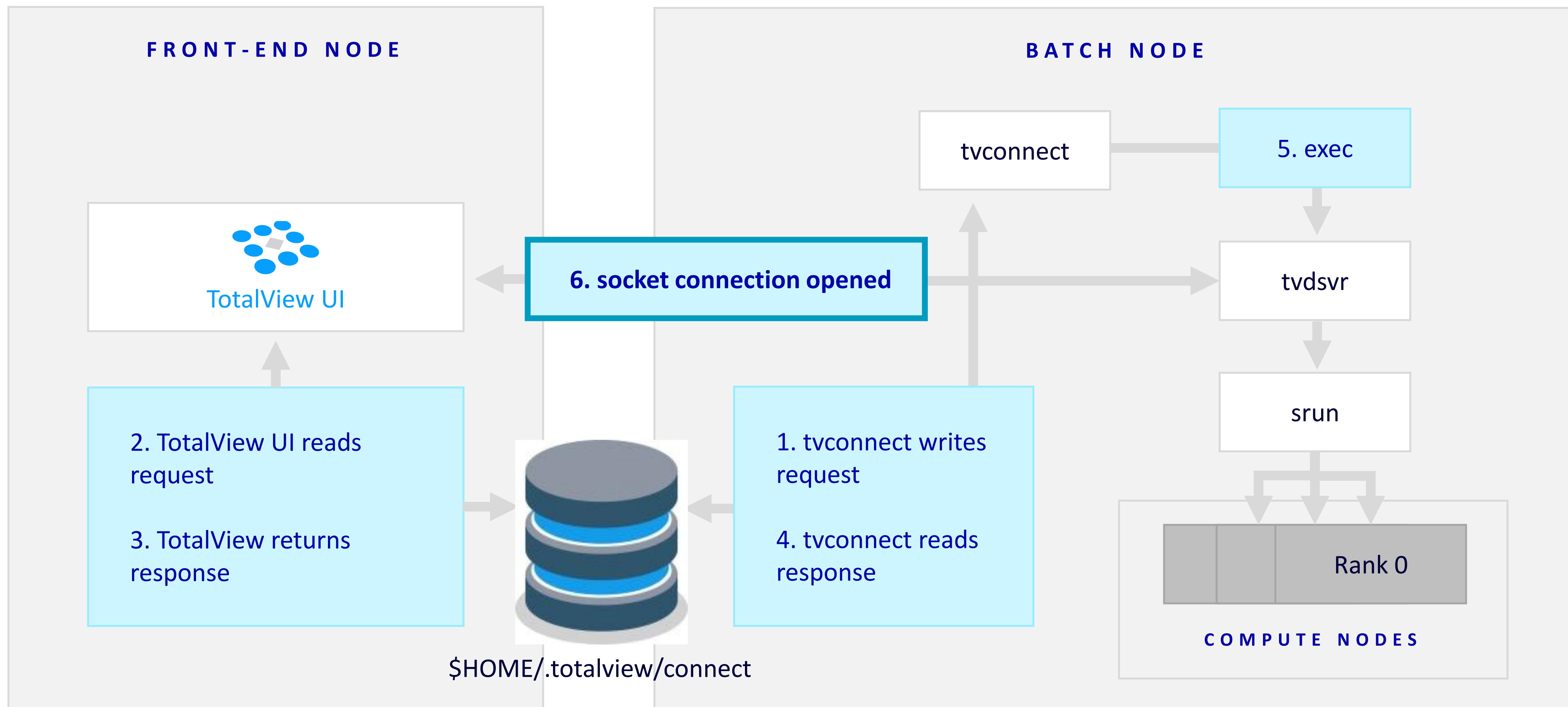
© Perforce Software, Inc. All rights reserved.

perforce



© Perforce Software, Inc. All rights reserved.

