

Debugging with TotalView



Tim Cramer

22.03.2017

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

- **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!**

■ 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

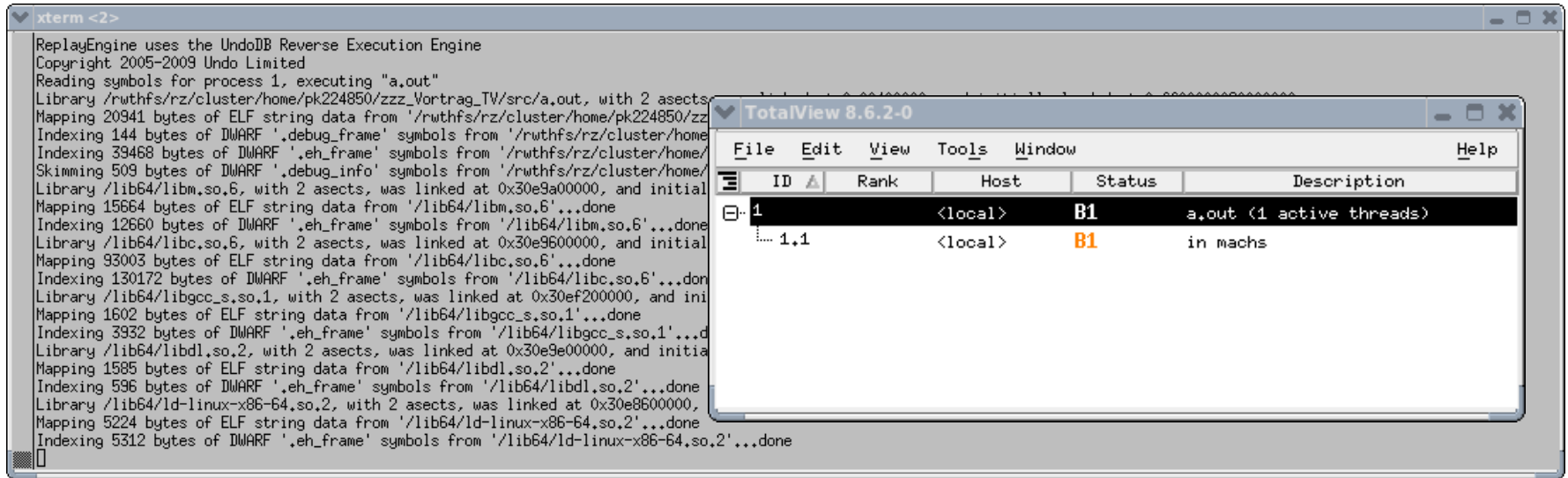
→ Load a core file (Post-Mortem)

The screenshot shows the TotalView Process Window for a.out. The window title is 'a.out'. The menu bar includes File, Edit, View, Group, Process, Thread, Action Point, Debug, Tools, Window, and Help. The toolbar contains icons for Go, Halt, Kill, Restart, Next Step, Out, Run To, Prev UnStep, Caller, and BackTo Live. A status bar at the top shows 'Process 1 (24077): a.out (At Breakpoint 1)' and 'Thread 1 (46912497811904) (At Breakpoint 1)'. The main area is divided into several panes:

- Stack Trace Pane:** Shows a list of stack frames with addresses and frame pointers (FP). The current frame is 'machs' at FP=7fffffffbb20.
- Stack Frame Pane:** Shows details for the current frame 'machs', including local variables: widerhol (10), maxiter (20000000), and registers for the frame.
- Source Pane:** Shows the source code for the function 'machs' in 'bsp_PPCEs_1.f90'. Line 19 is highlighted: 'ALLOCATE(var(maxiter))'. Line 20 is also highlighted: 'var = 3,14'.
- Tabbed Pane:** Shows a list of action points. The first action point is selected: '1 bsp_PPCEs_1.f90#20 machs+0x178'.

