Debugging with TotalView

Tim Cramer
cramer@rz.rwth-aachen.de

Why to use a Debugger?

- **If your program goes haywire, you may...**
  - → (... buy a magic wand)
  - → ... read the source code again and again and ...
  - → ... enrich your application with printf's

- **OR**
  - → Use an adequate tool – a debugger.
  - → Debuggers will enhance your productivity.

What is TotalView?

A comprehensive debugging solution for demanding parallel and multi-core applications

Wide compiler & platform support

- **C, C++, Fortran 77 & 90**, UPC
- **Linux**, OS X, Unix
- Windows frontend (client)

Handles concurrency

- **multi-threaded debugging**
- **parallel debugging**
  - **MPI**, PVM, others
- remote and client/server debugging

Integrated Memory Debugging

Reverse Debugging available

- ReplayEngine

Supports a Variety of Usage Models

- **powerful and easy GUI visualization**
- CLI for scripting
- Long distance remote debugging
- Unattended batch debugging /

GUI-free debugging with TVScript

About this Session

- **TotalView is a GUI-based Debugger**
  - → I don’t like slides with one screenshot after the other
  - → I want to do this as interactive as possible
  - → If you want: Start your Laptops and login
  - → Login to RWTH Compute Cluster using the NX client
  - → All examples should be distributed in the hpclabXX accounts
  - → Slides for interactive sessions will be online with screenshots

Before you start

- **Lean back and relax**
- **Check your environment**

  → increase ulimits (ulimit -a)

  -s Stack size (crucial for Fortran and OpenMP!)

  -t CPU time

  -v Address space and others

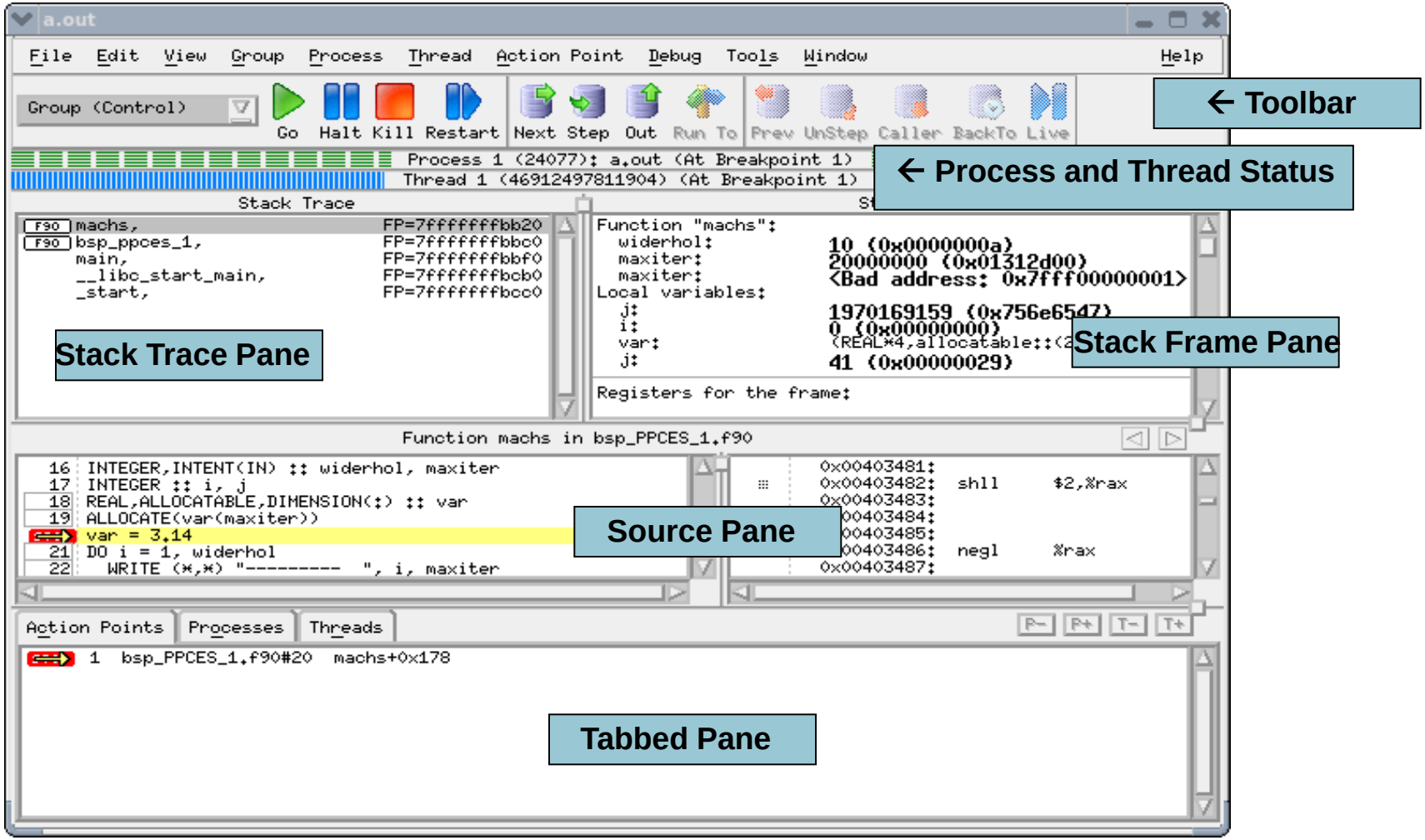
  -c Core file size (crucial for debugging on core files)

- **Remove all objects and intermediate files**
- **Rebuild with debugging info ON (-g)**
  optimization OFF (-O0)
- **Problem still here? Use a debugger!**

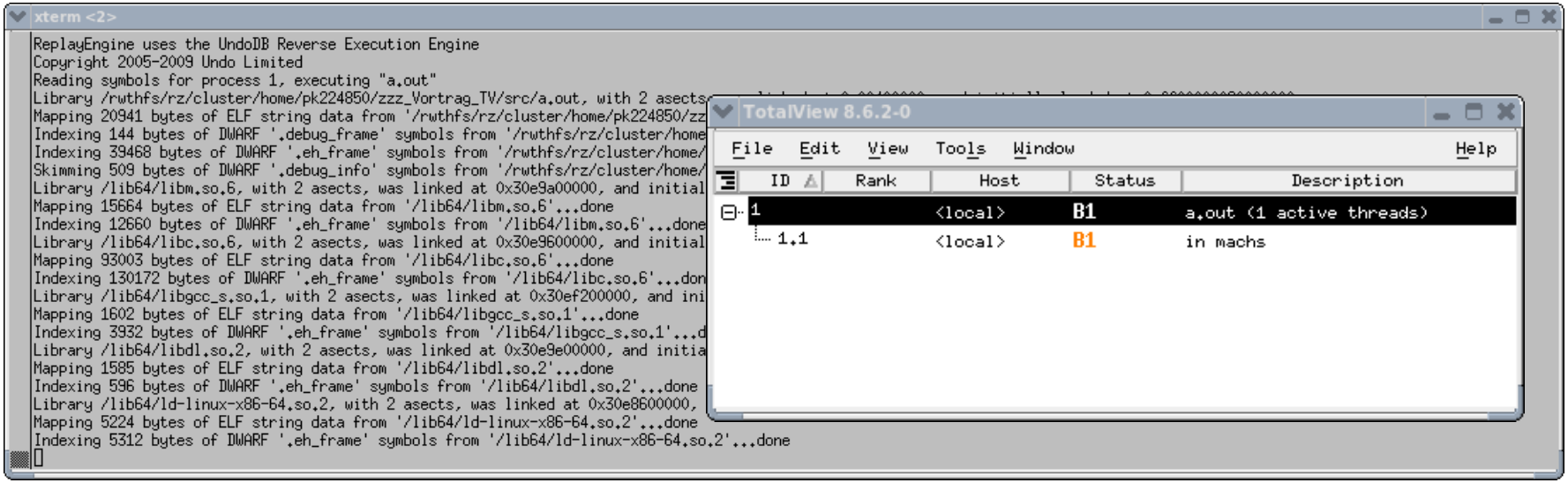
TotalView in the RWTH Environment

- **Initialize the environment and startup:**
  - → `$ module load totalview`
  - → `$ totalview`
- **or load a binary directly (here called a.out):**
  - → `$ totalview a.out -a <options of a.out>`
- **Main modes:**
  - → Start a new process
  - → Attach to a running process
  - → Load a core file (Post-Mortem)

Process Window



Root Window and Console

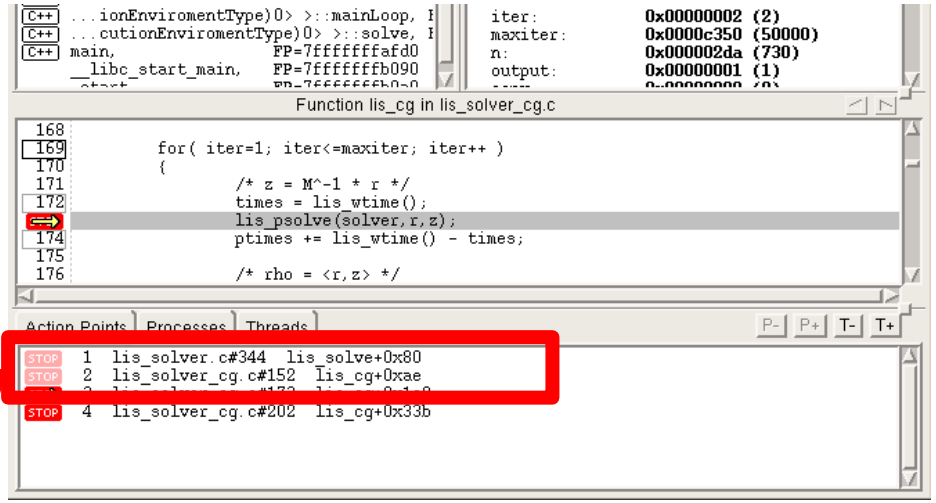


Status Info

- B = Breakpoint
- E = Error
- W = Watchpoint
- R = Running
- M = Mixed
- T = Stopped