perforce



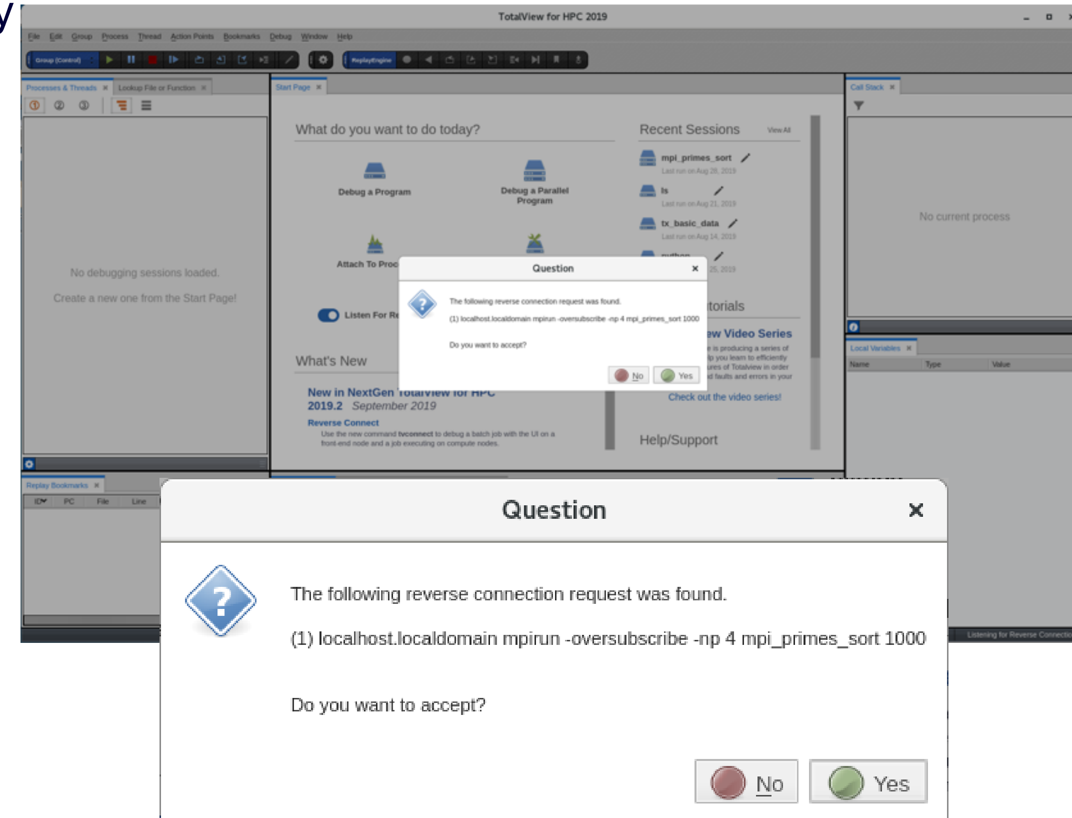
© Perforce Software, Inc. All rights reserved.

perforce

Batch Script Submission with Reverse Connect

- Start a 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
```



© Perforce Software, Inc. All rights reserved.

perforce

Parallel Debugging Group, Process and Thread Control

Select either

- Group (Control)
- Group (Share)
- Process
- Thread

File Edit Group Process Thread Action Points Debug Window Help

Group (Control)
Group (Share)
Process
Thread

Description	# P	# T	Members
Running	1	1	p1
<unknown address>	1	4	p1.1-4
1	1	4	p1
Breakpoint	4	4	0-3
exec1.cpp#15	2	2	0-1.1
6	1	1	0
7	1	1	1
exec2.cpp#15	2	2	2-3.1
8	1	1	2
9	1	1	3
<unknown line>	4	8	0-3.2-3

```
#include <mpi.h>
#include <iostream>
#include <cstring>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if (size != 4) {
        std::cerr << "This program should be run with 4 MPI ranks." << std::endl;
        MPI_Finalize();
        return 1;
    }
    if (rank == 0) {
        std::string message = "Hello from Executable 1!";
        MPI_Send(message.c_str(), message.size() + 1, MPI_CHAR, 2, 0, MPI_COMM_WORLD);
    }
    MPI_Finalize();
    return 0;
}
```

You may wish to contact your debugger vendor to inquire about support for PMIx-based debugger attachment mechanisms. Meantime, you can disable this warning by setting the OMPI_MPIR_DO_NOT_WARN envar to 1.