Callouts in the image identify the following components:

- ← **Toolbar**
- ← **Process and Thread Status**
- Stack Trace Pane**
- Stack Frame Pane**
- Source Pane**
- Tabbed Pane**



Status Info

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

■ 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

■ 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

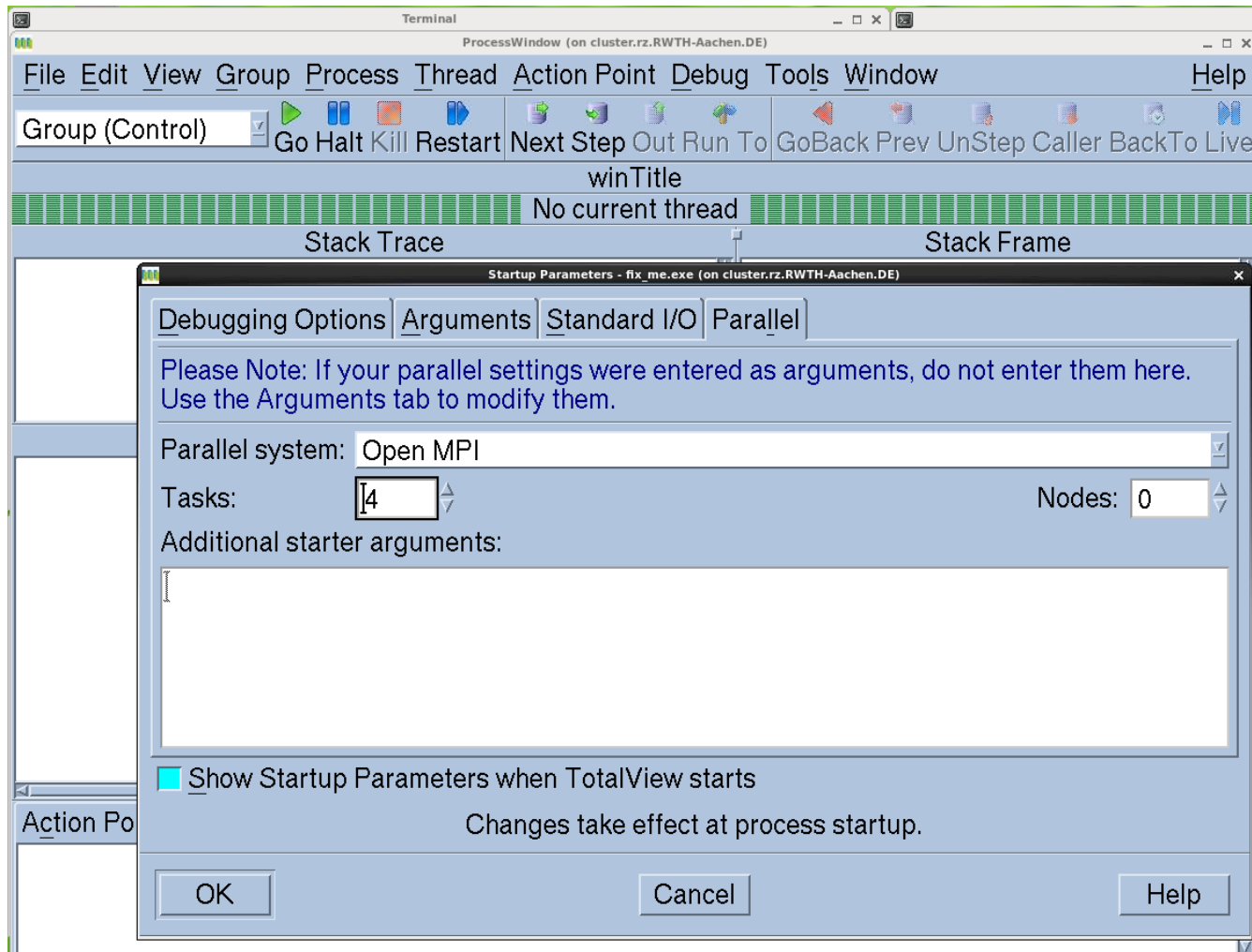
→ Classic launch: `$ mpiexec -tv -np 4 mpi-a.out`