Break- and Watchpoints

- **Breakpoints (just presented)**
  - → Interrupt execution when reaching a specific code line
  - → Conditional Breakpoints possible
  - → Set by clicking in the source pane
  - → Temporary disabling is possible



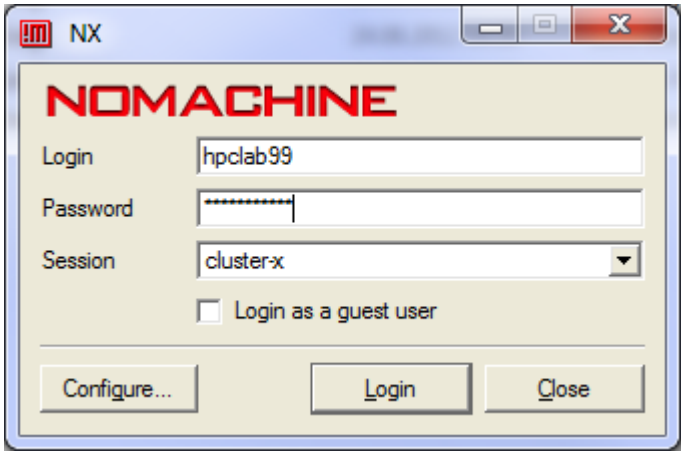
- **Watchpoints**
  - → Interrupt when a change occurs to a specific memory location
  - → Conditional watchpoints possible

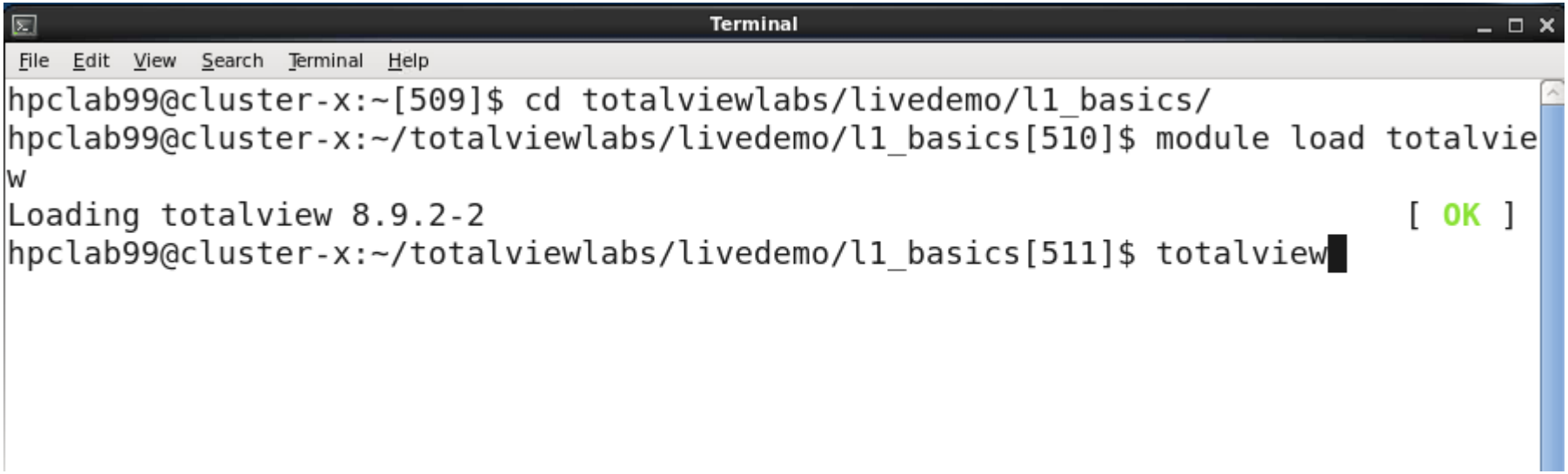
    (e.g. only stop if the sign of the value changes or specified threshold reached)

Demo 1: TotalView Startup and Basics

Live Demo Startup

L1: Login with NX & Startup TotalView





L1: TotalView Window

Terminal

File Edit View Search Terminal Help

```
hpclab99@cluster-x:~[509]$ cd totalviewlabs/livedemo/l1 basics/
hpclab99@cluster-x:~/totalviewlabs/livedemo/l1_
w
Loading totalview 8.9.2-2
hpclab99@cluster-x:~/totalviewlabs/livedemo/l1_
Linux x86_64 TotalView 8.9.2-2
Copyright 2010-2012 by Rogue Wave Software Inc.
Copyright 2007-2010 by TotalView Technologies,
Copyright 1999-2007 by Etnus, LLC.
```

TotalView 8.9.2-2

File Edit View Tools Window Help

ID	Rank	Host	Status	Description

New Program

Start a new process

Attach to process

Open a core file

Program | Arguments | Standard I/O | Parallel

Program: Browse...

On host: (local) Add Host...

Enable ReplayEngine
Record all program state while running. Roll back your progr

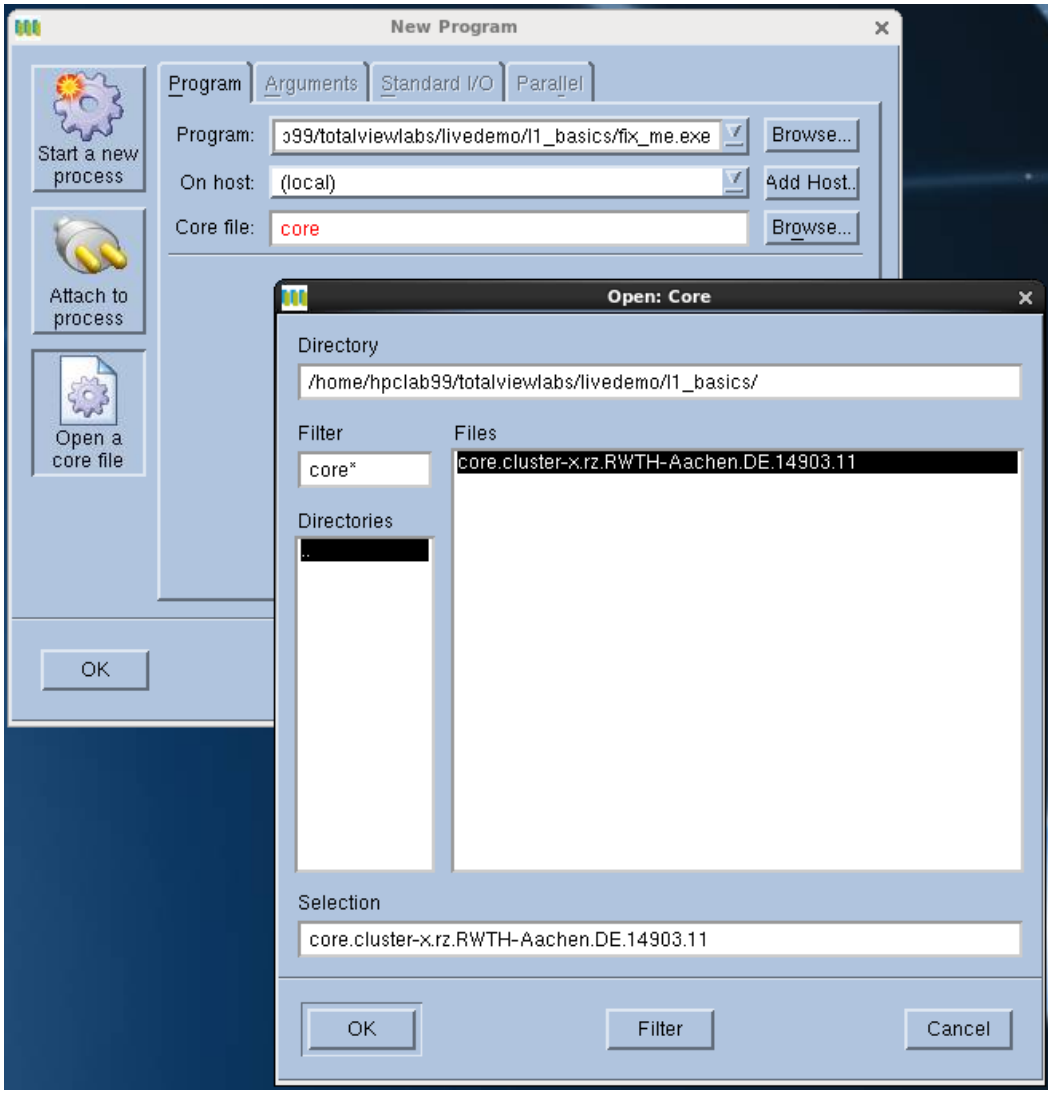
Enable memory debugging
Track dynamic memory allocations. Catch common errors, leaks, and shc

Halt on memory errors

Enable CUDA memory checking
Detect global memory addressing violations and misaligned g

OK Cancel Help

L1: Post-Mortem Debugging (1/2)



L1: Post-Mortem Debugging (2/2)

fix_me.exe

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) | Go Halt Kill Restart | Next Step Out Run To | GoBack Prev UnStep Caller BackTo Live

Process 1 (13811): fix_me.exe (Error)

Thread 1 (13811) (Error) <Segmentation violation>

Stack Trace

```
init,                FP=7fff3b84b880
main,                FP=7fff3b84b8c0
__libc_start_main,   FP=7fff3b84b980
_start,              FP=7fff3b84b990
```

Stack Frame

```
Function "init":
  a:                    0x00601010 -> 0
  count:                0x000186a0 (100000)
Local variables:
  i:                    0x000071fe (29182)

Registers for the frame:

      %rax: 0x0063a000 (6529024)
      %rdx: 0x000186a0 (100000)
      %rcx: 0x00000001 (1)
```