TotalView Debugger Server 2023.2.15 (ICCDP protocol version 101)
TotalView by Perforce Copyright 2020-2023 Perforce Software, Inc.

Rank: 0 (4980@Ubuntu-20) mpirun<exec1>.0 Thread: 0.1 (0x7f16a1c77740) - Breakpoint Frame: main File: /home/dstewart/Projects/MPMD/comm/exec1.cpp Line: 15

perforce

© Perforce Software, Inc. All rights reserved.

Understanding Complex Data Structures

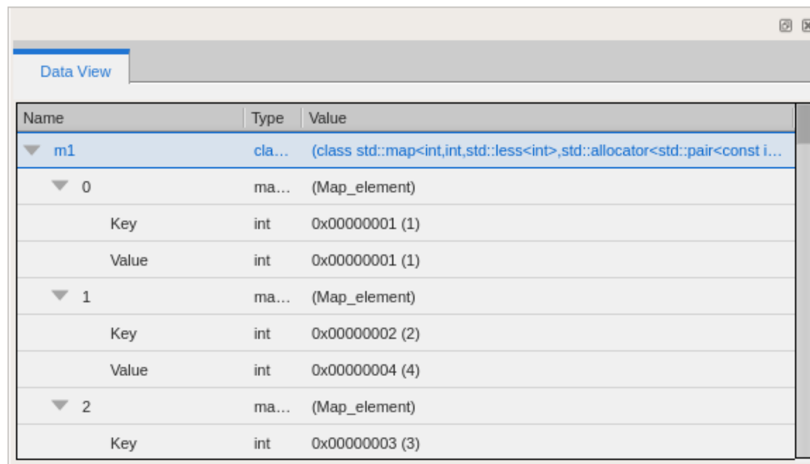
© Perforce Software, Inc. All rights reserved.

perforce

C++ Container Transformations

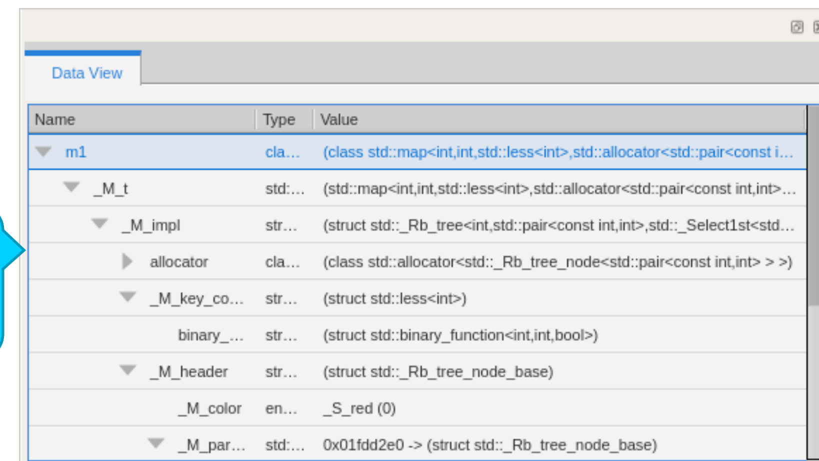
TotalView transforms many of the C++ and STL containers including:

array, vector, deque, forward_list, list, set, multiset, map, multimap, stack, queue, priority_queue, pair, tuple, unique_ptr, shared_pt, weak_ptr and others.



See This!

Instead of This



© Perforce Software, Inc. All rights reserved.