→ Arguments depend on MPI vendor

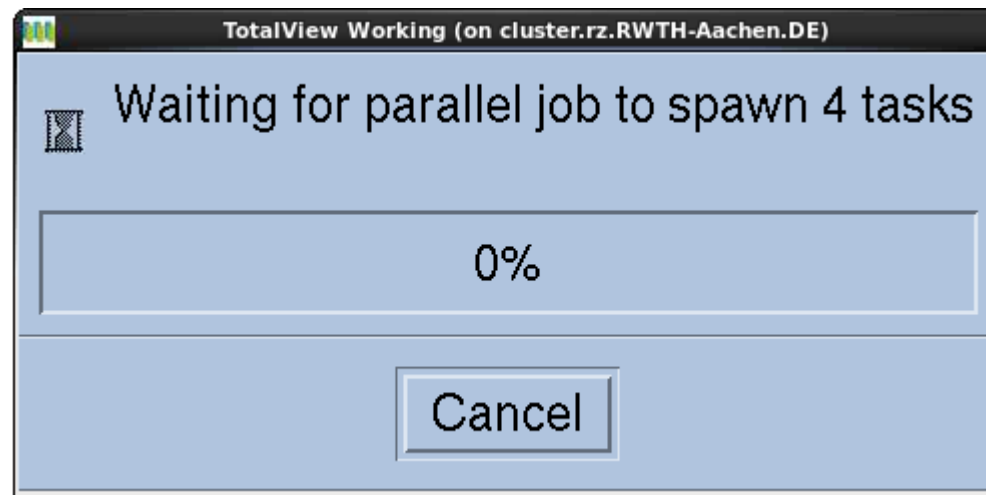
→ Attach / Attach to subset / Detache / Reattache possible



Live Demo Parallel Debugging



■ Be patient



Rank 0: fix_me.exe.0 (Running)
Thread 1 (140037624768256): fix_me.exe (Running)

