

Debugging with TotalView



Tim Cramer
17.03.2015

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

■ 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 X-Win32
- All examples should be distributed in the hpclabXX accounts
(directory: `${HOME}/totalviewlabs/livedemo`)
- Slides for interactive sessions will be online with screenshots

- **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):

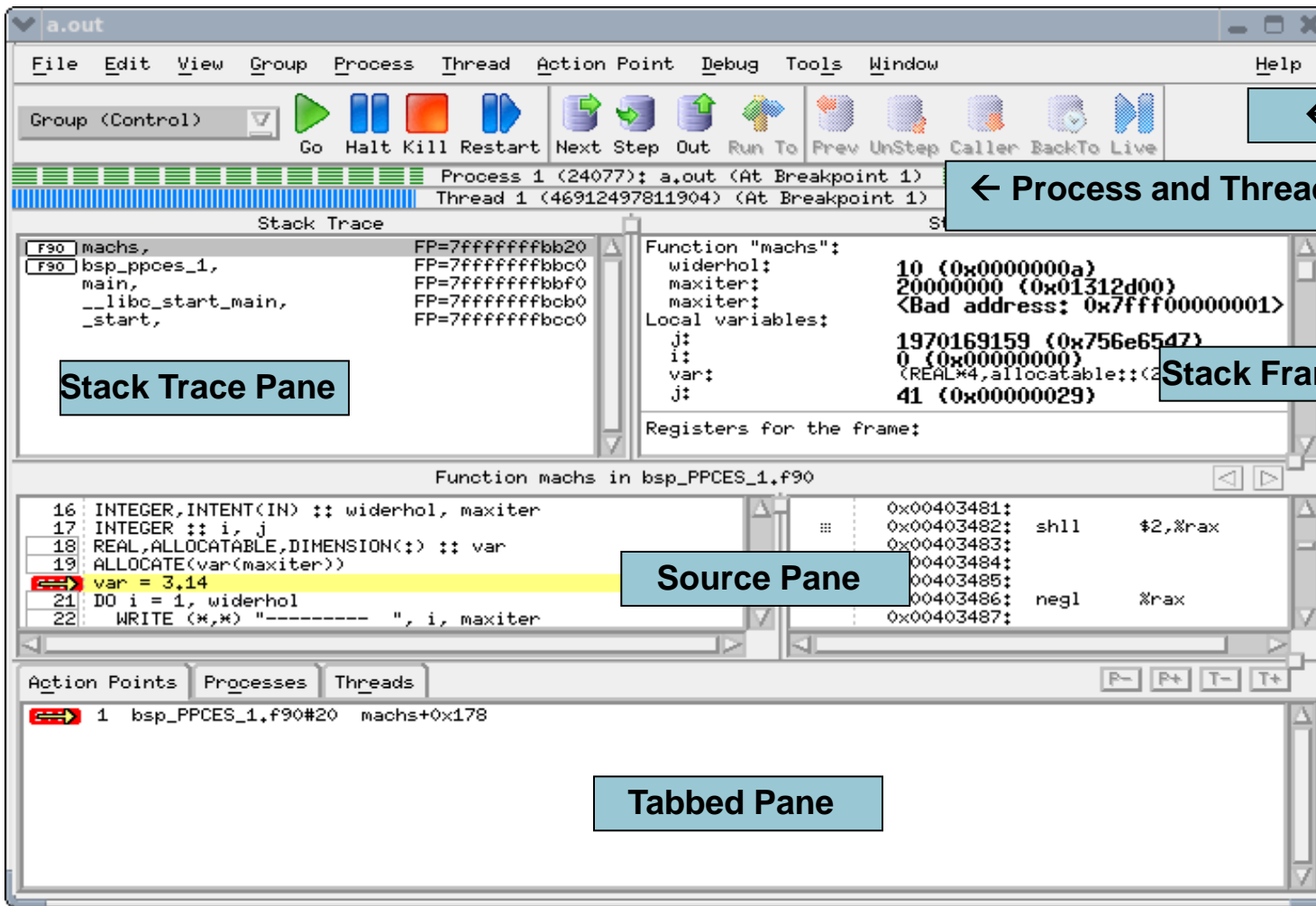
→ `$ totalview a.out -a <options of a.out>`

■ Main modes:

→ Start a new process

→ Attach to a running process

→ Load a core file (Post-Mortem)



← Toolbar

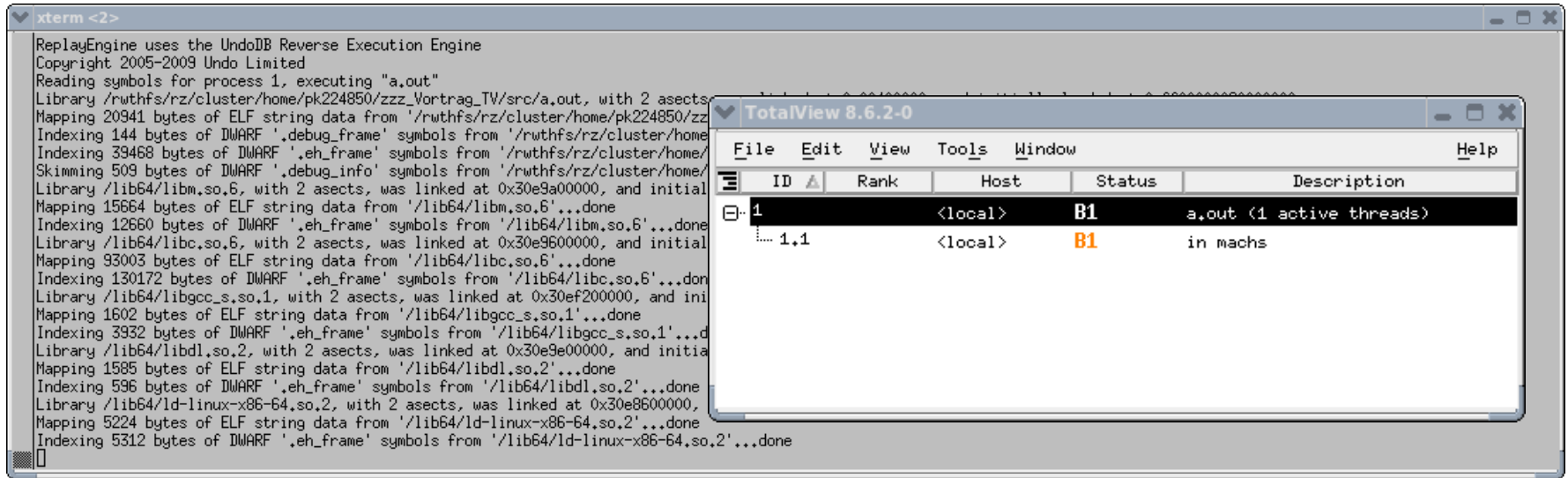
← Process and Thread Status

Stack Trace Pane

Stack Frame Pane

Source Pane

Tabbed Pane

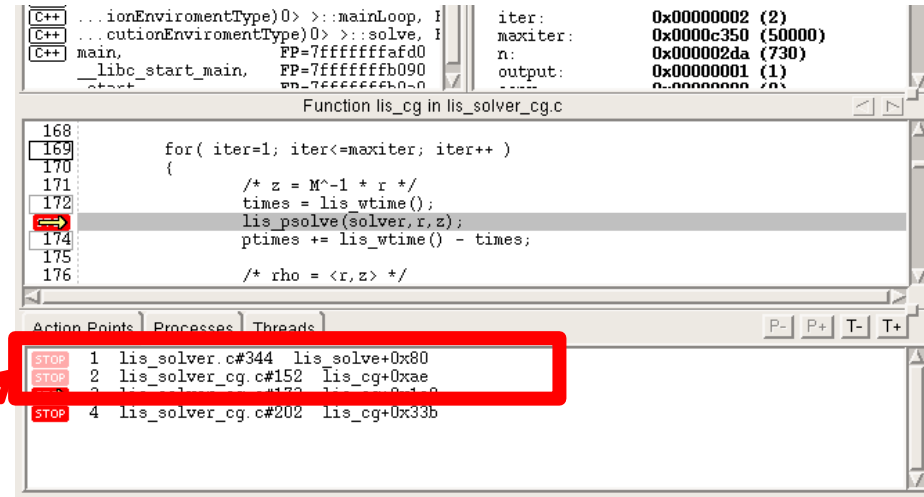


Status Info

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

■ Breakpoints

- Interrupt execution when reaching a specific code line
- Conditional Breakpoints possible
- Set by clicking in the source pane
- Temporary disabling is possible