perforce

Array Statistics

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

Start Page | combined.cxx | vol

Array: vol (Thread: 1.1)

UPDATE

Slice: [:][:]

Type: double[20][20]

Statistic	Value
Count	400
Zero Count	0
Sum	597909.794
Minimum	6.28304
Maximum	7273.40418
Median	915.75308
Mean	1494.774485
Standard Deviation	1566.23066267959
First Quartile	283.91487
Third Quartile	2259.14557
Lower Adjacent Value	6.28304
Upper Adjacent Value	5195.2887
Nan Count	0
Infinity Count	0
Denormalized Count	0
Checksum	45976

© Perforce Software, Inc. All rights reserved.

perforce

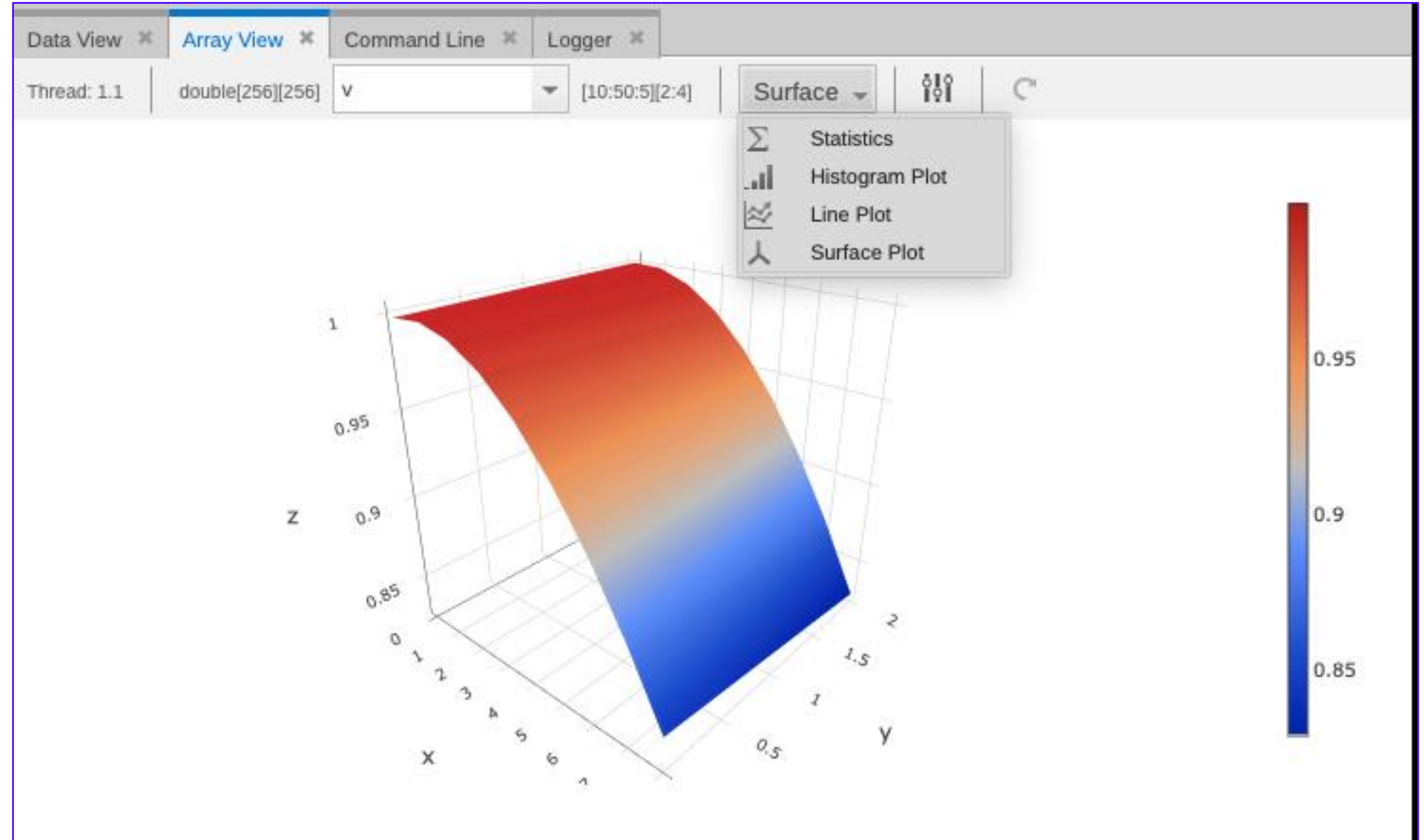
Viewing Array Data

- The Array View provides convenient array debugging and visualization features.