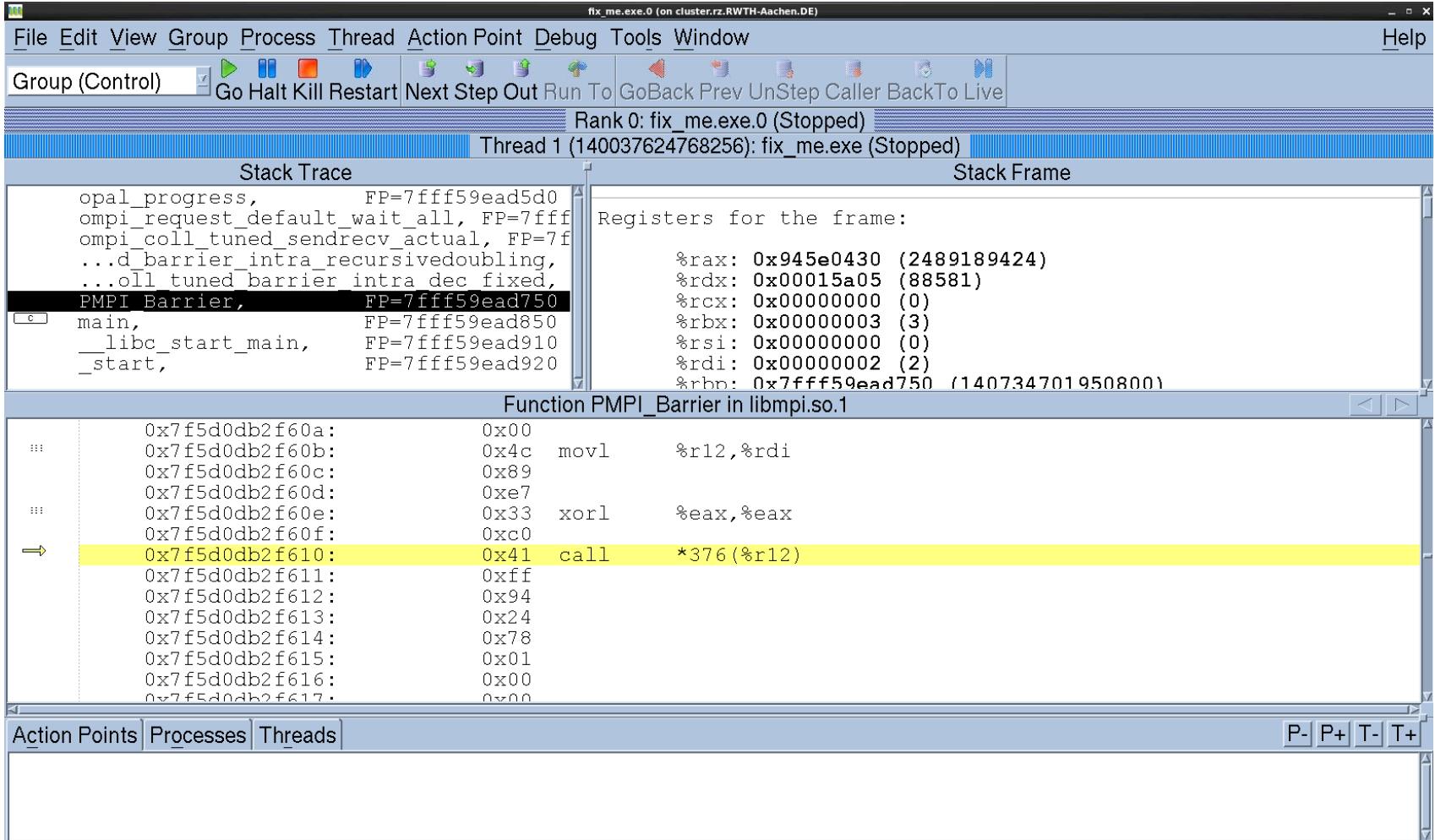
Stack Trace Stack Frame

Thread must be stopped for frame display.

TotalView 8.9.2-2 (on cluster.rz.RWTH-Aachen.DE)

	ID	Rank	Host	Status	Description
1	0	134.61.220.74	R	fix_me.exe.0 (3 active threads)	
3	3	134.61.220.74	R	fix_me.exe.3 (3 active threads)	
4	1	134.61.220.74	R	fix_me.exe.1 (3 active threads)	
5	2	134.61.220.74	R	fix_me.exe.2 (3 active threads)	

■ Press the “HALT” button



The screenshot shows the TotalView debugger interface for a process named 'fix_me.exe.0' on a cluster. The process is stopped at Rank 0. The stack trace shows the following frames:

```
Stack Trace
opal_progress, FP=7fff59ead5d0
mpi_request_default_wait_all, FP=7fff59ead5d0
mpi_coll_tuned_sendrecv_actual, FP=7fff59ead5d0
...d_barrier_intra_recurse_doubling,
...oll_tuned_barrier_intra_dec_fixed,
PMPI_Barrier, FP=7fff59ead750
main, FP=7fff59ead850
__libc_start_main, FP=7fff59ead910
_start, FP=7fff59ead920
```