Function init in fix_me.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void init(double* a, const int count){
5      int i;
6
7      for (i=0; i<count; i++){
8          a[i] = (double)i;
9          printf("Init a[%d] with %f\n", i, a[i]);
10     }
11 }
12
13 int main(int argc, char **argv){
14     const int count  = 100000;
15     double* a;
16
17     a = (double*)malloc(count);
18
19     init(a, count);
```

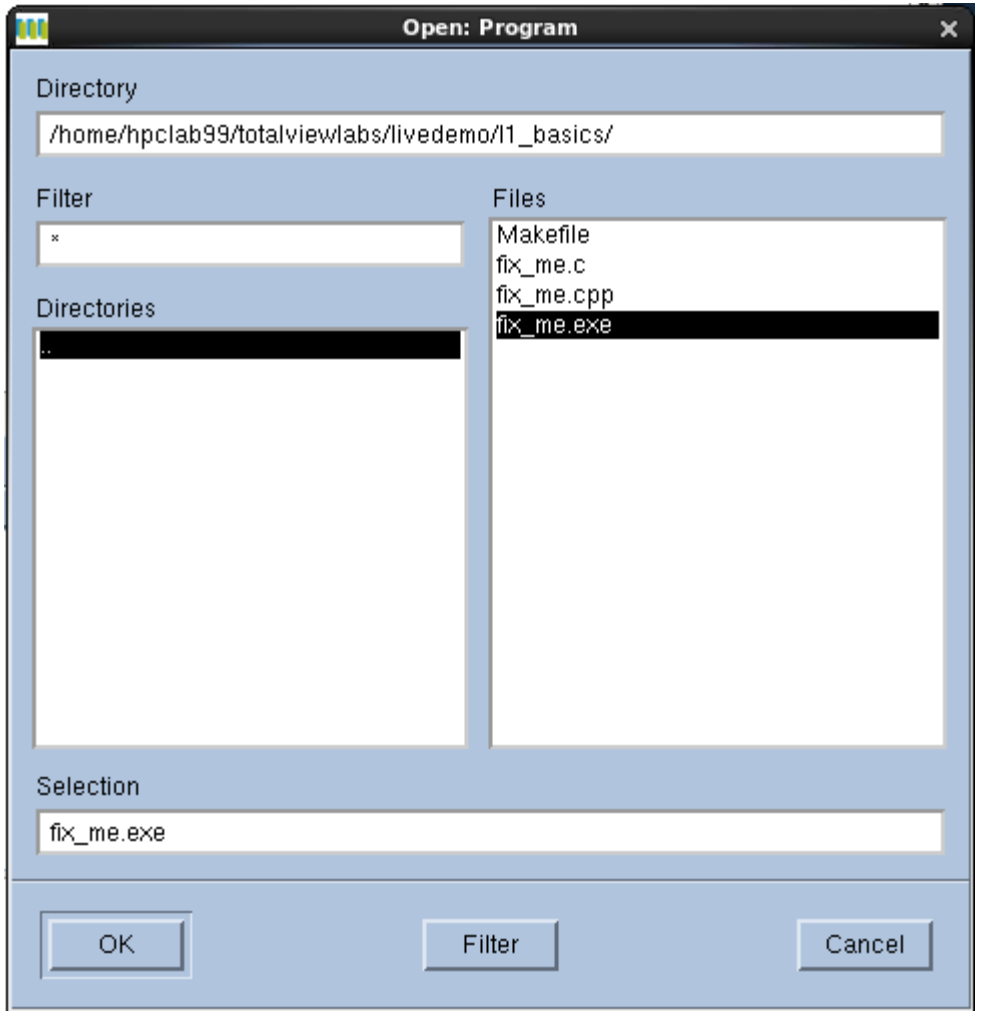
TotalView 8.9.2-2

File Edit View Tools Window Help

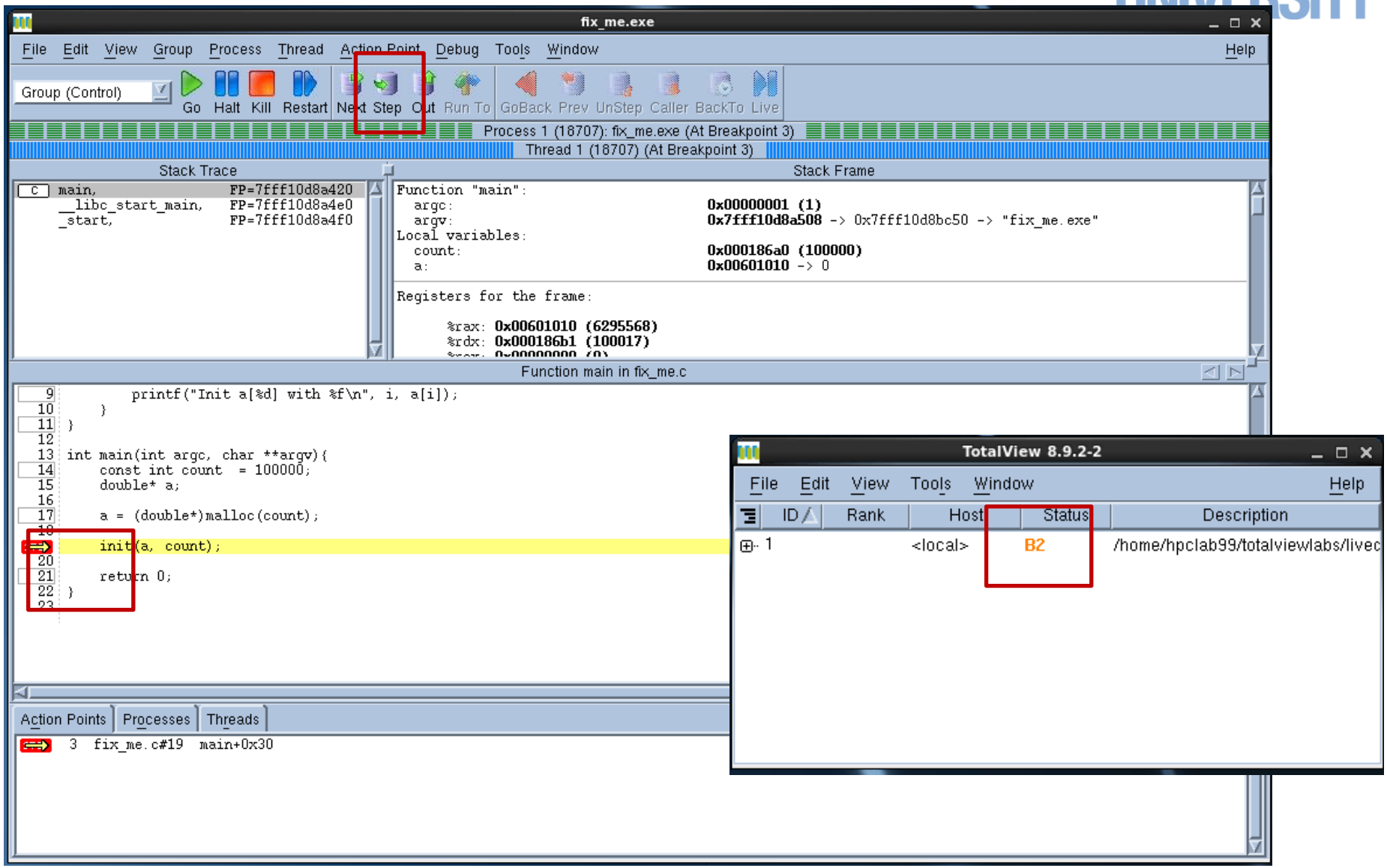
ID	Rank	Host	Status	Description
1		<local>	E	fix_me.exe (1 active threads)

Action Points | Processes | Threads

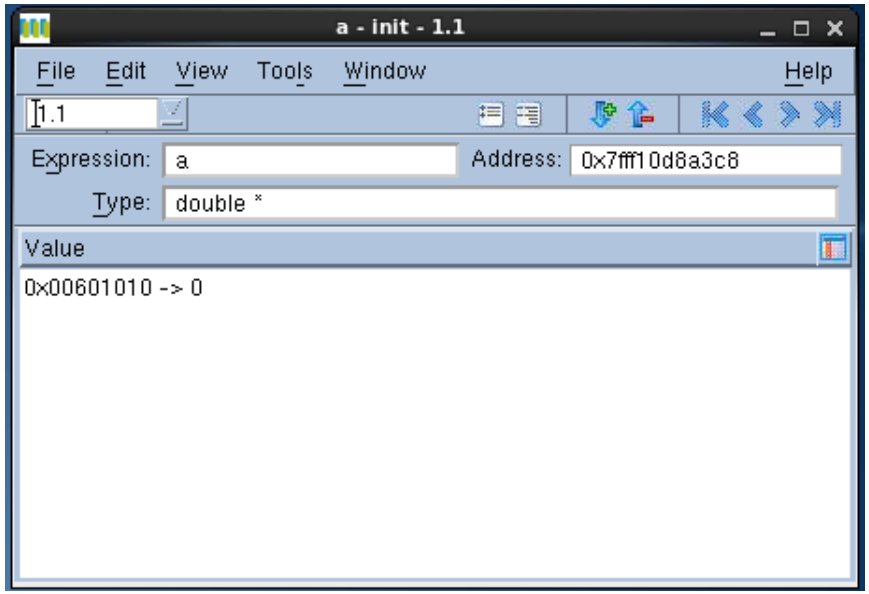