- The Data Table displays a 2D slice of a multi-dimensional array.
- Use the Array View Options to control the slice and stride of the array data.

© Perforce Software, Inc. All rights reserved.

Visualizing Array Data

- Select the array in the source code pane or Data View
- Right click and select ”Add to Array View”
- Select visualization from the Array View drop down
  - Statistics
  - Histogram Plot
  - Line Plot
  - Surface Plot



© Perforce Software, Inc. All rights reserved.

Debugging CUDA Applications

perforce

© Perforce Software, Inc. All rights reserved.

Debugging CUDA Applications

- NVIDIA Tesla, Fermi, Kepler, Pascal, Volta, Turing, Ampere and Hopper
- NVIDIA CUDA 9, 10, 11 and 12
  - With support for Unified Memory
- Debug 64-bit CUDA programs
- 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 types and separate memory address spaces
  - Leverages CUDA memcheck

© Perforce Software, Inc. All rights reserved.

perforce

Mixed Language C/C++ and Python Debugging

© Perforce Software, Inc. All rights reserved.

perforce

Mixed Language Python Debugging with TotalView

- 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

© Perforce Software, Inc. All rights reserved.

perforce

Python without Filtering

File Edit Group Process Thread Action Points Bookmarks Debug Window Help

Group (Control) | ReplayEngine

Processes & T... | Lookup File or F... | Doc...

Description	# P	# T	Members
python3.7-...	1	1	p1
Br...	1	1	p1
fact	1	1	p1.1
	1	1	p1.1

Start Page | python.c | tv_python_example.cpp

```
1  /* File: example.c */
2
3  #include "tv_example.h"
4
5  int fact(int n) {
6      if (n < 0){ /* This should probably return an error, but this is simpler */
7          return 0;
8      }
9      if (n == 0) {
10         return 1;
11     }
12     else {
13         /* testing for overflow would be a good idea here */
14         return n * fact(n-1);
15     }
16 }
17
18 int getSquare(int n) {
19   return n * n;
20 }
```

Call Stack

C++	fact
C++	_wrap_fact
C	_PyMethodDef_RawFastCallKeywords
C	_PyCFunction_FastCallKeywords
C	call_function
C	_PyEval_EvalFrameDefault
C	PyEval_EvalFrameEx
C	function_code_fastcall
C	_PyFunction_FastCallKeywords
C	call_function

Glue code

Data View | Command Line | Logger

Name	Type	Thread ID	Value
[Add New Expression]			

Local Variables

Name	Type	Value
Arguments		
n	int	0x00000003 (3)

Action Points | Replay Bookmarks

	ID	Type	Stop	Location
✓	1	Break	Group	dot
✓	2	Break	Group	fact

Process: 1 (2943) python3.7-dbg | Thread: 1.1 (2943) - Breakpoint | Frame: fact | File: ...jects/Python/Python Examples/tv_python_example.cpp | Line: 5

© Perforce Software, Inc. All rights reserved.

perforce

Python with filtering

File Edit Group Process Thread Action Points Bookmarks Debug Window Help

Group (Control) | ReplayEngine

Processes & T... | Lookup File or F... | Doc...