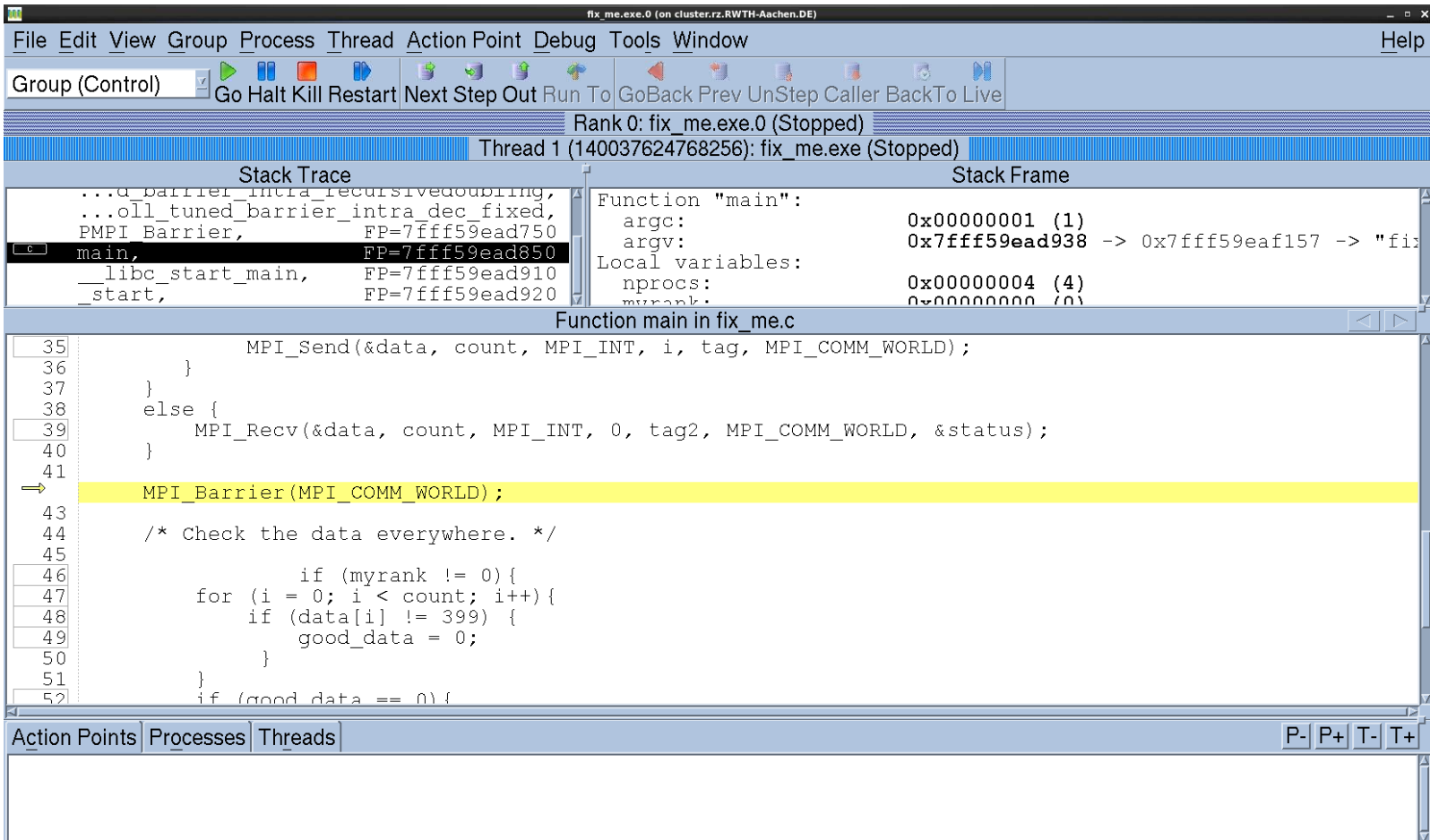
The registers for the frame are:

```
Registers for the frame:
%rax: 0x945e0430 (2489189424)
%rdx: 0x00015a05 (88581)
%rcx: 0x00000000 (0)
%rbx: 0x00000003 (3)
%rsi: 0x00000000 (0)
%rdi: 0x00000002 (2)
%rbp: 0x7fff59ead750 (140734701950800)
```

The assembly view shows the function 'Function PMPI_Barrier in libmpi.so.1' with the following instructions:

```
0x7f5d0db2f60a: 0x00
0x7f5d0db2f60b: 0x4c movl %r12,%rdi
0x7f5d0db2f60c: 0x89
0x7f5d0db2f60d: 0xe7
0x7f5d0db2f60e: 0x33 xorl %eax,%eax
0x7f5d0db2f60f: 0xc0
→ 0x7f5d0db2f610: 0x41 call *376(%r12)
0x7f5d0db2f611: 0xff
0x7f5d0db2f612: 0x94
0x7f5d0db2f613: 0x24
0x7f5d0db2f614: 0x78
0x7f5d0db2f615: 0x01
0x7f5d0db2f616: 0x00
0x7f5d0db2f617: 0x00
```