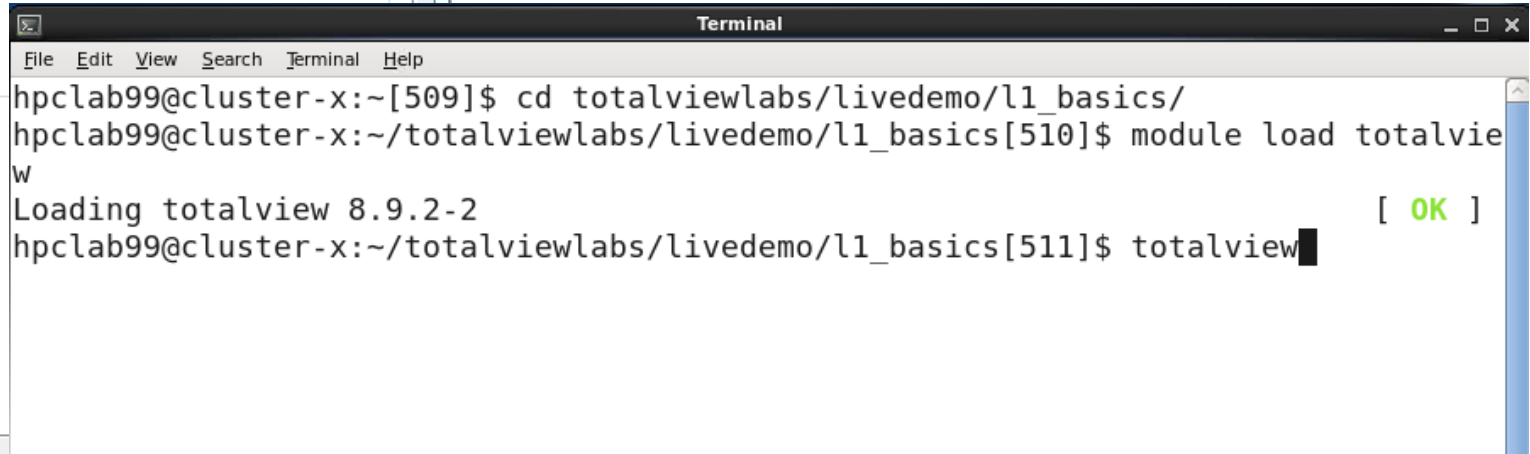
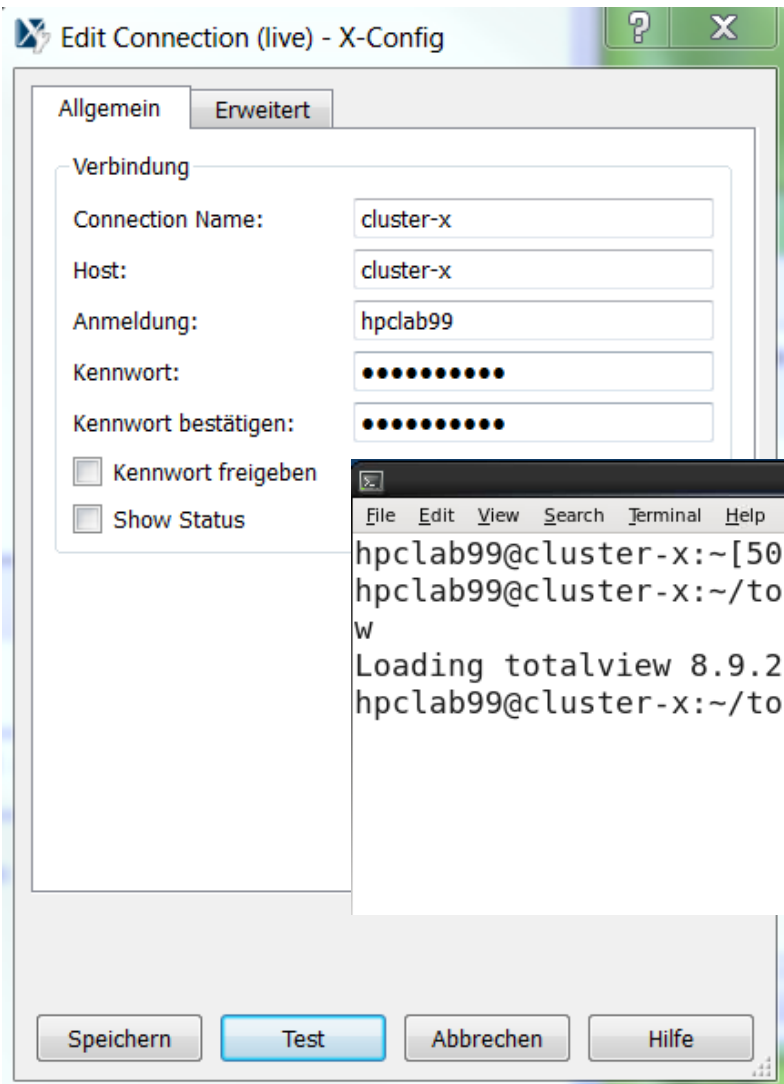
■ Watchpoints

- 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)



Live Demo Startup

L1: Login with X-Win32 & Startup TotalView



```
hpclab99@cluster:~/totalviewlabs/livedemo/l1_basics[32]$ ulimit -c unlimited
hpclab99@cluster:~/totalviewlabs/livedemo/l1_basics[32]$ make
icc -O0 -g fix_me.c -o fix_me.exe
hpclab99@cluster:~/totalviewlabs/livedemo/l1_basics[33]$ ./fix_me.exe
Init a[0] with 0.000000
Init a[1] with 1.000000

[...]

Init a[29179] with 29179.000000
Init a[29180] with 29180.000000
Init a[29181] with 29181.000000
zsh: segmentation fault (core dumped) ./fix_me.exe
hpclab99@cluster:~/totalviewlabs/livedemo/l1_basics[34]
```

```
hpclab99@cluster:~/totalviewlabs/livedemo/l1_basics[48]$ cat fix_me.c
#include <stdio.h>
#include <stdlib.h>

void init(double* a, const int count){
    int i;

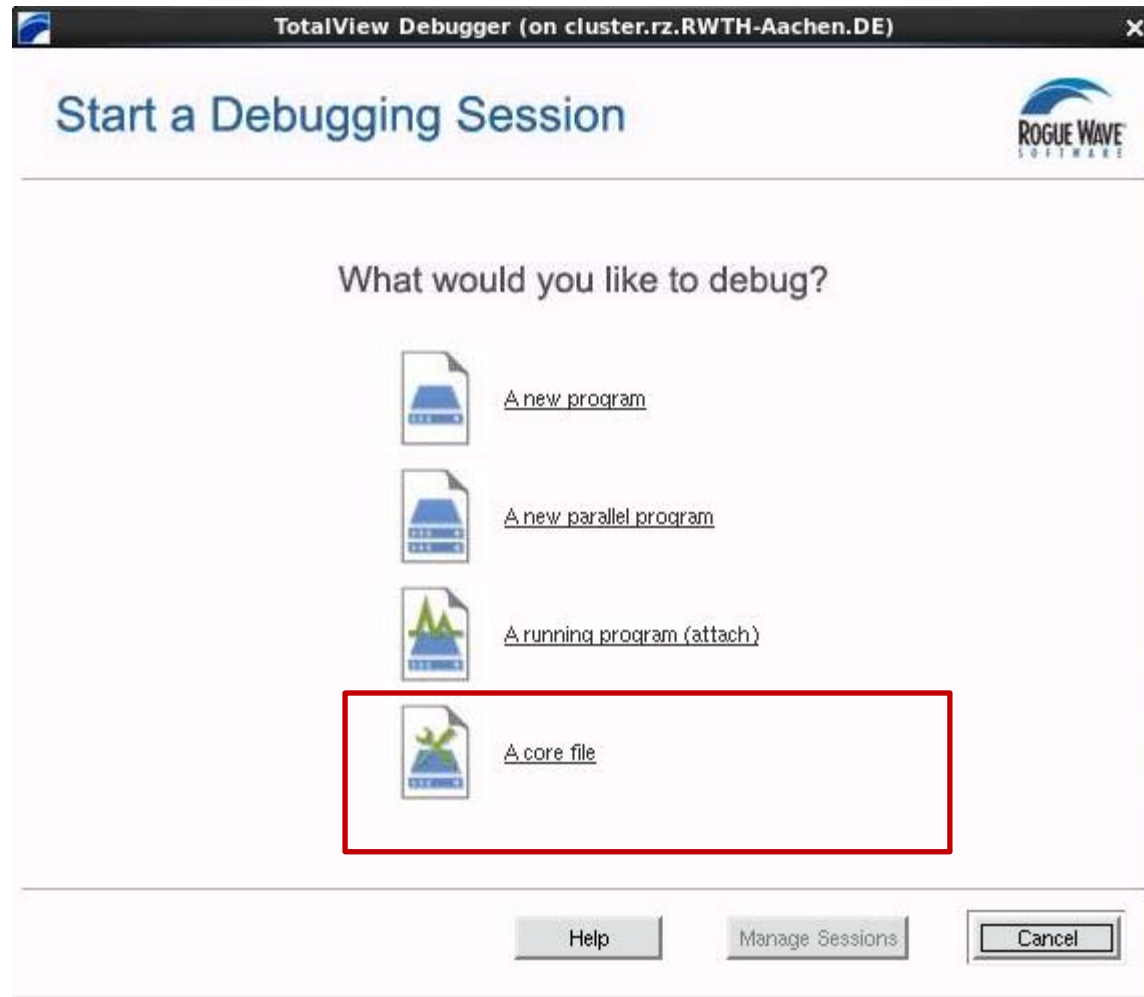
    for (i=0; i<count; i++){
        a[i] = (double)i;
        printf("Init a[%d] with %f\n", i, a[i]);
    }
}

int main(int argc, char **argv){
    const int count = 100000;
    double* a;

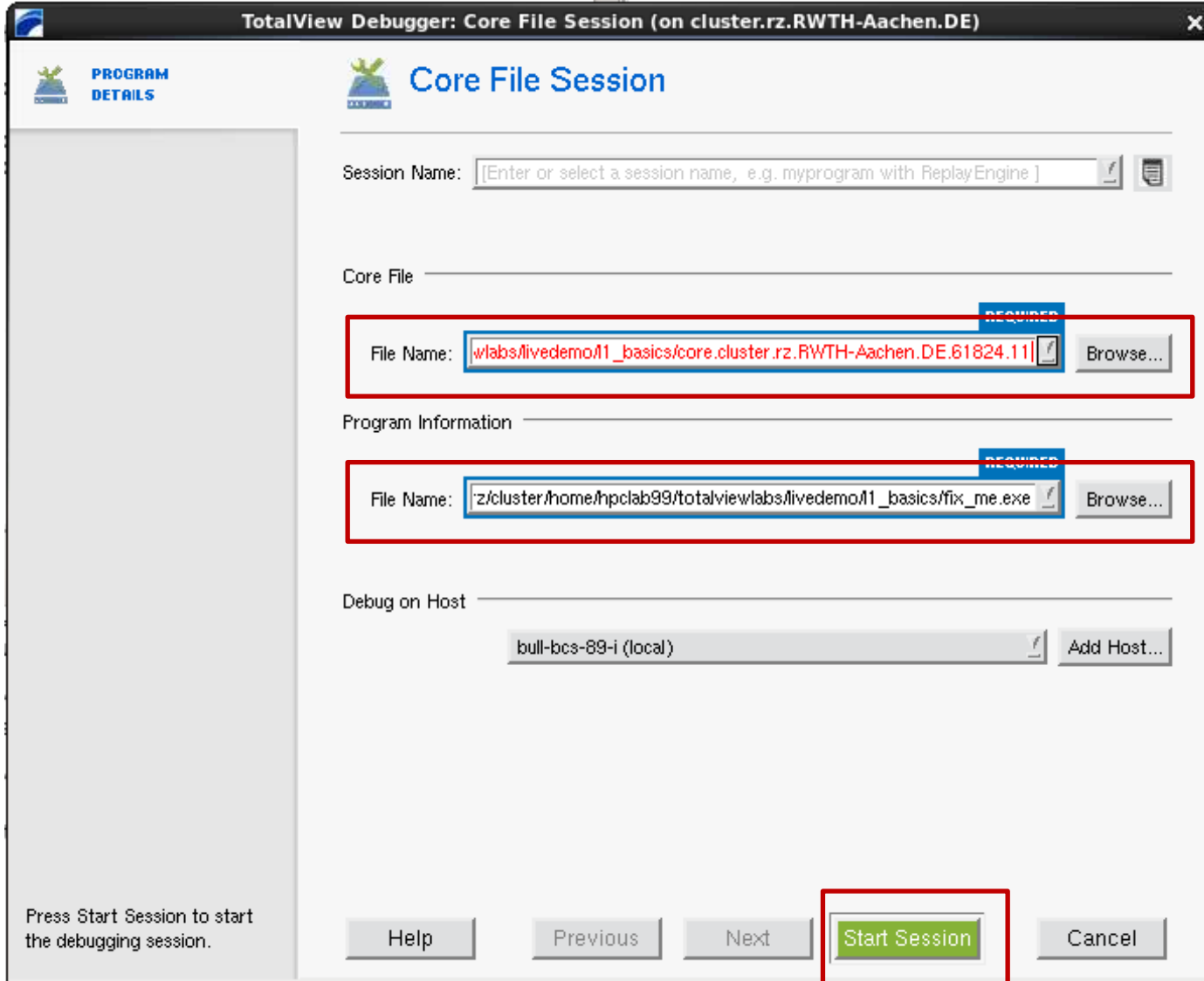
    a = (double*)malloc(count);

    init(a, count);

    return 0;
}
```