Description	# P	# T	Members
python3.7-...	1	1	p1
Br...	1	1	p1
fact	1	1	p1.1
	1	1	p1.1

Start Page | python.c | tv_python_example.cpp

```
1  /* File: example.c */
2
3  #include "tv_example.h"
4
5  int fact(int n) {
6      if (n < 0){ /* This should probably return an error, but this is simpler */
7          return 0;
8      }
9      if (n == 0) {
10         return 1;
11     }
12     else {
13         /* testing for overflow would be a good idea here */
14         return n * fact(n-1);
15     }
16 }
17
18 int getSquare(int n) {
19    return n * n;
20 }
```

Call Stack

C++	fact
Py	callFact
Py	pySupportedTypes
Py	<module>
C	main
	__libc_start_main

Data View | Command Line | Logger

Name	Type	Thread ID	Value
[Add New Expression]			

Action Points | Replay Bookmarks

	ID	Type	Stop	Location
✓	1	Break	Group	dot
✓	2	Break	Group	fact

Local Variables

Name	Type	Value
Arguments		
n	int	0x00000003 (3)

Process: 1 (2943) python3.7-dbg | Thread: 1.1 (2943) - Breakpoint | Frame: fact | File: ...jects/Python/Python Examples/tv_python_example.cpp | Line: 5

© Perforce Software, Inc. All rights reserved.

Reverse Debugging with ReplayEngine

perforce

© Perforce Software, Inc. All rights reserved.

Reverse Debugging with TotalView

- Reverse debugging provides the ability for developers to go back in execution history
- Activated either before program starts running or at some point after execution begins
- Capturing and deterministically replay execution
- Enables stepping backwards and forward by function, line, or instruction
- Run backwards to breakpoints
- Run backwards and stop when a variable changes value
- Saving recording files for later analysis or collaboration
- Reverse debugging is an add-on capability to TotalView

Start Page × common-main.c ×

```
15  static void restore_sigpipe_to_default(void)
16  {
17          sigset_t unblock;
18
19          sigemptyset(&unblock);
20          sigaddset(&unblock, SIGPIPE);
21          sigprocmask(SIG_UNBLOCK, &unblock, NULL);
22          signal(SIGPIPE, SIG_DFL);
23  }
24
25  int main(int argc, const char **argv)
26  {
27          /*
28           * Always open file descriptors 0/1/2 to avoid clobbering files
29           * in die().  It also avoids messing up when the pipes are dup'ed
30           * onto stdin/stdout/stderr in the child processes we spawn.
31           */
32          sanitize_stdfds();
33
34          git_setup_gettext();
35
36          git_extract_argv0_path(argv[0]);
37
38          restore_sigpipe_to_default();
39
40          return cmd_main(argc, argv);
41  }
42
```

© Perforce Software, Inc. All rights reserved.

perforce

Memory Debugging

© Perforce Software, Inc. All rights reserved.

perforce

Memory Debugging with TotalView

Memory Debugging Features

- Leak detection
- Dangling pointer detection
- Heap status
- Automatically detect allocation problems
- Memory Corruption Detection
- Memory Block Painting
- Memory Hoarding
- Lightweight memory block tracking

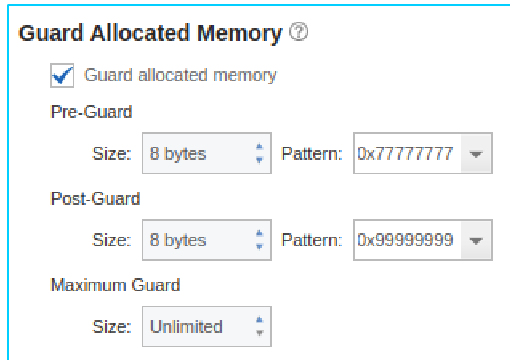
perforce

© Perforce Software, Inc. All rights reserved.

Guard Blocks

Guard allocated memory

When selected, the Memory Debugger writes guard blocks before and after a memory block that your program allocates



Pre-Guard and Post-Guard Size:

Sets the size in bytes of the block that the Memory Debugger places immediately before and after the memory block that your program allocates

Pattern:

Indicates the pattern that the Memory Debugger writes into guard blocks. The default values are 0x77777777 and 0x99999999

© Perforce Software, Inc. All rights reserved.