L1: Running within TotalView

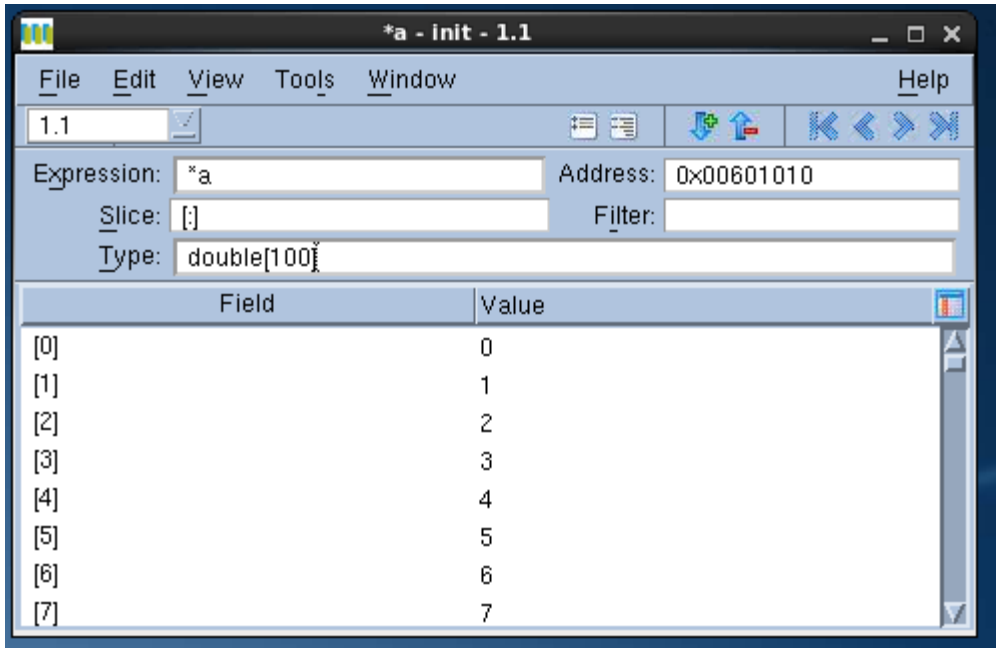


Setting a breakpoint



Inspecting an array in C/C++

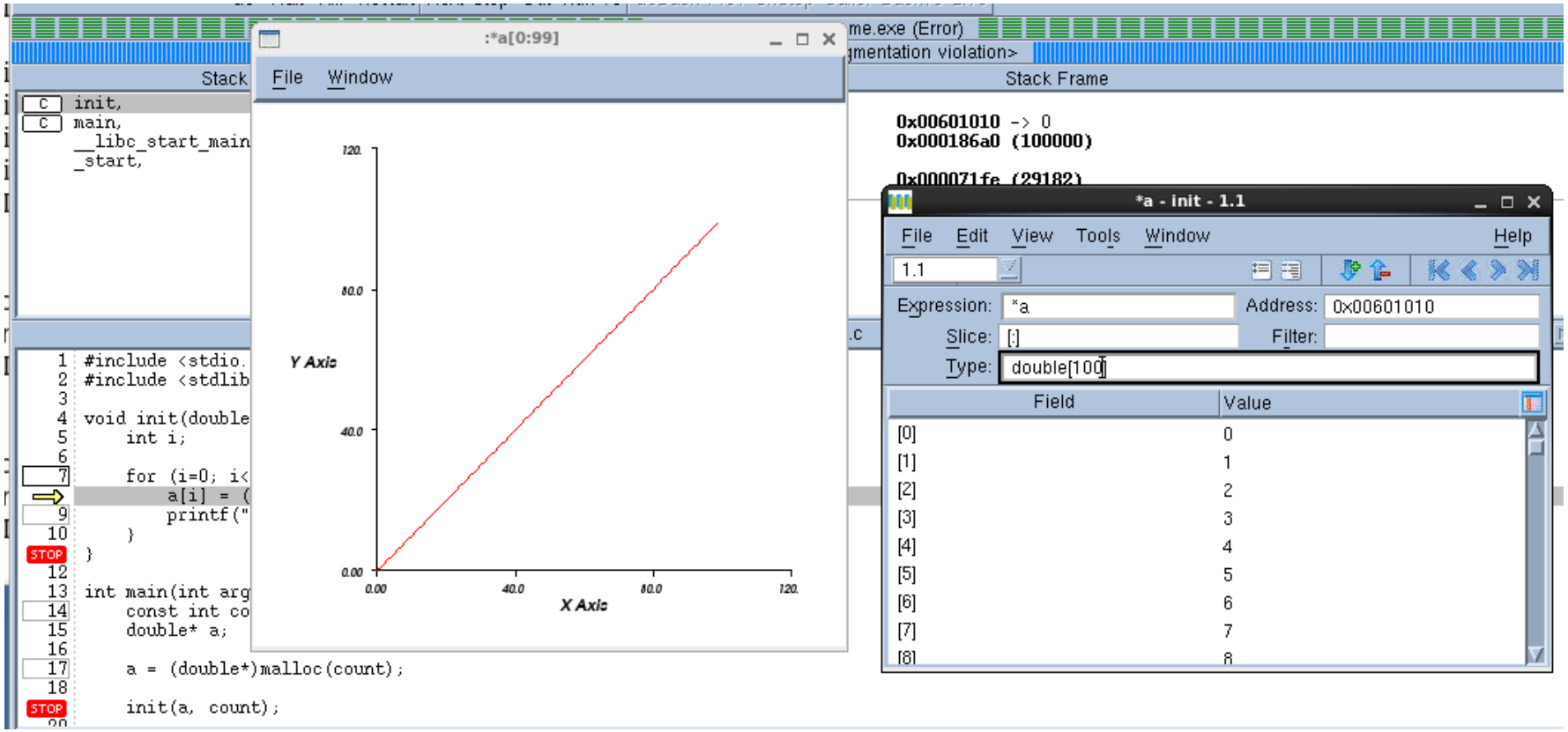




- **Typecast necessary**

Data Visualizations

- **Helpful for big data arrays**

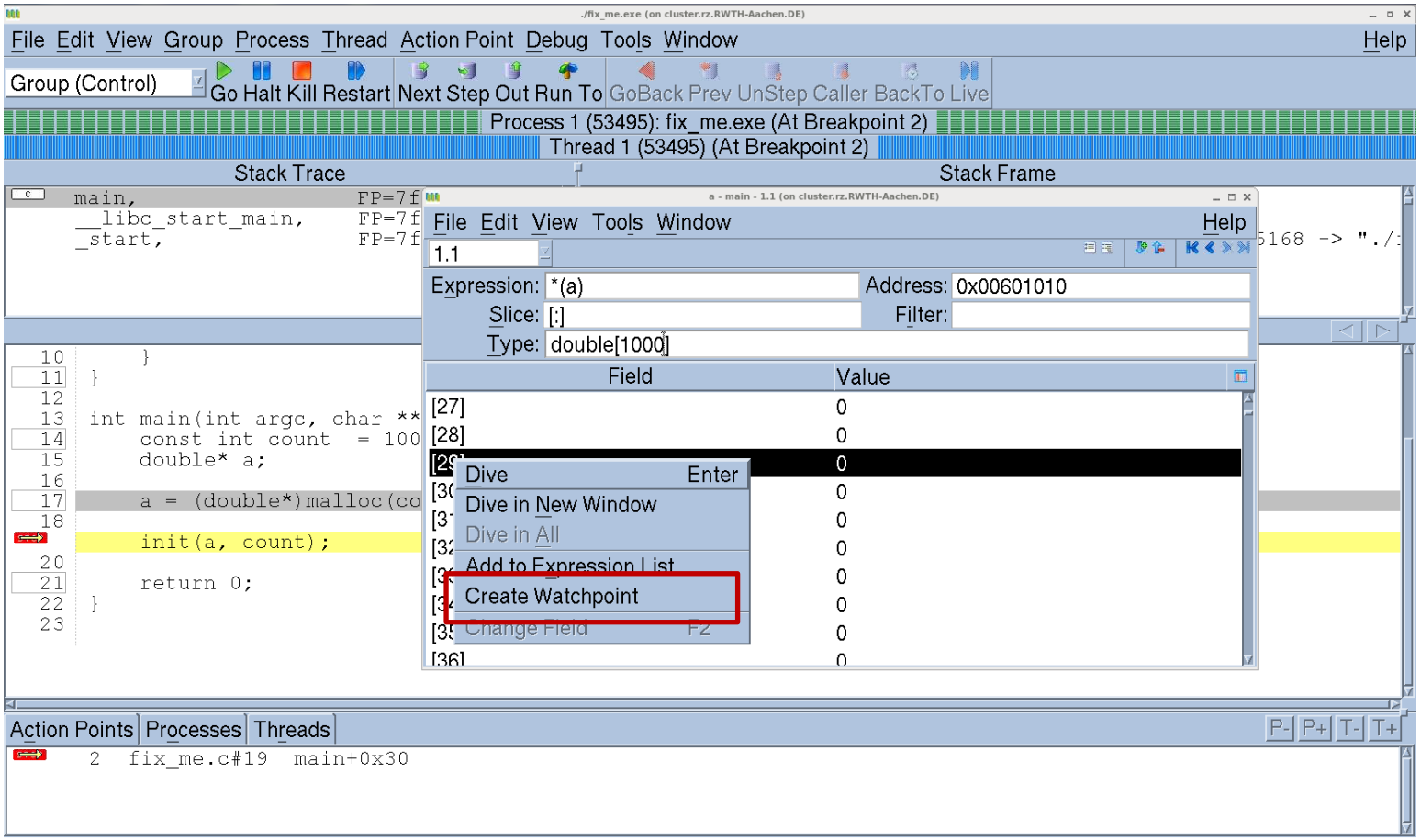


Demo 2: TotalView Watchpoints

Live Demo Watchpoints

L2: Watchpoints (1/2)

- **Create a watchpoint for a[29]**



L2: Watchpoints (2/2)

- **Will interrupt as soon as `a[29]` changes**

./fix_me.exe (on cluster.rz.RWTH-Aachen.DE)

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Process 1 (53495): fix_me.exe (At Watchpoint 5)

Thread 1 (53495) (At Watchpoint 5) <Watchpoint Trap>

Stack Trace