L1: Post-Mortem Debugging (1/2)



<= core file

<= executable

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

FP	Address
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)

%ebx: 0x00000000 (0)

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);
20 }
```

TotalView 8.9.2-2

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

L1: Running within TotalView



The screenshot displays the TotalView Debugger interface. The main window is titled "TotalView Debugger: Program Session (on cluster.rz.RWTH-Aachen.DE)". On the left, there is a sidebar with "PROGRAM DETAILS", "DEBUG OPTIONS", and "ENVIRONMENT". The "Program Session" window has several sections: "Session Name" with a text input field, "Program Information" with a "File Name" field (marked "REQUIRED") and an "Arguments" field, and "Debug on Host" with a dropdown menu showing "bull-bcs-89-i (local)". A "Choose a program to load" dialog box is open in the foreground, showing a file explorer view of the directory "C:\home\hpclab99\totalviewlabs\livedemo\l1_basics\". The file list includes "Makefile", "core.cluster.rz.RWTH-Aachen.DE.1262.11", "core.cluster.rz.RWTH-Aachen.DE.61824.11", "fix_me.c", "fix_me.cpp", and "fix_me.exe", which is currently selected. The dialog also shows "File name: fix_me.exe" and "File type: Program Files (*.*)".

Setting a breakpoint



The screenshot displays the TotalView 8.9.2-2 interface. The main window shows the source code for `fix_me.c` with a breakpoint set on line 19, `init(a, count);`. The stack trace and registers are visible, and the Action Points window shows the breakpoint details.

Stack Trace:

Function	FP
main	7fff10d8a420
__libc_start_main	7fff10d8a4e0
_start	7fff10d8a4f0

Stack Frame:

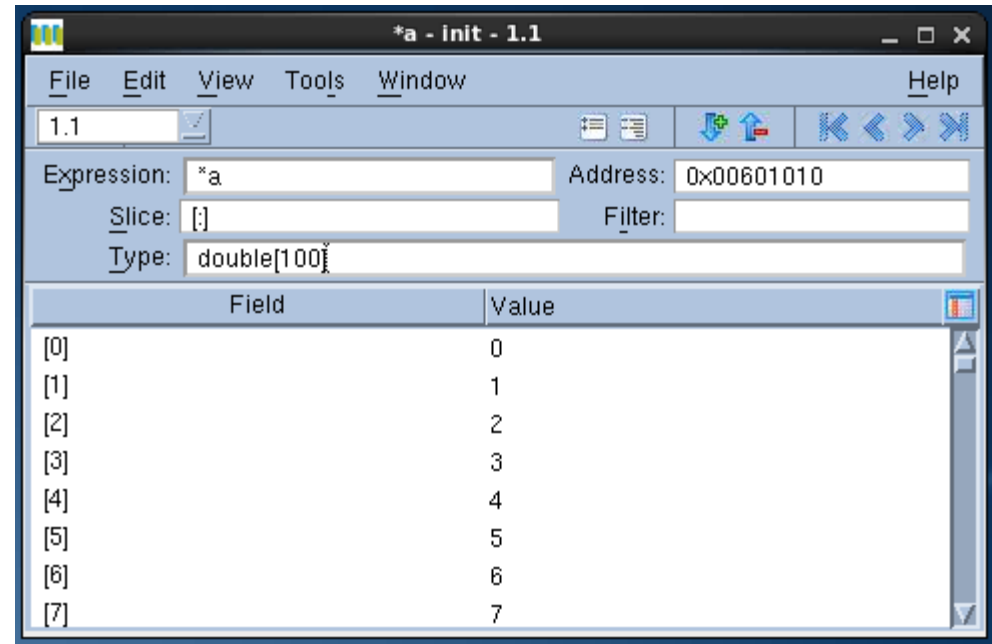
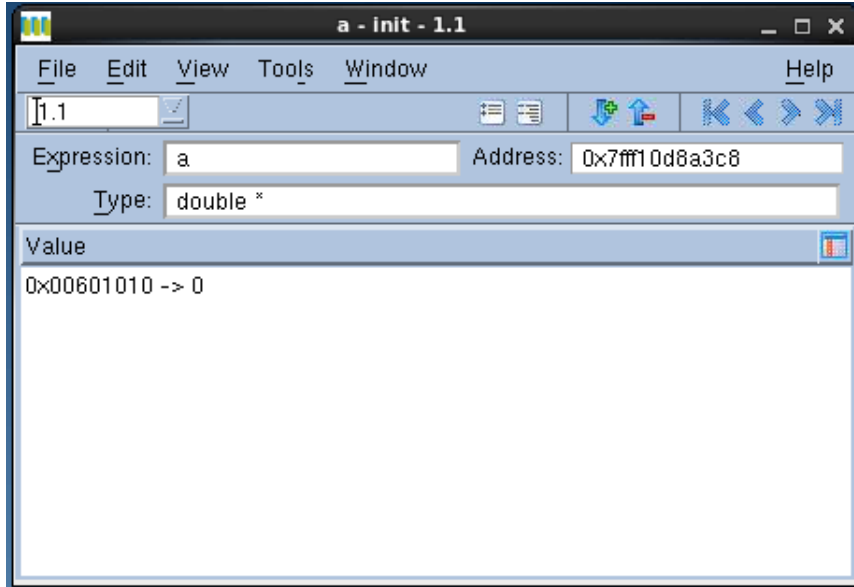
Function "main":
argc: 0x00000001 (1)
argv: 0x7fff10d8a508 -> 0x7fff10d8bc50 -> "fix_me.exe"
Local variables:
count: 0x000186a0 (100000)
a: 0x00601010 -> 0

Registers for the frame:

%rax: 0x00601010 (6295568)
%rdx: 0x000186b1 (100017)
%rcx: 0x00000000 (0)

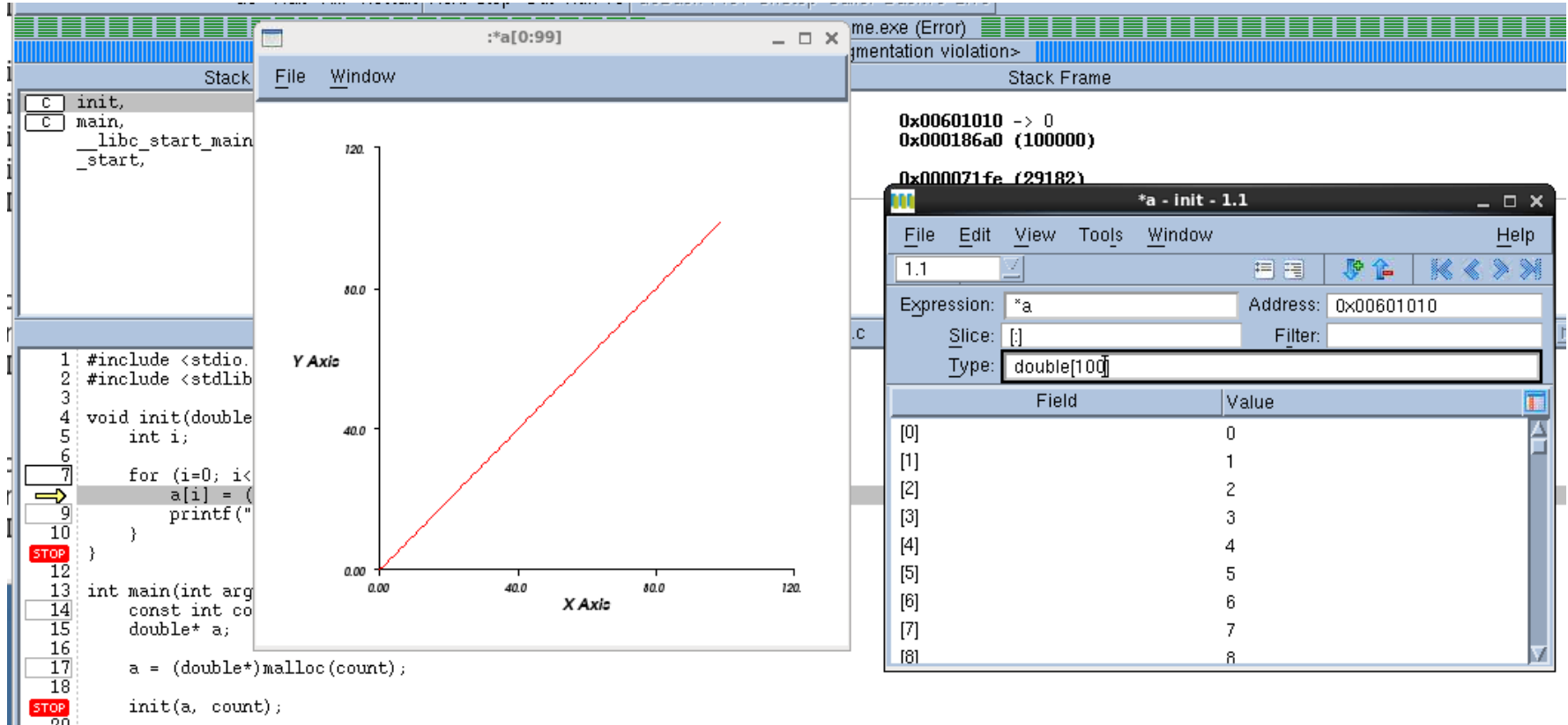
Action Points:

ID	Rank	Host	Status	Description