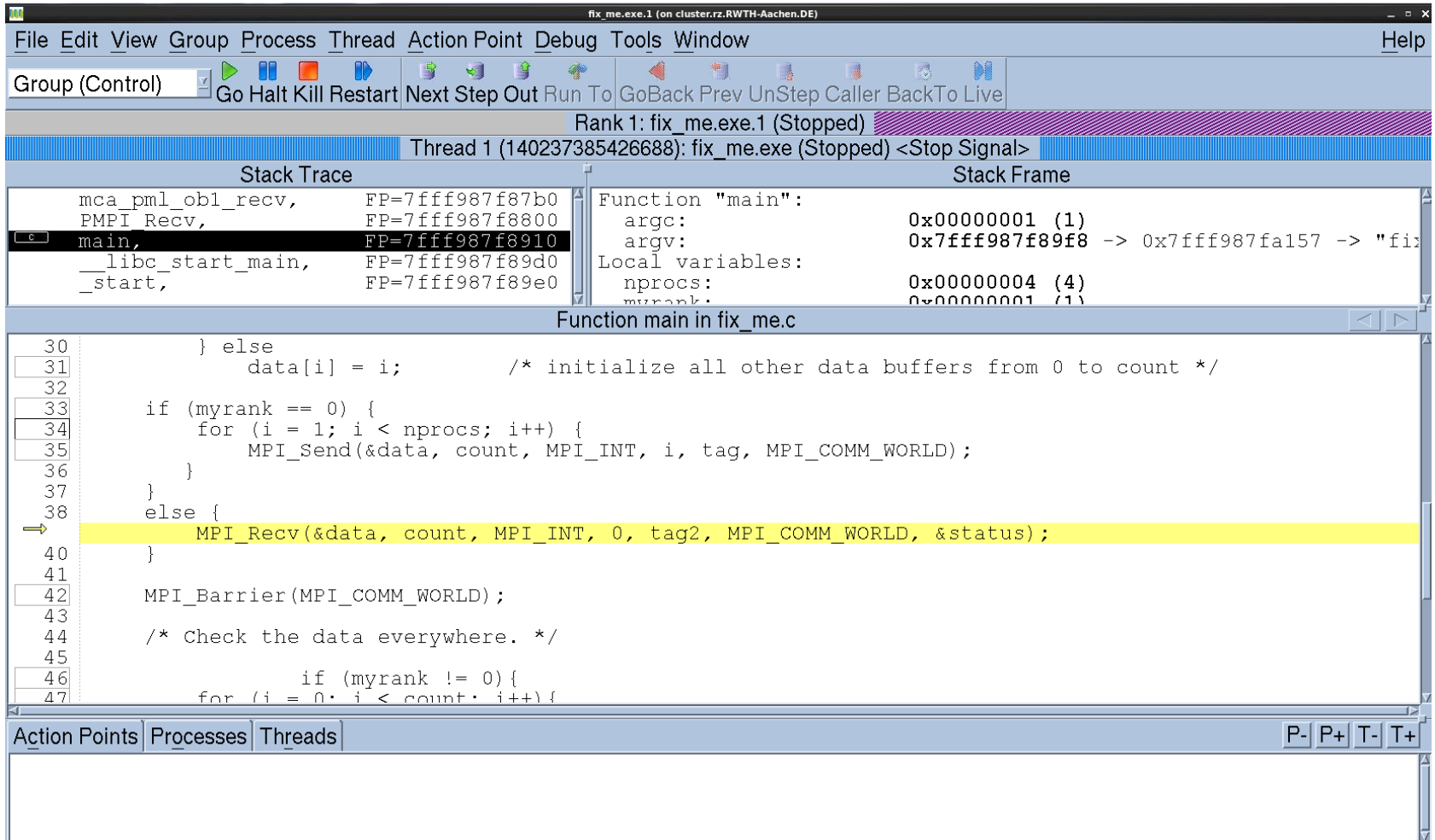
Rank 0 is waiting in barrier



The screenshot shows the TotalView debugger interface for the process `fix_me.exe.0` (Stopped). The main window displays the source code for `Function main in fix_me.c`. The execution has stopped at line 41, which is `MPI_Barrier(MPI_COMM_WORLD);`. The stack trace on the left shows the call stack, with `main` at the top. The stack frame on the right shows the function arguments and local variables, including `nprocs: 0x00000004 (4)` and `myrank: 0x00000000 (0)`. The code below the barrier includes a loop that checks data values for each rank.

```
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);
42
43 /* Check the data everywhere. */
44
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){
```