```
init,                FP=7fff89532800
main,                FP=7fff89532840
__libc_start_main,   FP=7fff89532900
_start,              FP=7fff89532910
```

Stack Frame

Function "init":

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void init(double* a, const int count){
5      int i;
6
7      for (i=0; i<count; i++){
8          a[i] = (double)i;
⇒          printf("Init a[%d] with %f\n",
10     }
11 }
12
13 int main(int argc, char **argv){
14     const int count  = 100000;
15     double* a;
16
17     a = (double*)malloc(count);
```

a - main - 1.1 (on cluster.rz.RWTH-Aachen.DE)

File Edit View Tools Window Help

1.1

Expression: *(a) Address: 0x00601010

Slice: [:] Filter:

Type: double[1000]

Field	Value
[27]	27
[28]	28
[29]	29
[30]	0
[31]	0
[32]	0
[33]	0
[34]	0
[35]	0
[36]	0

Action Points Processes Threads P- P+ T- T+

```
STOP  2  fix_me.c#19  main+0x30
      5  8 bytes @ 0x006010f8 ((a)[29])
```

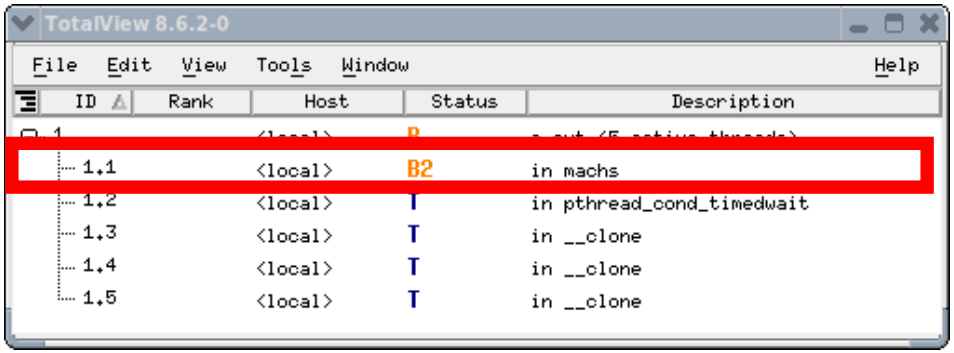
Parallel Debugging with TotalView

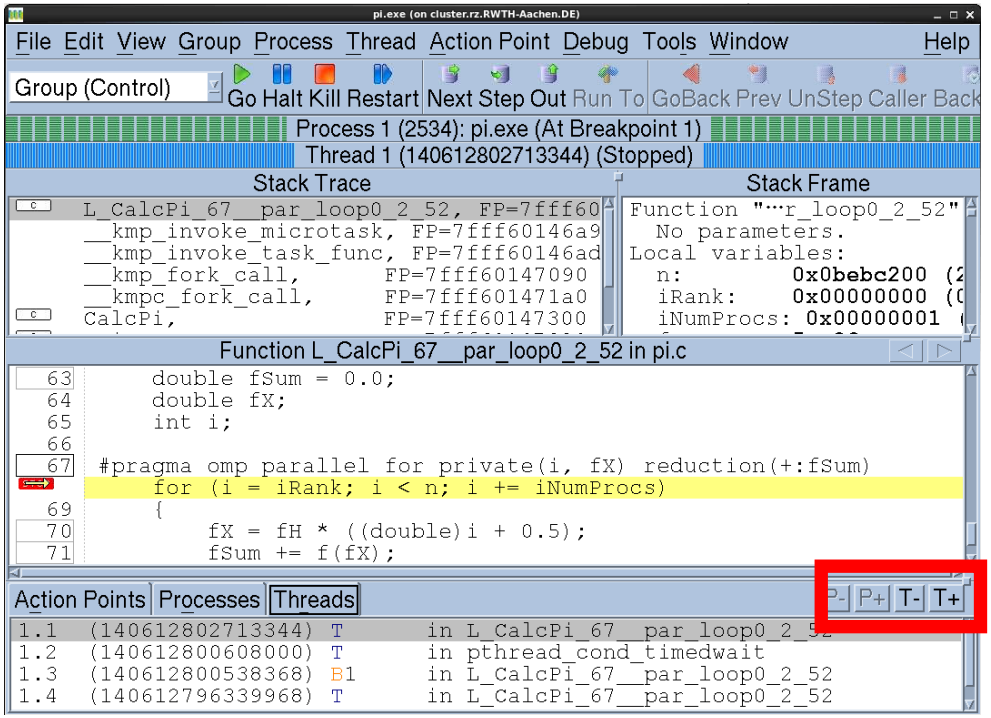
- **Parallel Debugging might be very hard**
  - → Try to debug a *serial* version of the program first!
  - → Typical multithreading errors may not be found (e.g., *race conditions*)
  - → Some errors only occur
    - → with optimized code (uninitialized variables?)
    - → with many processes
    - → outside of debug sessions (different timing)

- **Nevertheless, TV is better than no TV**
  - → Stack frames for every process / thread in one GUI
  - → Switching between processes / threads (even for accelerators like GPGPUs)
  - → Variable inspection (and visualization) across all processes / threads
  - → Deadlock detection
  - → Visualization of the MPI message queue

OpenMP and TotalView (1/2)

- **Overview of thread number**
  - → Number of threads in root window
  - → some OpenMP implementations use an additional system thread
- **Thread state and navigation**
  - → Step into a parallel region is not possible -> Set a breakpoint
  - → Easy switching between the threads with T- / T+

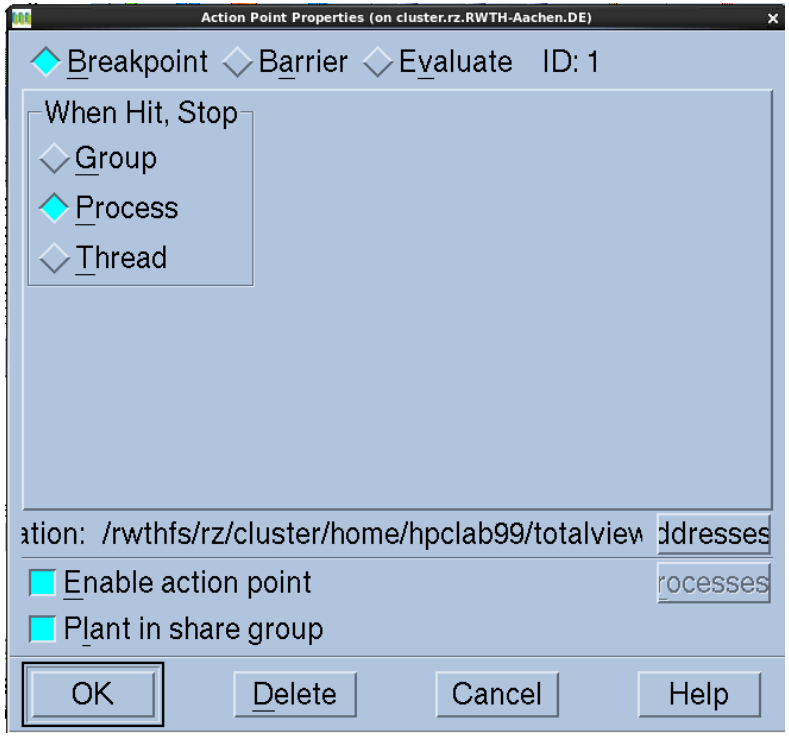




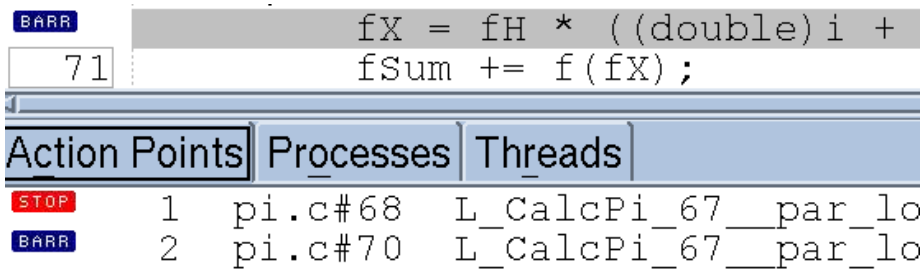
OpenMP and TotalView (2/2)

- **Breakpoint properties and barriers**
  - → Default behavior: Same breakpoint for all threads
    - → Change “Properties”

Group	Stop all threads of all processes
Process	Stop all threads of one process (default)
Thread	Stop only the current thread