1	<local>	<local>	B2	/home/hpclub99/totalviewlabs/livec



■ **Typecast necessary**

Helpful for big data arrays



The screenshot displays the TotalView debugger interface. On the left, the source code for a program is shown, with a yellow arrow pointing to line 7: `a[i] = i;`. The stack trace on the right shows the current frame at `me.exe (Error)` and the caller `0x000071fe (29182)`. A graph window titled `:*a[0:99]` shows a linear plot of the array data, with the X-axis labeled `X Axis` and the Y-axis labeled `Y Axis`. The graph shows a red line representing the array values, which are integers from 0 to 99. A memory dump window titled `*a - init - 1.1` shows the expression `*a` at address `0x00601010`, with a slice of `[]` and a filter of `double[100]`. The dump shows the values of the array elements from index 0 to 8.

Field	Value
[0]	0
[1]	1
[2]	2
[3]	3
[4]	4
[5]	5
[6]	6
[7]	7
[8]	8



Live Demo Watchpoints

■ Create a watchpoint for a[29]

The screenshot shows the TotalView debugger interface. The main window displays a C program with a breakpoint at line 19: `init(a, count);`. A watchpoint dialog is open, showing the expression `*(a)` at address `0x00601010`. The dialog lists memory locations and their values:

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

The 'Create Watchpoint' option is highlighted in the dialog. The bottom status bar shows the current location: `2 fix_me.c#19 main+0x30`.

Will interrupt as soon as a [29] changes

The screenshot shows the TotalView debugger interface. The main window displays the source code of `fix_me.c` with a watchpoint set on line 8, `a[i] = (double)i;`. The watchpoint is configured to trigger on changes to the 8th byte of the array element at index 29. The `Stack Trace` shows the current execution path: `init`, `main`, `__libc_start_main`, and `_start`. The `Stack Frame` window shows the `init` function with the expression `*(a)` and type `double[1000]`. The `Value` column shows the values of array elements from index 27 to 35, with index 29 highlighted in yellow. The `Action Points` window shows the watchpoint configuration: `5 8 bytes @ 0x006010f8 ((a)[29])`.