perforce

Memory Painting

Memory Block Painting

- [x] Paint allocations Pattern: 0xa110ca7f
- [x] Paint deallocations Pattern: 0xdea110cf

- Useful to track down when you read uninitialized or undefined memory
  - Before it has been initialized
  - After it has been freed
- Memory Painting allows you to inject known values into memory upon allocation or deallocation
  - Good values don’t correspond to any valid address
  - Have a distinctive look
    - When cast to different types

© Perforce Software, Inc. All rights reserved.

perforce

Memory Hoarding

Hoard Deallocated Memory

- [x] Hoard deallocated memory

Maximum KB to hoard: Unlimited

Maximum blocks to hoard: Unlimited

- [x] Automatically release blocks when memory gets low

Warn when hoard size drops below: 6000 KB

Hoard Status: 0 KB | 0 Blocks

- Memory Hoarding
  - Stops the memory manager from reusing memory blocks
  - Can detect certain memory errors

- Hoard Low Memory Controls
  - Automatically release hoarded memory when available memory gets low, allowing your program to run longer

- Hoard Low Memory events
  - MemoryScape can stop execution as notification that the hoard droppped below a particular threshold. This provides an indication that the program is getting close to running out of memory.

© Perforce Software, Inc. All rights reserved.

perforce

Batch Debugging with TVScript

perforce

© Perforce Software, Inc. All rights reserved.

Batch Debugging with tvscript

- A straightforward language for unattended and/or batch debugging with TotalView and/or MemoryScape
- Usable whenever jobs need to be submitted or batched
- Can be used for automation
- A more powerful version of printf, no recompilation necessary between runs
- Schedule automated debug runs with *cron* jobs
- Use in continuous integration (CI)
- Expand its capabilities using Tcl

© Perforce Software, Inc. All rights reserved.

perforce

tvscript examples

Simple example

```
tvscript \
-create_actionpoint "method1=>display_backtrace -show_arguments" \
-create_actionpoint "method2#37=>display_backtrace \
        -show_locals -level 1" \
-event_action "error=>display_backtrace -show_arguments \
        -show_locals" \
-display_specifiers "noshow_pid,noshow_tid" \
-maxruntime "00:00:30" \
~/work/filterapp /filterapp -a 20
```

MPI example

```
tvscript -mpi "Open MPI" -tasks 4 \
-create_actionpoint \
"hello.c#14=>display_backtrace" \
~/tests/MPI_hello
```

© Perforce Software, Inc. All rights reserved.

perforce

Debugging I/O Bottlenecks

© Perforce Software, Inc. All rights reserved.

perforce

Debugging I/O Bottlenecks

1. Identifying Stalled Processes (Hangs)

If your application is waiting on I/O, you can use TotalView to identify which processes are affected.

Attach to a Running Job: Use TotalView to attach to an already running, slow, or hung job. Once attached, click "Halt" to freeze the processes.

Examine Call Stacks: Look at the call stacks for various processes in the Process Window. Processes stuck in I/O will often have stacks showing functions related to file system calls (e.g., read, write, fsync) or MPI I/O functions.

Focus on Process Group: Use the "Processes and Threads" window to identify if all nodes are waiting for I/O (a potential collective I/O issue) or if only a few are (a potential load imbalance).

2. Using Memory Debugging for I/O-Related Memory Issues

I/O bottlenecks are sometimes caused by inefficient memory management, such as excessive buffering.

Detect Memory Leaks/Excessive Allocation: Use the Leak Report to see if memory is filling up, which can cause excessive paging to disk.

© Perforce Software, Inc. All rights reserved.

perforce

Debugging I/O Bottlenecks

3. ReplayEngine for Post-Mortem Analysis

If the I/O issue causes a crash or is hard to reproduce, you can use Reverse Debugging (ReplayEngine).

Record Execution: Start recording with ReplayEngine.

Step Backward: If a crash or hang occurs, step backward to identify the exact point where a large file was opened, written to, or where a communication stall initiated.