  - → Barriers
    - → Similar to breakpoints
    - → Help to synchronize



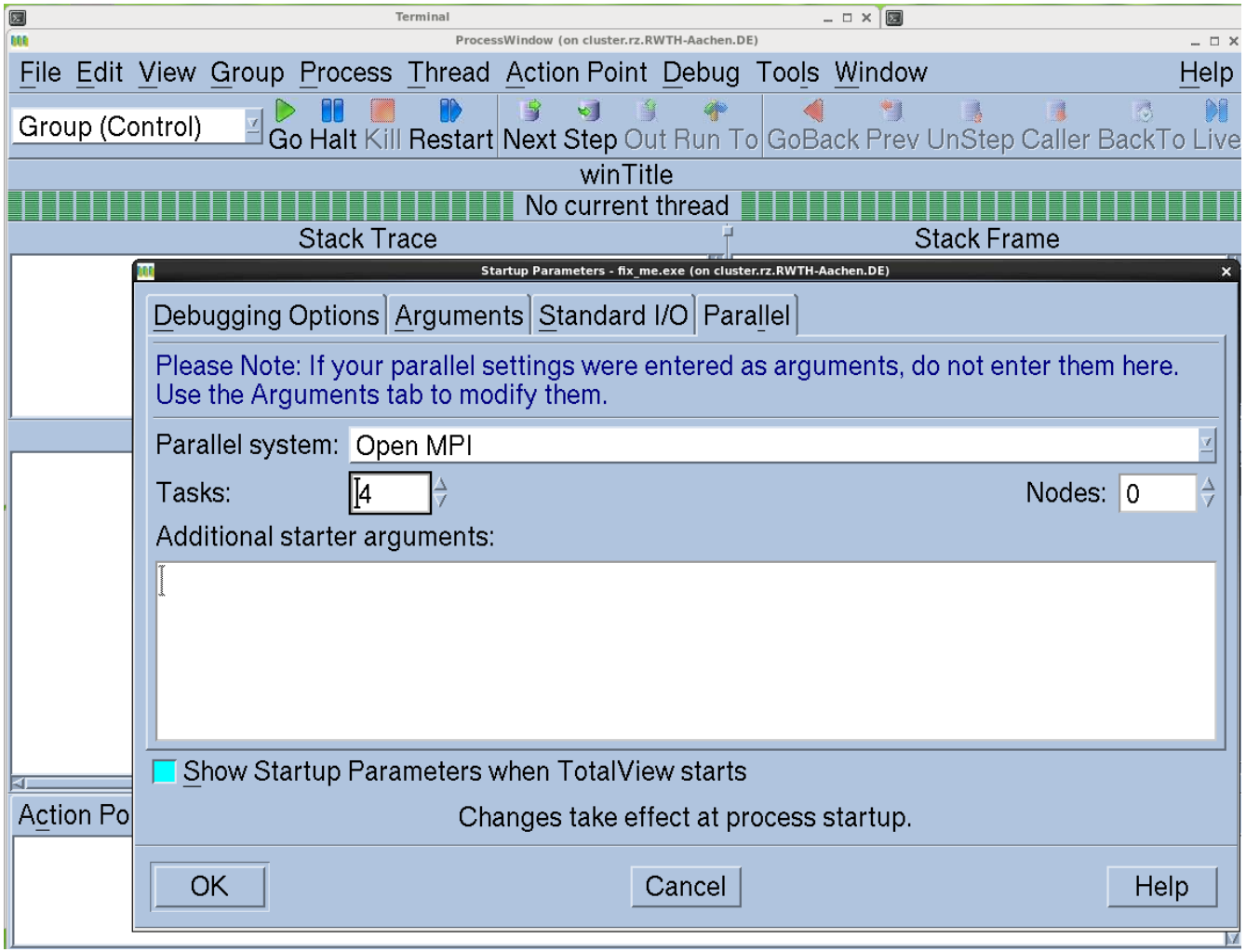
MPI and TotalView

- **Two startup methods for MPI jobs**
  - → New launch: `$ totalview mpi-a.out`
    - → Set parameters in GUI
    - → Easy and intuitive
    - → No detaching or re-attaching possible
    - → Not available on all platforms
  - → Classic lauch: `$ mpiexec -tv -np 4 mpi-a.out`
    - → Arguments depend on MPI vendor
    - → Attach / Attach to subset / Detache / Reattache possible
    - → Warning: Not possible @ RZ at the moment