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

■ Parallel Debugging might be very hard

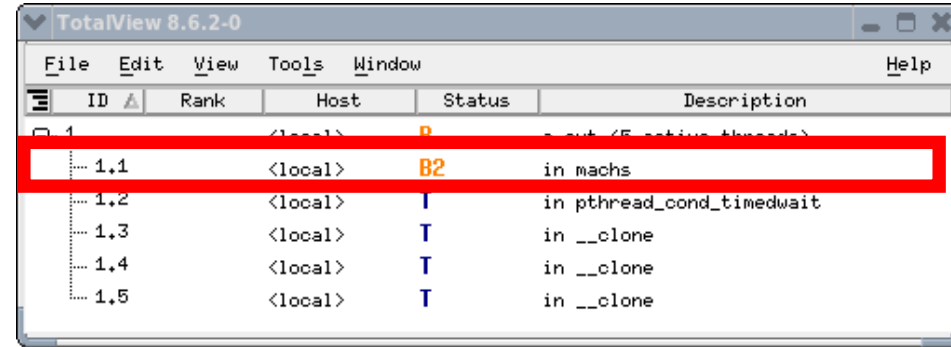
- Try to debug a *serial* version of the program first!
- Typical multithreading errors may not be found (e.g., *race conditions*)
- Some errors only occur
 - with optimized code (uninitialized variables?)
 - with many processes
 - outside of debug sessions (different timing)

■ 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

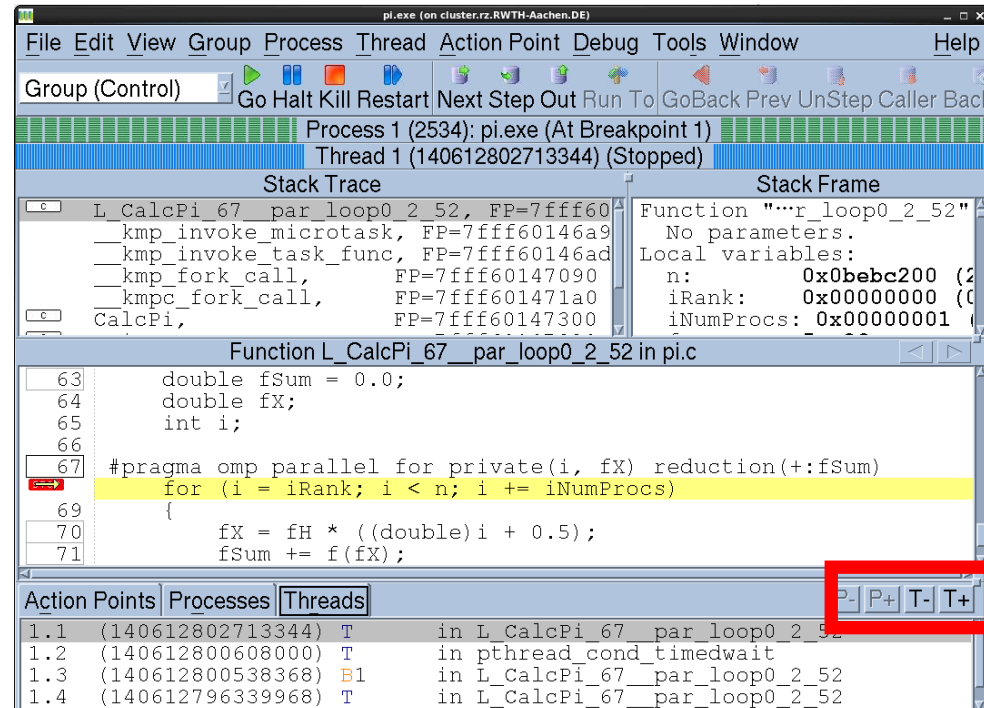
■ Overview of thread number

- 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+

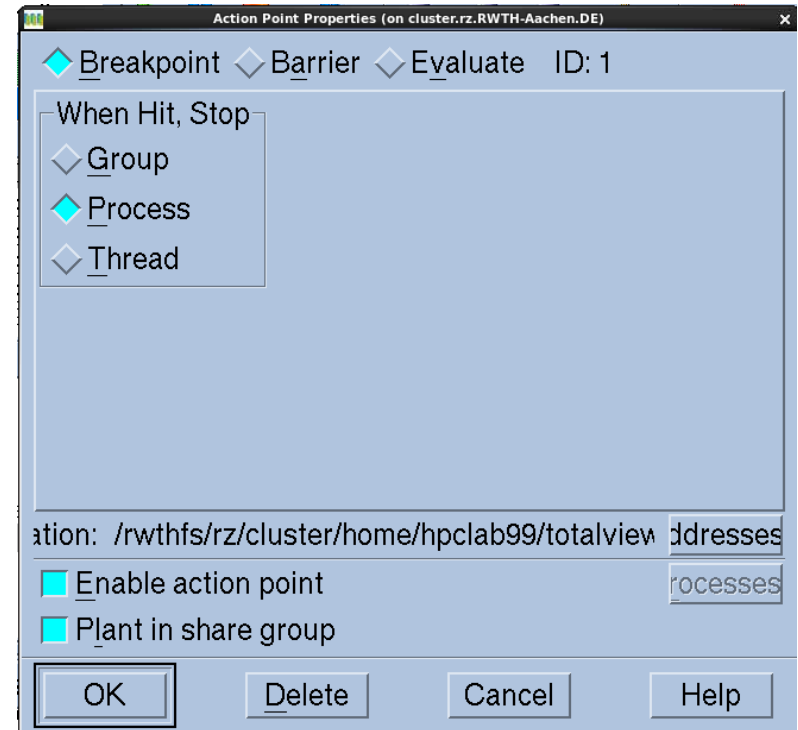


Breakpoint properties and barriers

→ Default behavior: Same breakpoint for all threads

→ Change “Properties”

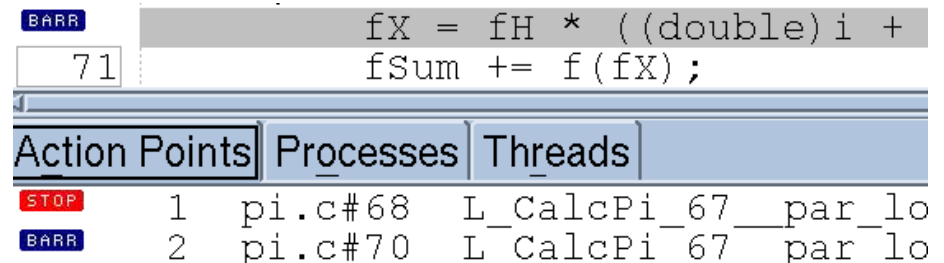
Group	Stop all threads of all processes
Process	Stop all threads of one process (default)
Thread	Stop only the current thread



→ Barriers

→ Similar to breakpoints

→ Help to synchronize



■ 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

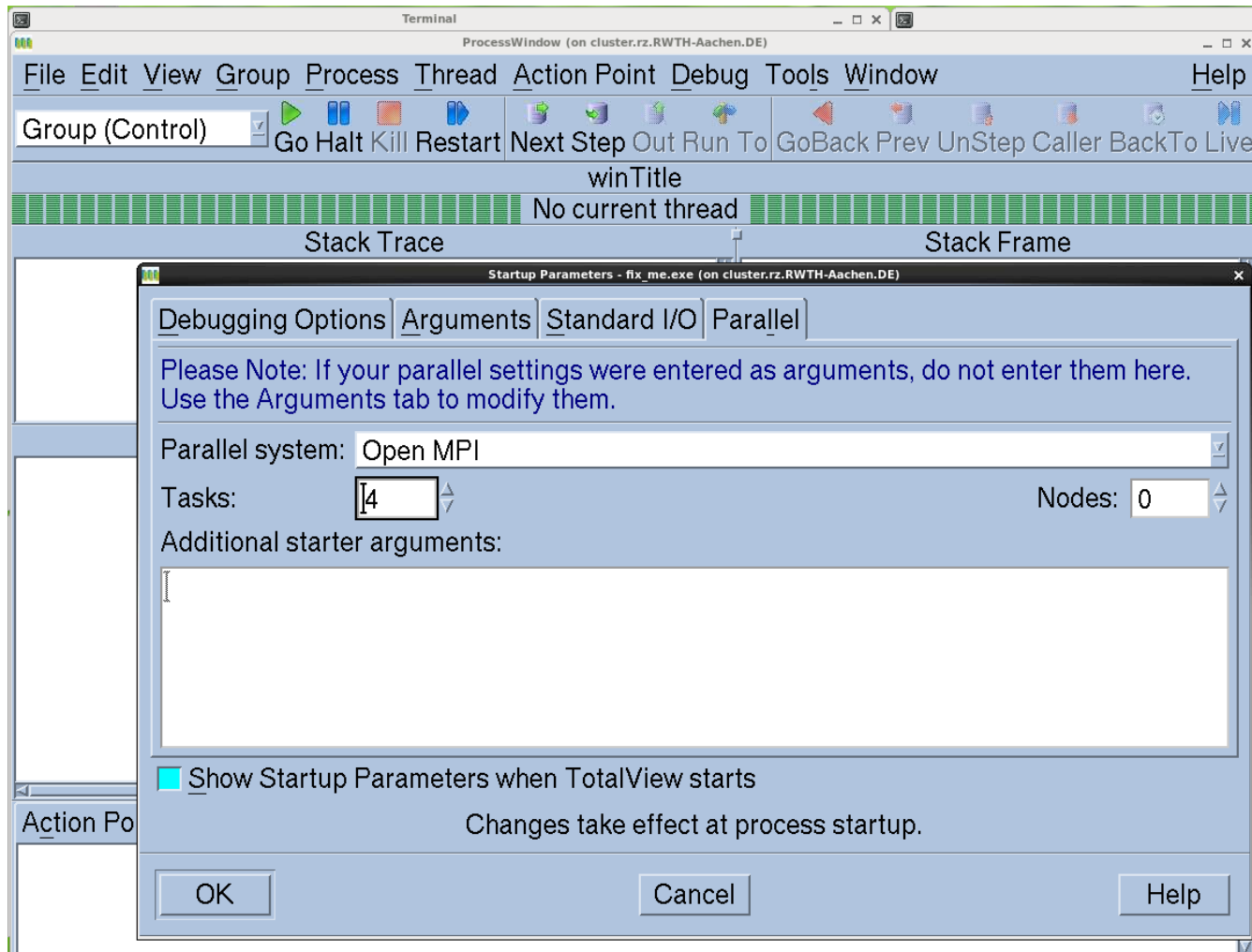
→ Classic launch: `$ mpiexec -tv -np 4 mpi-a.out`

→ Arguments depend on MPI vendor

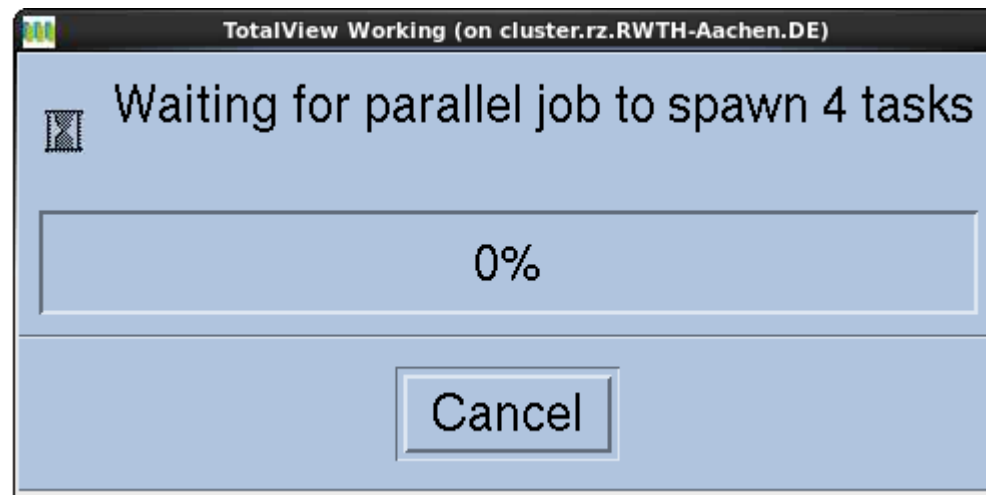
→ Attach / Attach to subset / Detache / Reattache possible



Live Demo Parallel Debugging



- Be patient



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

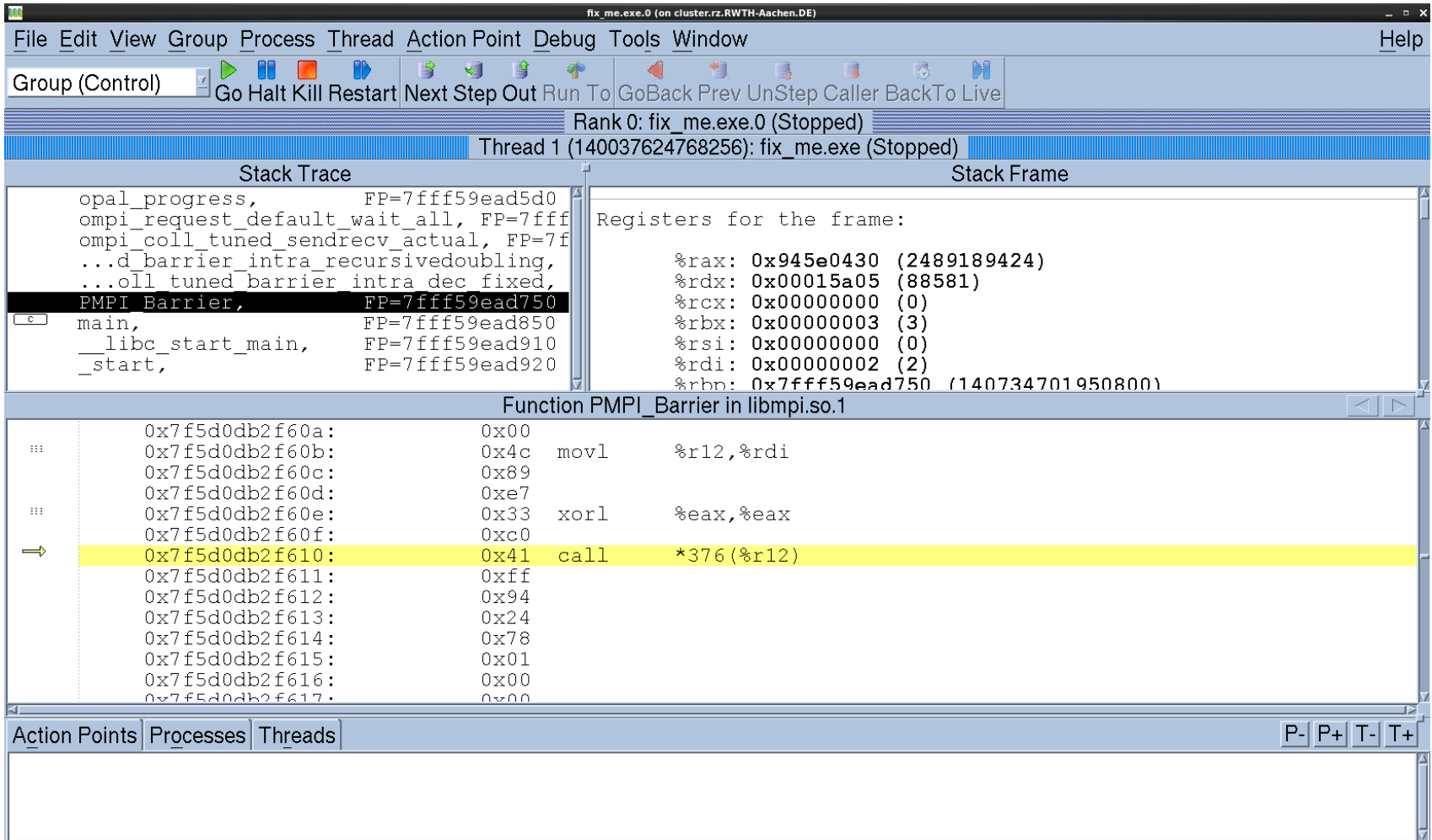
Stack Trace Stack Frame

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 at RWTH Aachen University. The process is in a 'Stopped' state. The 'Stack Trace' pane on the left shows the call stack, with the current frame being 'PMPI_Barrier' in 'libmpi.so.1'. The 'Stack Frame' pane on the right displays the registers for this frame, including %rax, %rdx, %rcx, %rbx, %rsi, %rdi, and %rbp. The 'Function PMPI_Barrier in libmpi.so.1' pane at the bottom shows the assembly code, with the instruction 'call *376(%r12)' highlighted in yellow. The 'Action Points' pane at the bottom is empty.

Rank 0: fix_me.exe.0 (Stopped)

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

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
```

