Debugging with TotalView

Tim Cramer

17.03.2016

What is TotalView?

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

TOTALVIEW TECHNOLOGIES
A Rogue Wave Software Company

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

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

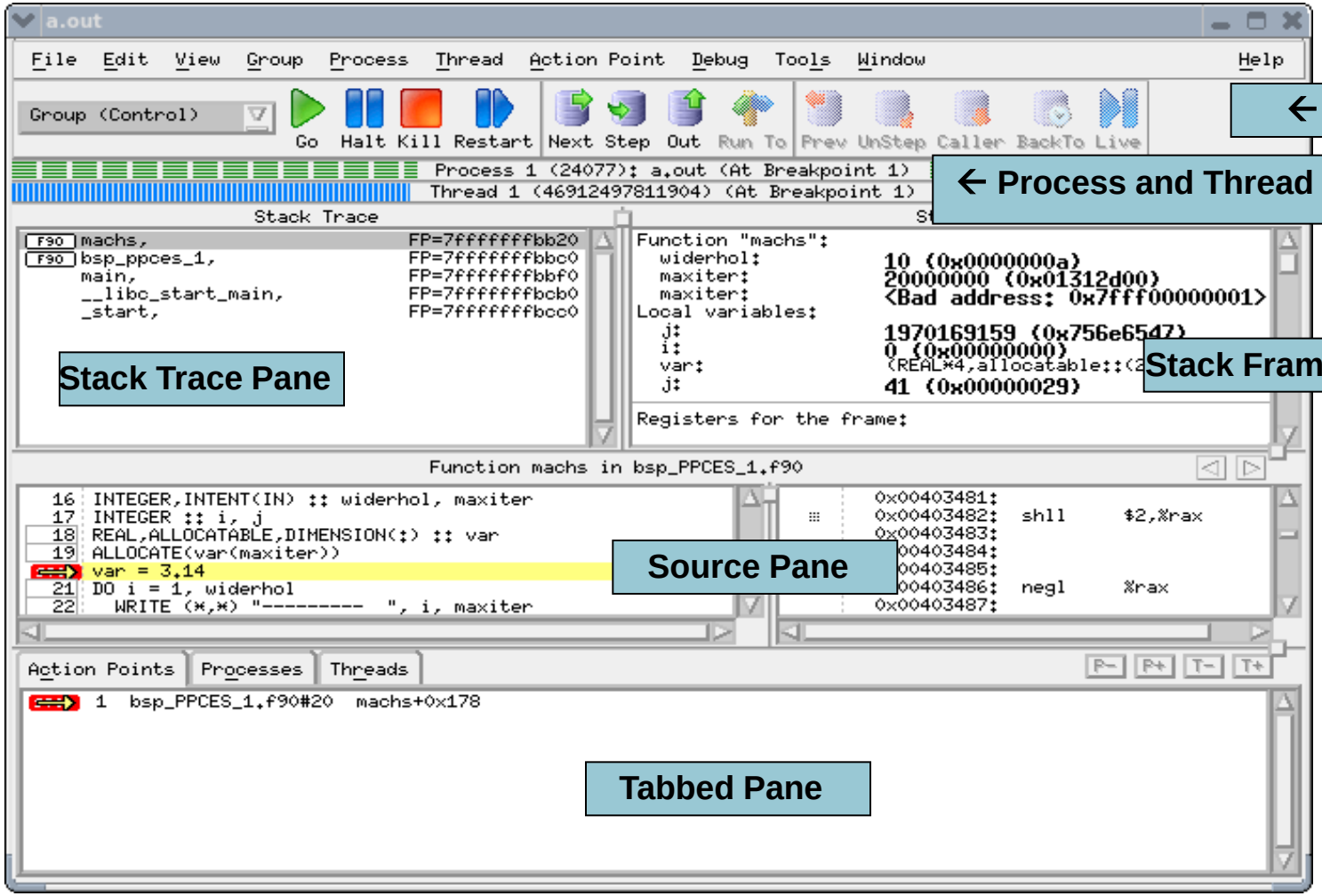
  -c Core file size (crucial for debugging on core files)