Demo 3: Parallel Debugging with MPI

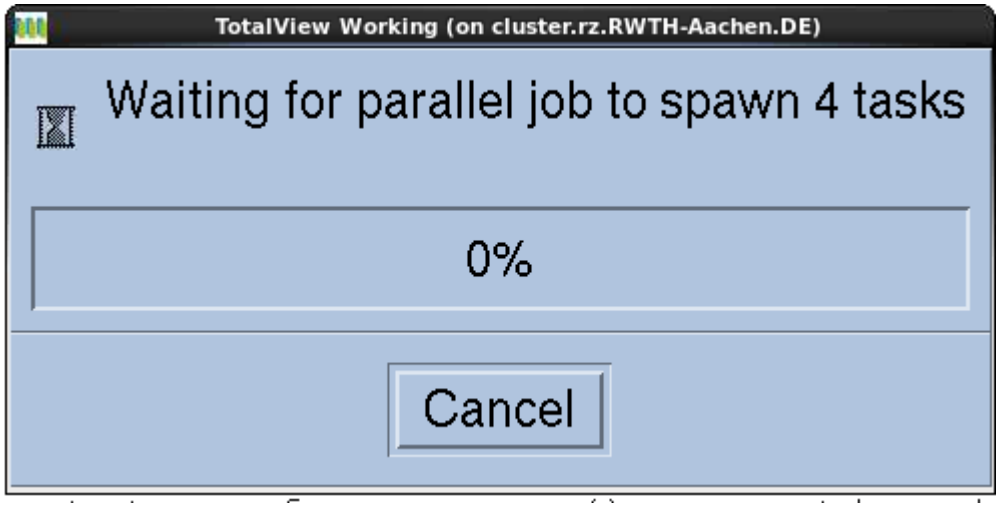
Live Demo Parallel Debugging

L3: MPI Debugging

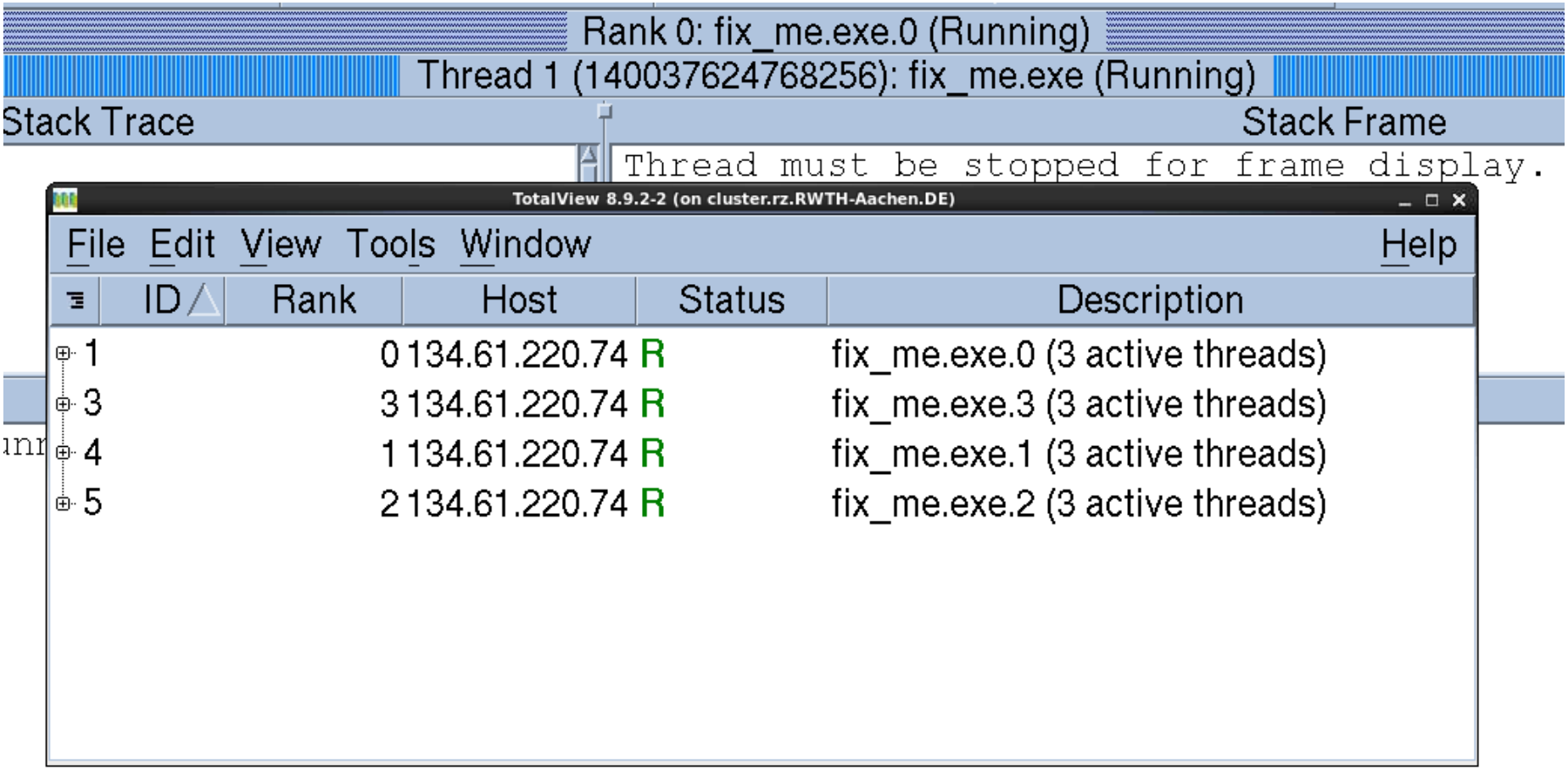


L3: MPI Debugging

- **Be patient**

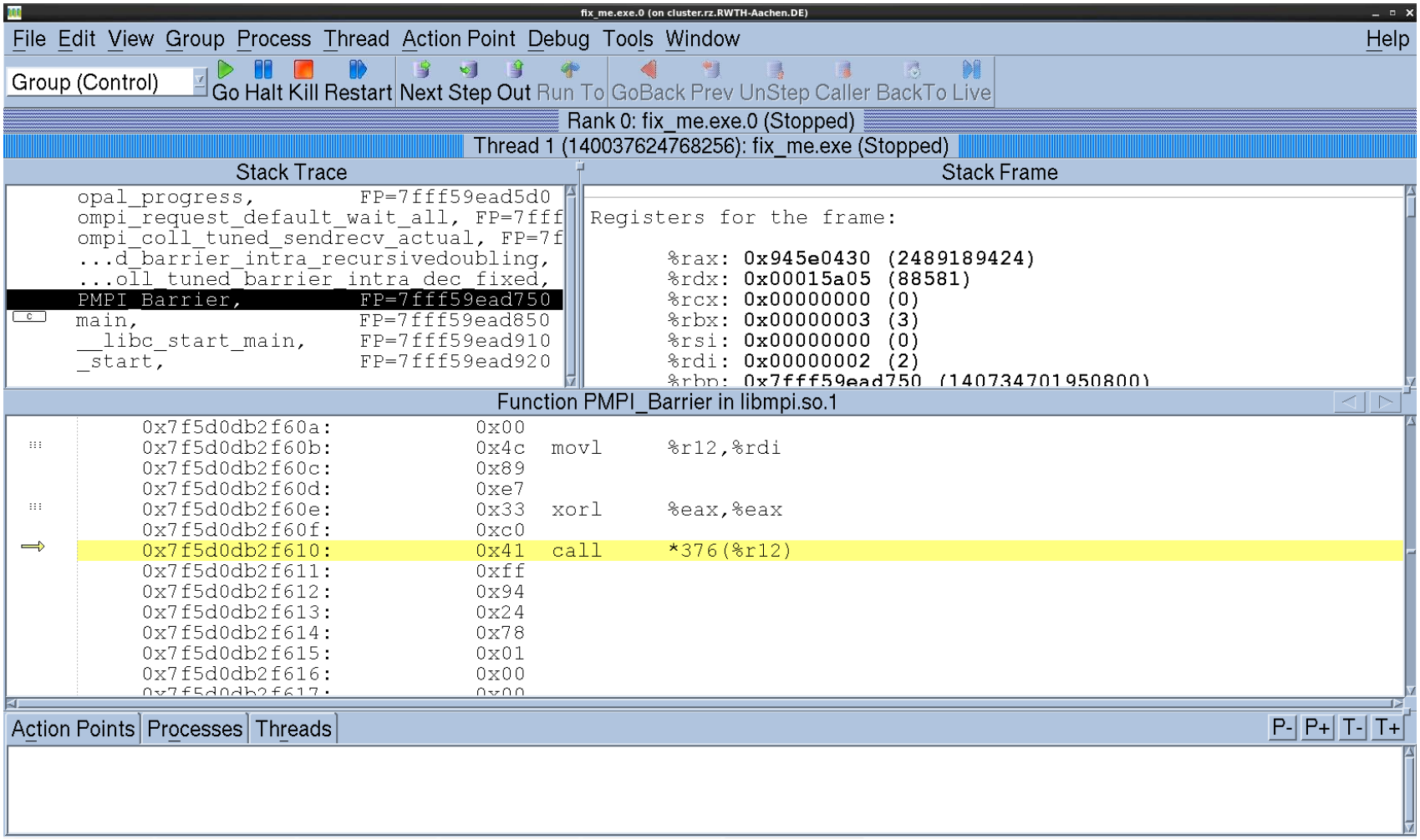


L3: MPI Debugging



L3: MPI Debugging

- **Press the “HALT” button**



L3: MPI Debugging

- **Rank 0 is waiting in barrier**

fix_me.exe.0 (on cluster.rz.RWTH-Aachen.DE)

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Rank 0: fix_me.exe.0 (Stopped)

Thread 1 (140037624768256): fix_me.exe (Stopped)

Stack Trace

```
...d_barrier_intra_recursivedoubling,
...oll_tuned_barrier_intra_dec_fixed,
PMPI_Barrier,          FP=7fff59ead750
main,                  FP=7fff59ead850
__libc_start_main,     FP=7fff59ead910
_start,                FP=7fff59ead920
```

Stack Frame

```
Function "main":
  argc:                0x00000001 (1)
  argv:                0x7fff59ead938 -> 0x7fff59eaf157 -> "fi
Local variables:
  nprocs:              0x00000004 (4)
  myrank:              0x00000000 (0)
```

Function main in fix_me.c

```
35              MPI_Send(&data, count, MPI_INT, i, tag, MPI_COMM_WORLD);
36          }
37      }
38      else {
39          MPI_Recv(&data, count, MPI_INT, 0, tag2, MPI_COMM_WORLD, &status);
40      }
41
⇒       MPI_Barrier(MPI_COMM_WORLD);
43
44      /* Check the data everywhere. */
45
46                  if (myrank != 0){
47          for (i = 0; i < count; i++){
48              if (data[i] != 399) {
49                  good_data = 0;
50              }
51          }
52          if (good_data == 0){
```

Action Points | Processes | Threads P- P+ T- T+

L3: MPI Debugging

- **Rank 1 still waits for data**

fix_me.exe.1 (on cluster.rz.RWTH-Aachen.DE)

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Rank 1: fix_me.exe.1 (Stopped)

Thread 1 (140237385426688): fix_me.exe (Stopped) <Stop Signal>

Stack Trace

```
mca_pml_ob1_recv,      FP=7fff987f87b0
PMPI_Recv,             FP=7fff987f8800
main,                  FP=7fff987f8910
__libc_start_main,     FP=7fff987f89d0
_start,                FP=7fff987f89e0
```

Stack Frame

```
Function "main":
  argc:                0x00000001 (1)
  argv:                0x7fff987f89f8 -> 0x7fff987fa157 -> "fi
Local variables:
  nprocs:              0x00000004 (4)
```

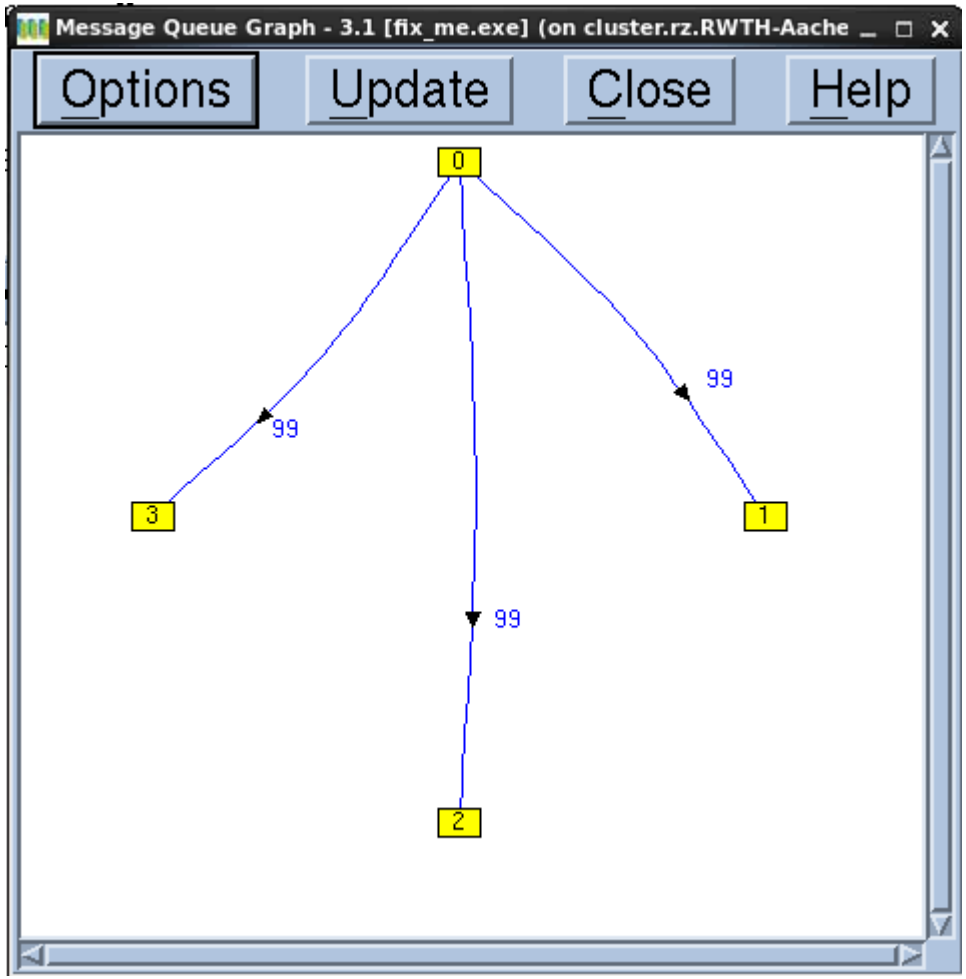
Function main in fix_me.c

```
30          } else
31              data[i] = i;        /* initialize all other data buffers from 0 to count */
32
33      if (myrank == 0) {
34          for (i = 1; i < nprocs; i++) {
35              MPI_Send(&data, count, MPI_INT, i, tag, MPI_COMM_WORLD);
36          }
37      }
38      else {
        MPI_Recv(&data, count, MPI_INT, 0, tag2, MPI_COMM_WORLD, &status);
40      }
41
42      MPI_Barrier(MPI_COMM_WORLD);
43
44      /* Check the data everywhere. */
45
46                  if (myrank != 0){
47          for (i = 0; i < count; i++){
```

Action Points | Processes | Threads P- P+ T- T+

L3: MPI Debugging

- **Message Queue Graph**
  - → Rank 1, 2, 3 are waiting for data on tag 99



Debugging of Large MPI Jobs

- **Limitations**
  - → Each MPI process consumes a TotalView license token
  - → RWTH has only 50 licenses
  - → Try to debug a small example
- **Debugging of subset of the whole job**
  - → Attaching to subset possible

    "File"->"Preferences"->"Parallel"