Stack Frame

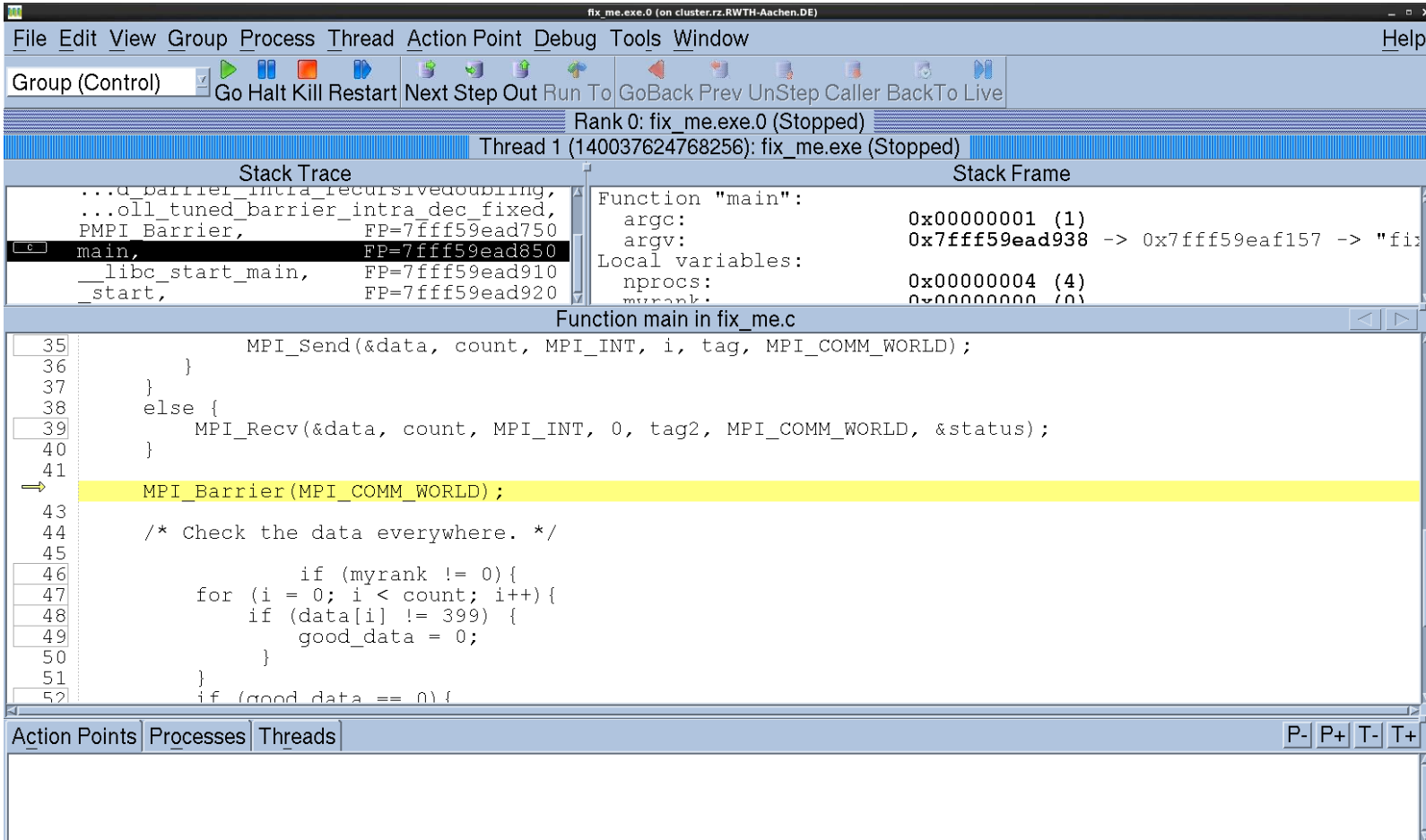
Registers for the frame:

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

Function PMPI_Barrier in libmpi.so.1

```
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
```

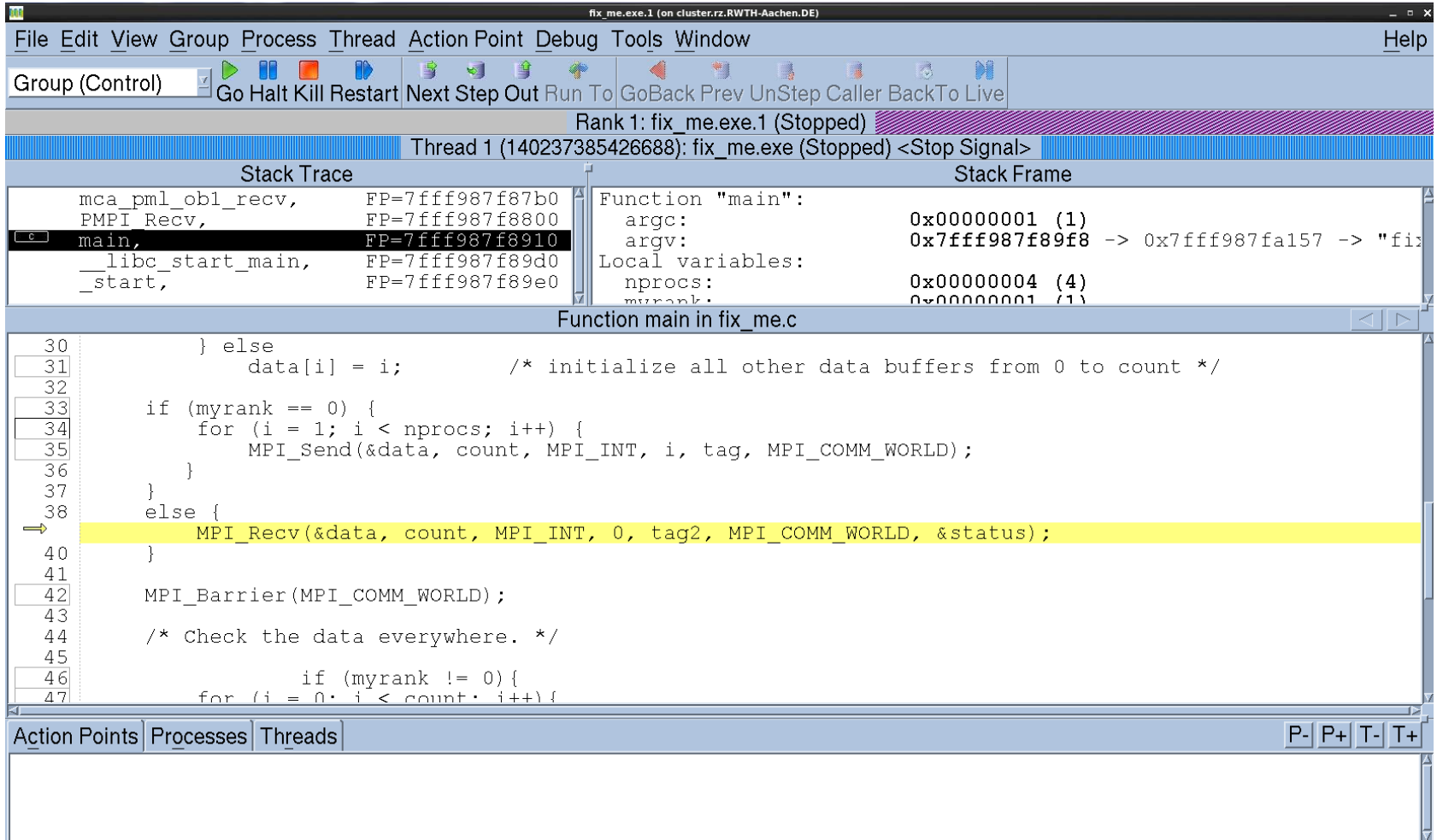
Rank 0 is waiting in barrier



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. The code window shows the following code:

```
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



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		Stack Frame	
mca_pml_obl_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)

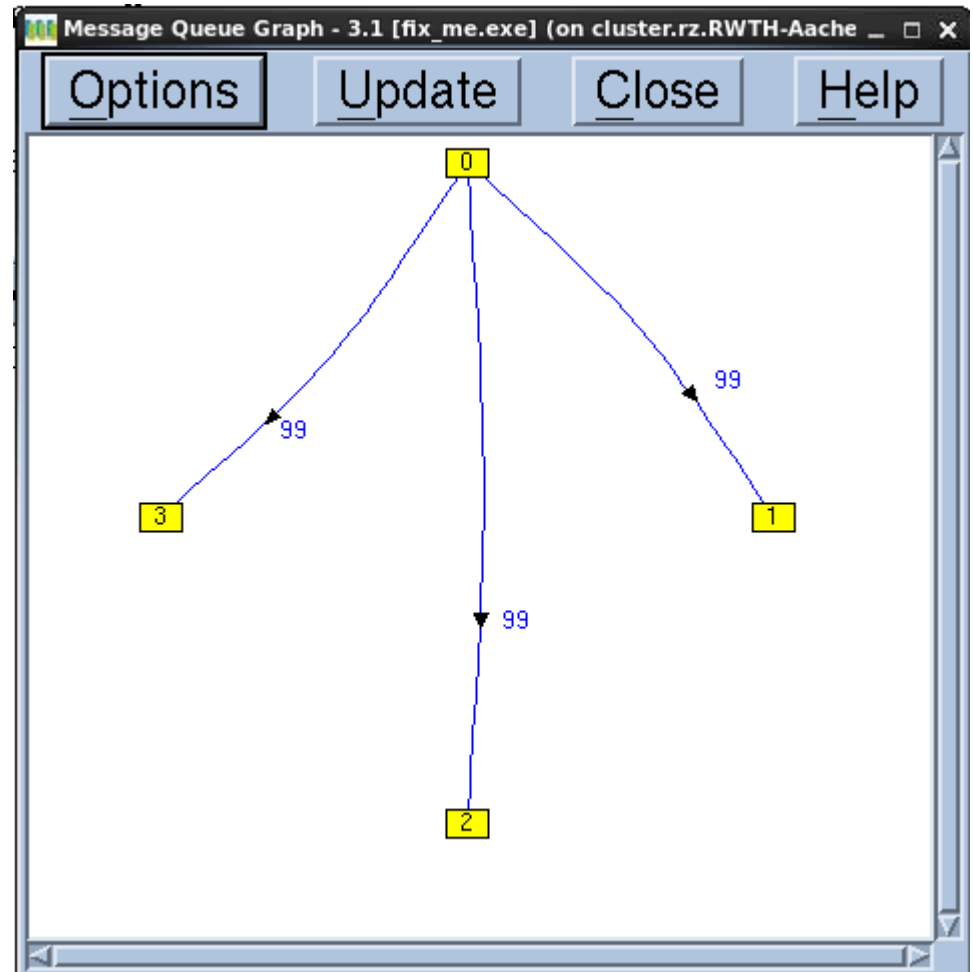
Function main in fix_me.c