Rank 1 still waits for data



The screenshot shows the TotalView debugger interface for a process named 'fix_me.exe.1' on cluster.rz.RWTH-Aachen.DE. The process is stopped at Rank 1. The stack trace shows the following frames:

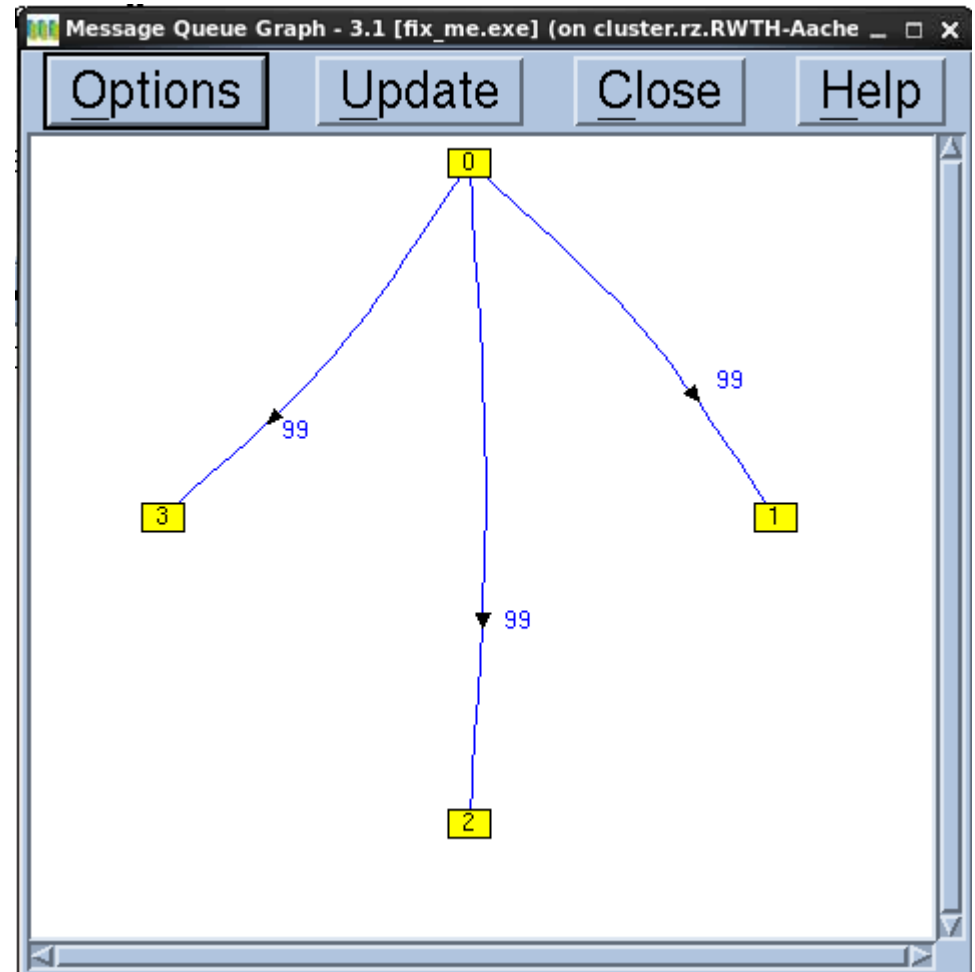
Stack Trace	Stack Frame
mca_pml_ob1_recv, FP=7fff987f87b0	Function "main":
PMPI_Recv, FP=7fff987f8800	argc: 0x00000001 (1)
main, FP=7fff987f8910	argv: 0x7fff987f89f8 -> 0x7fff987fa157 -> "fix"
__libc_start_main, FP=7fff987f89d0	Local variables:
_start, FP=7fff987f89e0	nprocs: 0x00000004 (4)
	myrank: 0x00000001 (1)

The code editor shows the following code snippet:

```
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 {
39         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++)!
```

■ Message Queue Graph

→ Rank 1, 2, 3 are waiting
for data on tag 99



■ 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"



“Tools” → “Message Queue Graph”