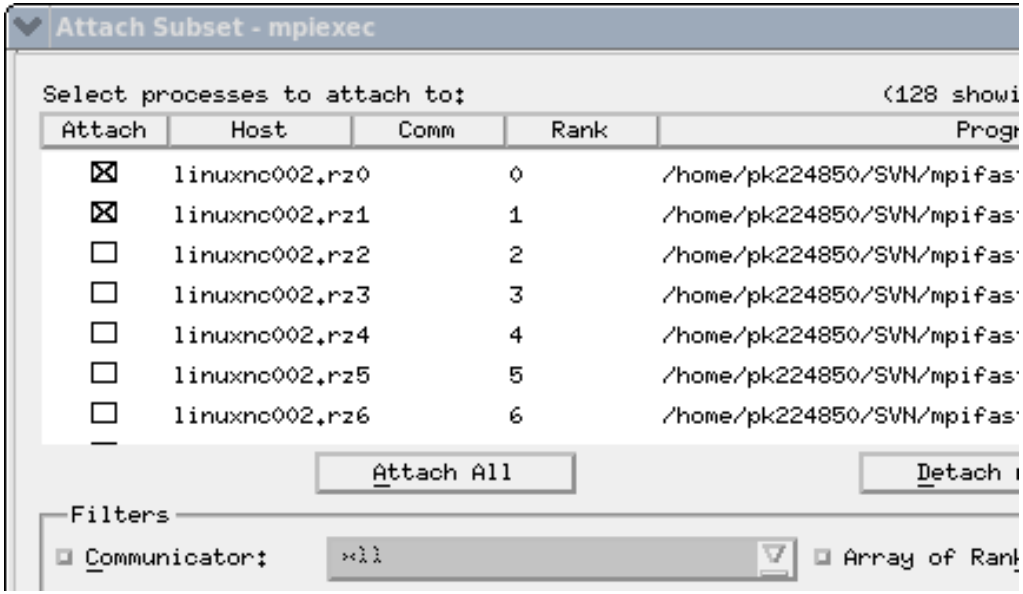
    "When a job goes parallel" menu set:

    [x] on "Ask what to do"

    (instead of "Attach to all")

    Choose a subset of processors in

    "Group" -> "Attach Subset"



Message Queue Graph

"Tools" → "Message Queue Graph"

Hangs & Deadlocks

Pending Messages

Communication Patterns

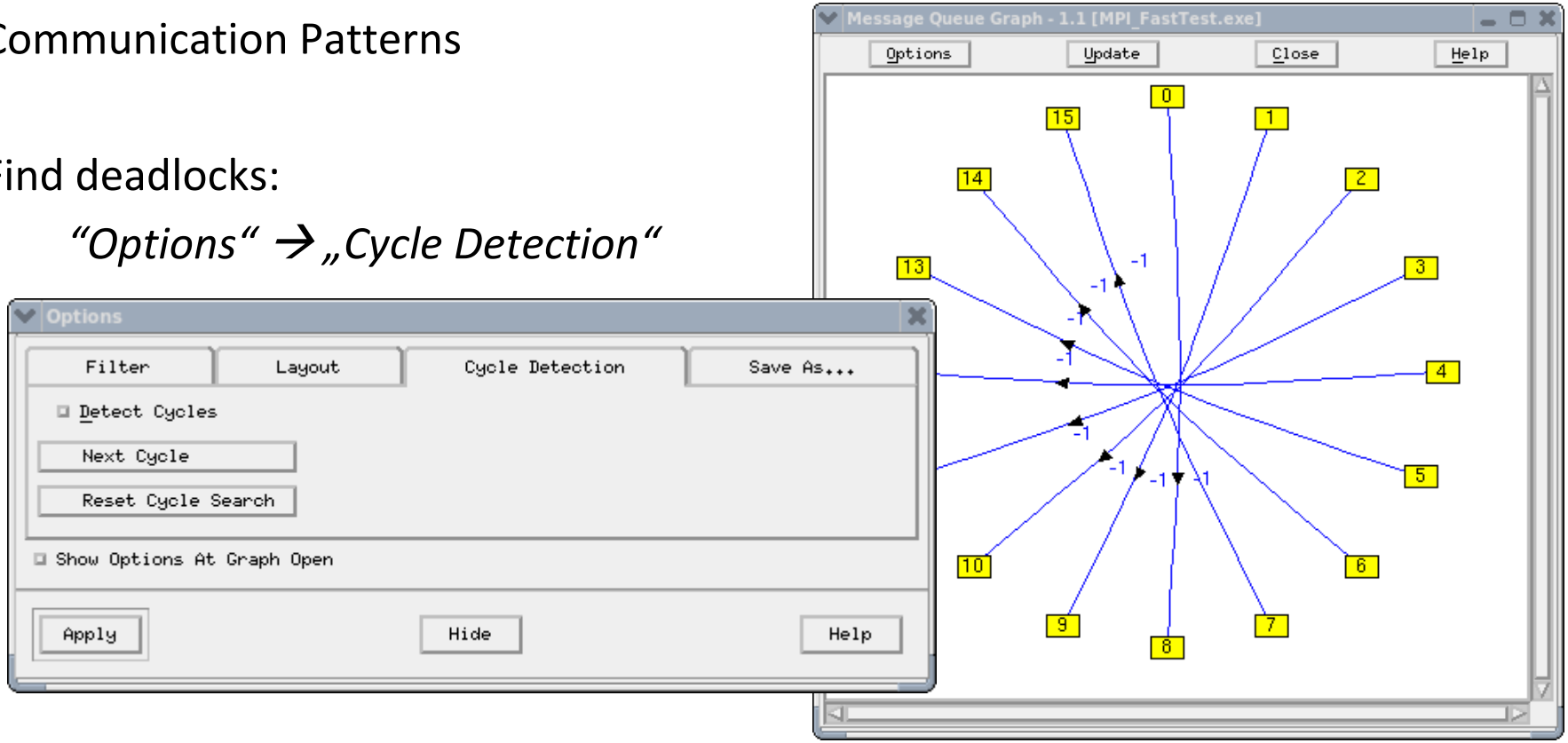
Green: Pending Sends

Blue: Pending Receives

Red: Unexpected Messages

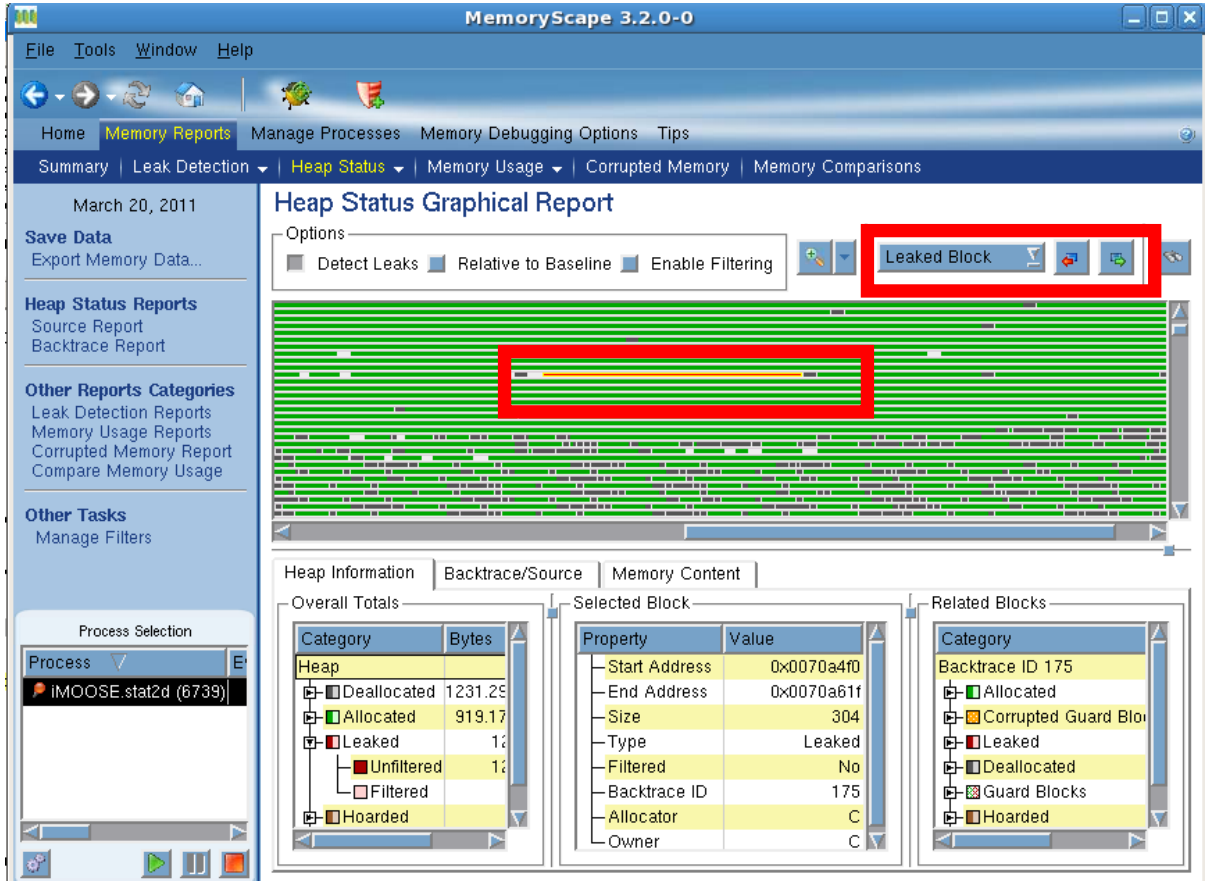
Find deadlocks:

"Options" → „Cycle Detection"



Memory Debugging: MemoryScape

- **Debugging for**
  - → Memory leaks (Leak detection)
  - → Double free
  - → Invalid arrary bounds
- **Memory reports**
  - → Consumption
  - → Comparisons
  - → …



Summary

- **Using a debugger enhances productivity**

- **TotalView helps you!**
  - → Serial programs
  - → Threaded (OpenMP) programs
  - → MPI programs
  - → Hybrid programs
  - → GPGPUs
- **Online information**

  http://www.roguewave.com/support/product-documentation/totalview.aspx#totalview

  http://www.rz.rwth-aachen.de/hpc/primer

The End

Thank you for your attention!

Questions?

We have prepared several exercises.