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

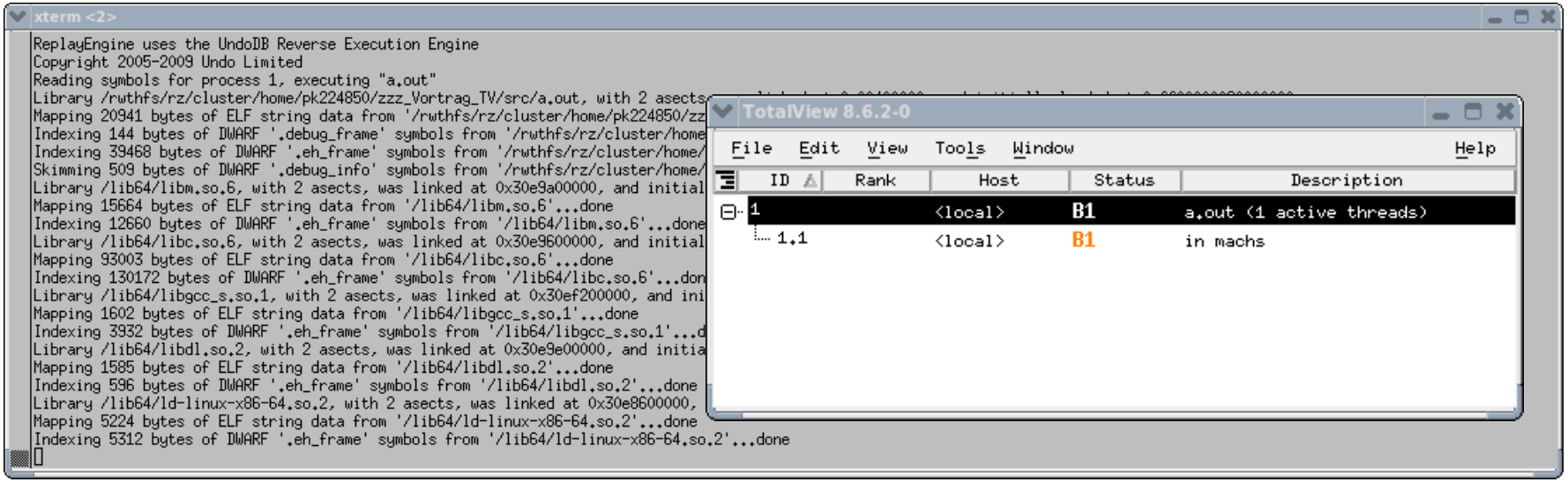
TotalView in the RWTH Environment

- **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

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)