Hangs & Deadlocks

Pending Messages

Communication Patterns

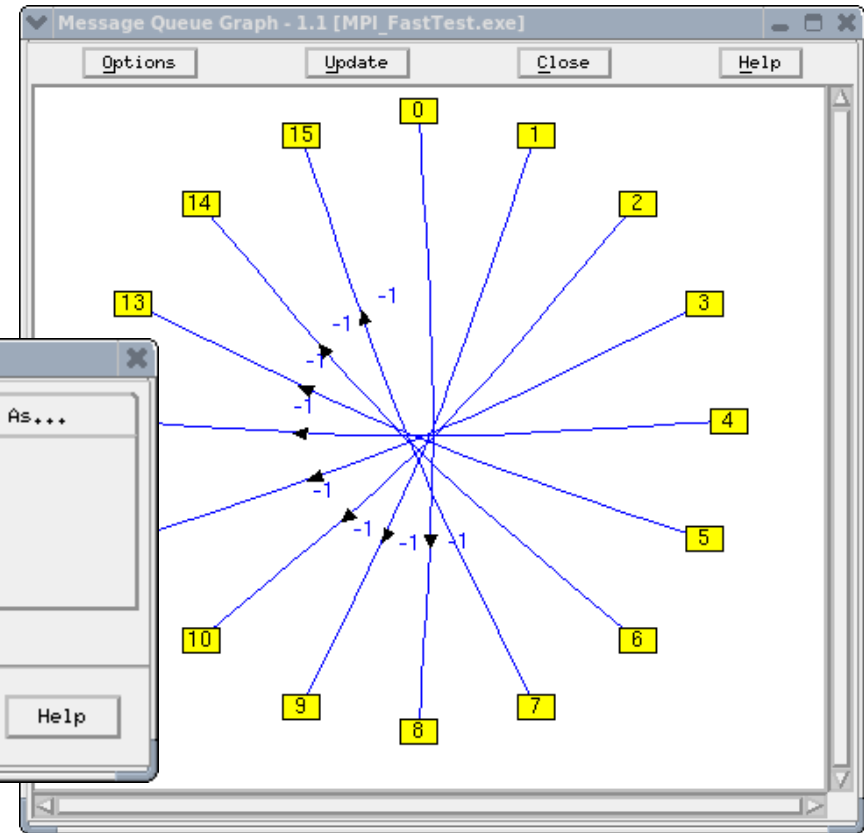
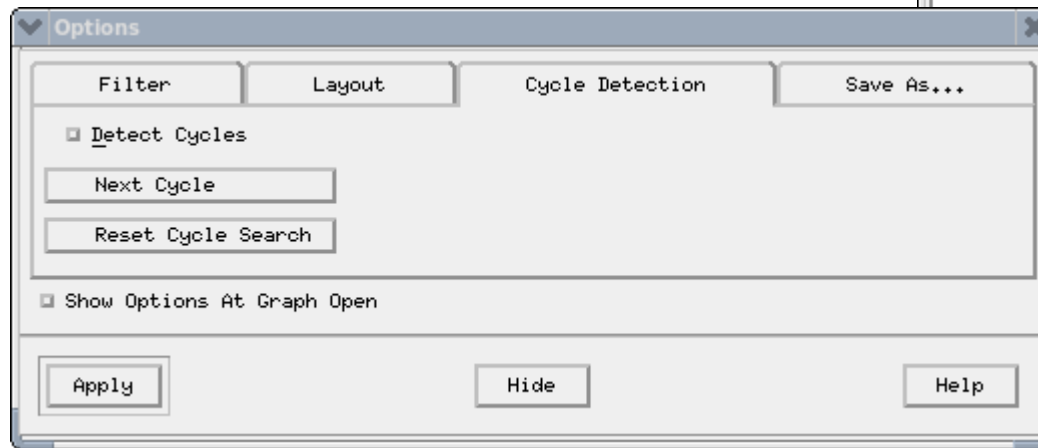
Find deadlocks:

“Options” → „Cycle Detection“

Green: Pending Sends

Blue: Pending Receives

Red: Unexpected Messages



Thank you for your attention!