```
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 {
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++)!
```

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

■ 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”

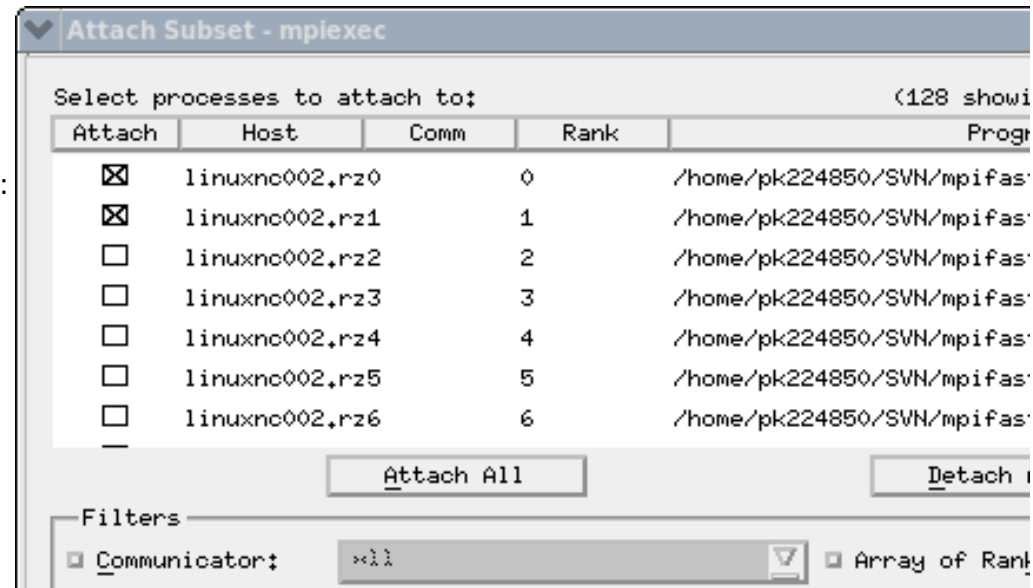
“When a job goes parallel” menu set:

[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

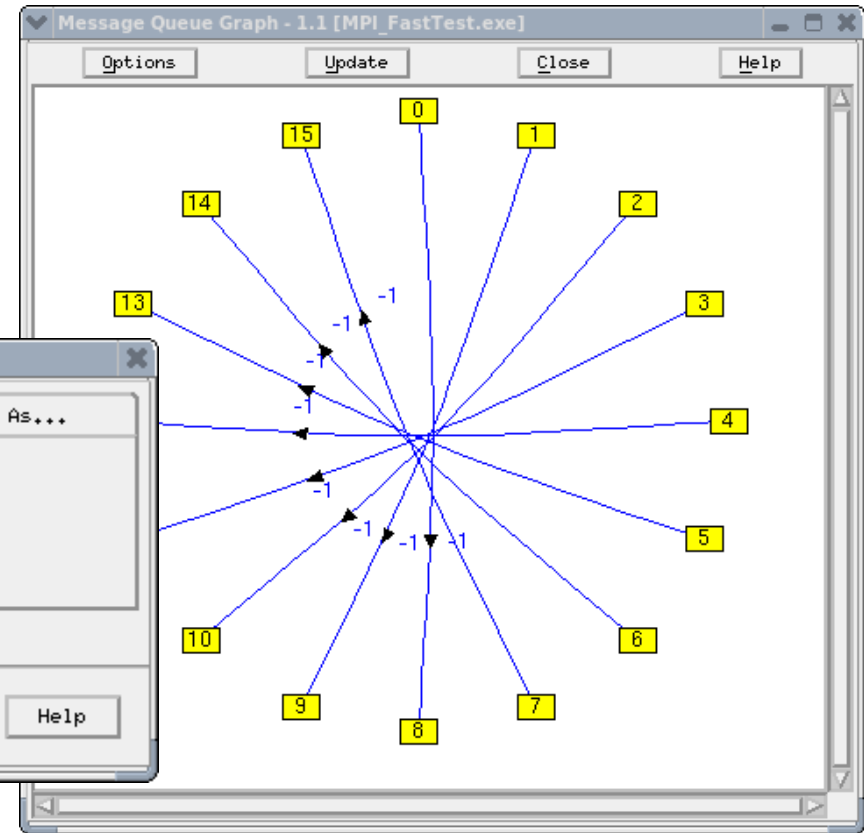
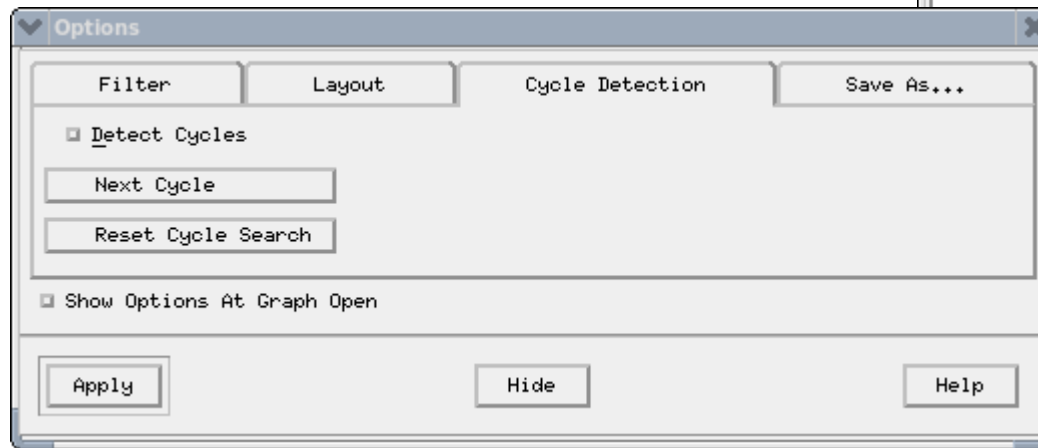
Find deadlocks:

“Options” → „Cycle Detection“

Green: Pending Sends

Blue: Pending Receives

Red: Unexpected Messages



■ Debugging for

→ Memory leaks
(Leak detection)

→ Double free

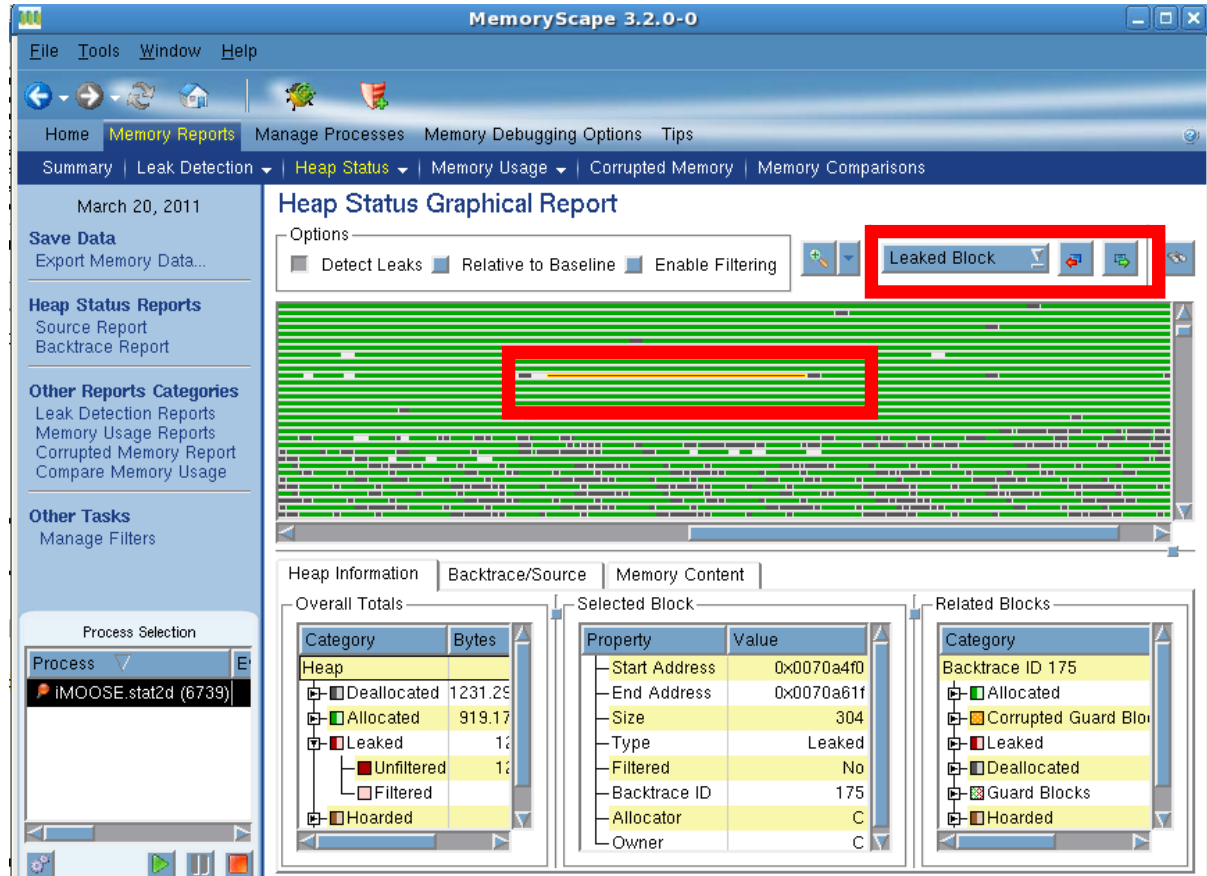
→ Invalid array
bounds

■ Memory reports

→ Consumption

→ Comparisons

→ ...



MemoryScape 3.2.0-0

File Tools Window Help

Home Memory Reports Manage Processes Memory Debugging Options Tips

Summary Leak Detection Heap Status Memory Usage Corrupted Memory Memory Comparisons

March 20, 2011

Save Data
Export Memory Data...

Heap Status Reports
Source Report
Backtrace Report

Other Reports Categories
Leak Detection Reports
Memory Usage Reports
Corrupted Memory Report
Compare Memory Usage

Other Tasks
Manage Filters

Process Selection

Process iMOOSE.stat2d (6739)

Heap Status Graphical Report

Options

Detect Leaks Relative to Baseline Enable Filtering

Leaked Block

Heap Information Backtrace/Source Memory Content

Category	Bytes
Heap	
Deallocated	1231.29
Allocated	919.17
Leaked	12
Unfiltered	12
Filtered	
Hoarded	

Property	Value
Start Address	0x0070a4f0
End Address	0x0070a61f
Size	304
Type	Leaked
Filtered	No
Backtrace ID	175
Allocator	C
Owner	C

Category
Backtrace ID 175
Allocated
Corrupted Guard Block
Leaked
Deallocated
Guard Blocks
Hoarded

■ Using a debugger enhances productivity

■ TotalView helps you!

- Serial programs
- Threaded (OpenMP) programs
- MPI programs
- Hybrid programs
- Intel Xeon Phi
- GPGPUs

■ Online information

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

<http://www.itc.rwth-aachen.de/hpc/primer>

Thank you for your attention!