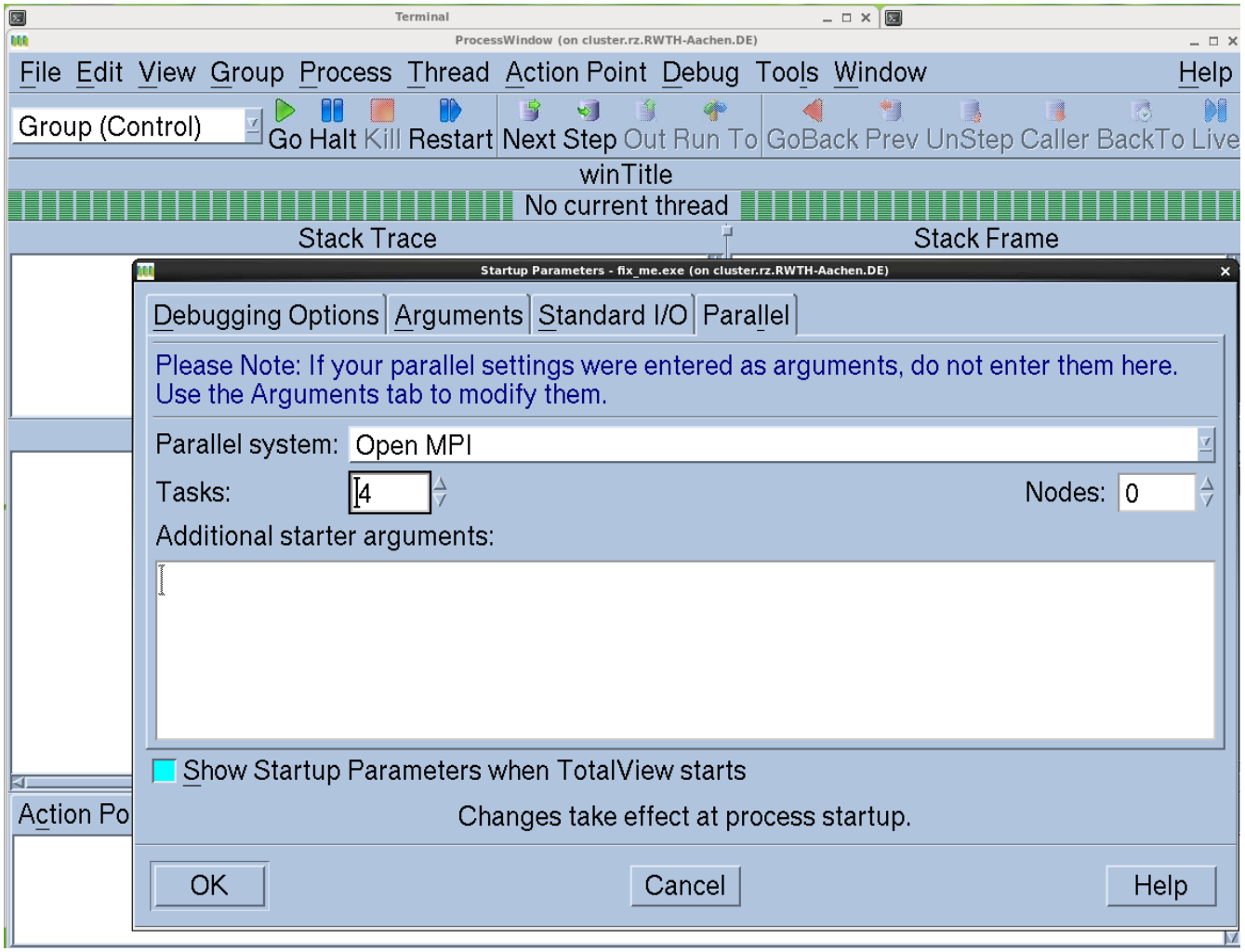
- **Nevertheless, TV is better than no TV**
  - → Stack frames for every process / thread in one GUI
  - → Switching between processes / threads (even for accelerators like GPGPUs)
  - → Variable inspection (and visualization) across all processes / threads
  - → Deadlock detection
  - → Visualization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