4. **Combining with Performance Tools**

TotalView is often used alongside performance tools like TAU profiler from ParaTools which provide detailed I/O statistics.

TotalView can help you connect directly to a job that has been flagged as slow by a profiler to see *exactly* what the code is doing at that moment.

© Perforce Software, Inc. All rights reserved.

perforce

Best Practices for HPC Debugging with TotalView

perforce

© Perforce Software, Inc. All rights reserved.

Best Practices for HPC Debugging with TotalView

- TotalView can’t find the program source
  - Did you compile with -g ?
  - How to adjust the TotalView search paths? Preferences -> Search Path
- Python Debugging
  - Making sure proper system debug packages are installed for Python
- Understanding different ways to stop program execution with TotalView Action Points
  - Using a watchpoint on a local variable
- Focus
  - Diving on a variable that is no longer in scope. Check the Local Variables window for in scope variables
  - Totalview doesn’t change focus to the thread hitting a breakpoint? Set Action Point Preferences to “Automatically focus on threads/processes at breakpoint”

© Perforce Software, Inc. All rights reserved.

Best Practices for HPC Debugging with TotalView

- MPI Debugging
  - Differences in launching MPI job from within the TotalView UI vs the command line.
  - TotalView runs an MPI program without stopping? Set the Parallel Preferences to “Ask What To Do” in After Attach Behavior
  - Using wrong attributes in processes and threads view
- Reverse Debugging
  - Running out of memory by not setting the maximum memory allocated to ReplayEngine
  - Defer turning on reverse debugging until later in program execution to avoid slow initialization phases
  - Adjust reverse debugging circular buffer size to reduce resources
- Memory Debugging
  - Starting with All memory debugging options enabled rather than Low
  - Not setting a size restriction for Red Zones
  - Issues with getting memory debugging turned on in an MPI job? May have to set LD_PRELOAD environment variable or worst case, prelink HIA

© Perforce Software, Inc. All rights reserved.

Best Practices for HPC Debugging with TotalView

- Reverse Connect with tvconnect
  - When I use Reverse Connect I get the following obscure message: myProgram *is an invalid or incompatible executable file format for the target platform*
  - The message indicates an incompatible file format but most often this occurs if the program provided to tvconnect for TotalView to debug cannot be found. The easiest way to resolve problem is to provide the full path to the target application, e.g., *tvconnect /home/usr/myProgram*
- How do I get help?
  - How to submit a support ticket? support-TotalView@perforce.com
  - Where is TV documentation (locally and on the internet). https://help.totalview.io/
  - Are there videos I can watch to learn how to use TotalView? https://perforce.com/resources/videos

© Perforce Software, Inc. All rights reserved.

perforce

Resources and Documentation

© Perforce Software, Inc. All rights reserved.

perforce

Resources and Documentation

totalview.io

PERFORCE

Blog Company Contact

TotalView

PRODUCTS & SOLUTIONS RESOURCES CUSTOMERS SUPPORT TRY FREE

The Most Advanced Debugger for HPC

Debug HPC Applications Written in C, C++, Fortran, and Python

TRY FREE

Why Do Top HPC Developers Use TotalView for Debugging Code?

You need special tools for multithreaded, multiprocess, and GPU-specific applications. TotalView is a powerful debugging solution that meets the unique and demanding requirements of HPC developers.

See why industry leaders use TotalView to get unprecedented HPC code visibility and control.

SEE CASE STUDIES

SEND FEEDBACK

Visit totalview.io for more information

DOCUMENTATION

https://help.totalview.io/

VIDEO TUTORIALS

https://www.perforce.com/resources/videos

BLOG

https://www.perforce.com/blog

© Perforce Software, Inc. All rights reserved.

perforce

Thank you

Name: Dean Stewart

Email: dstewart@perforce.com

Name: Bill Burns

Email: bburns@perforce.com

Name: Suzanne Horn

Email: Susanne.Horn@SMB-Net.de

© Perforce Software, Inc. All rights reserved.