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

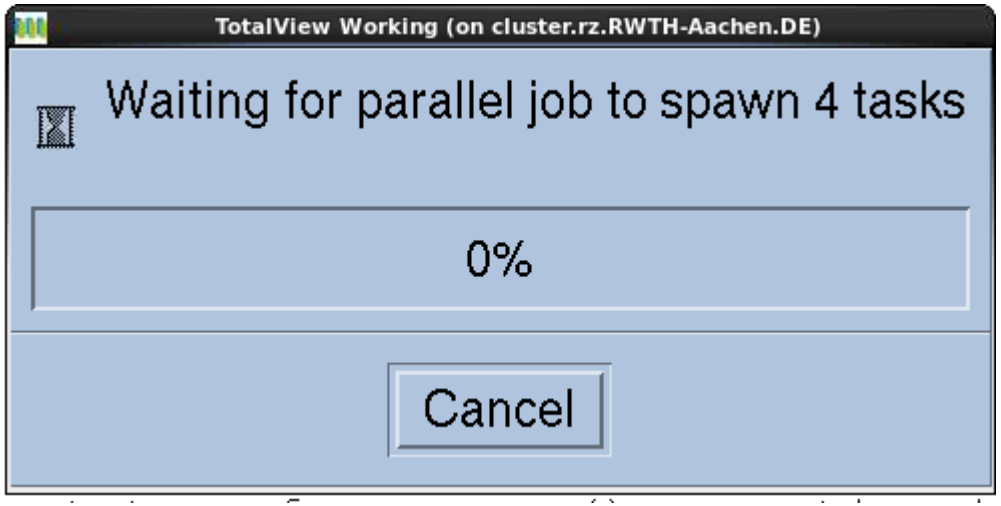
Live Demo Parallel Debugging

L3: MPI Debugging

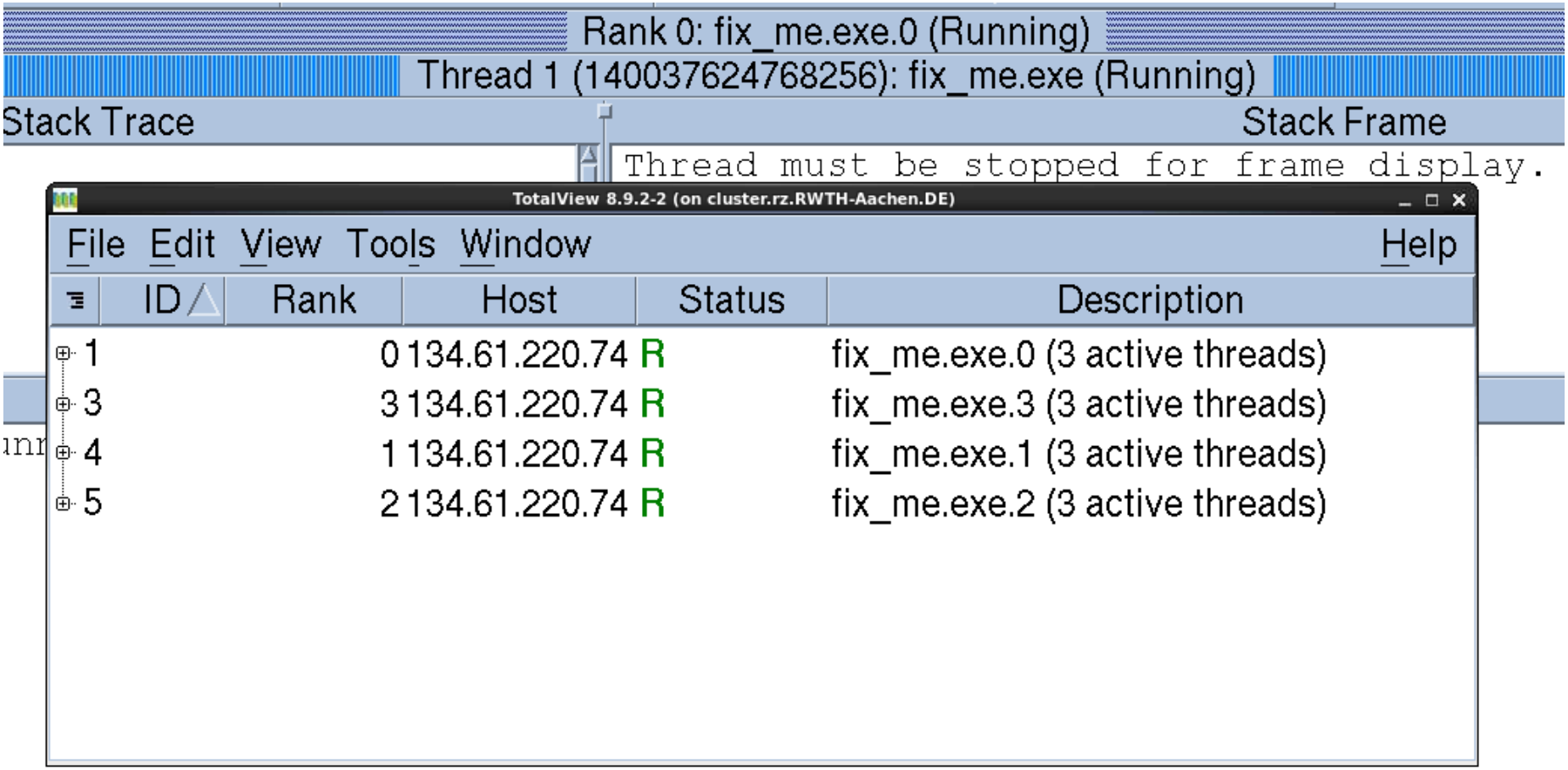


L3: MPI Debugging

- **Be patient**

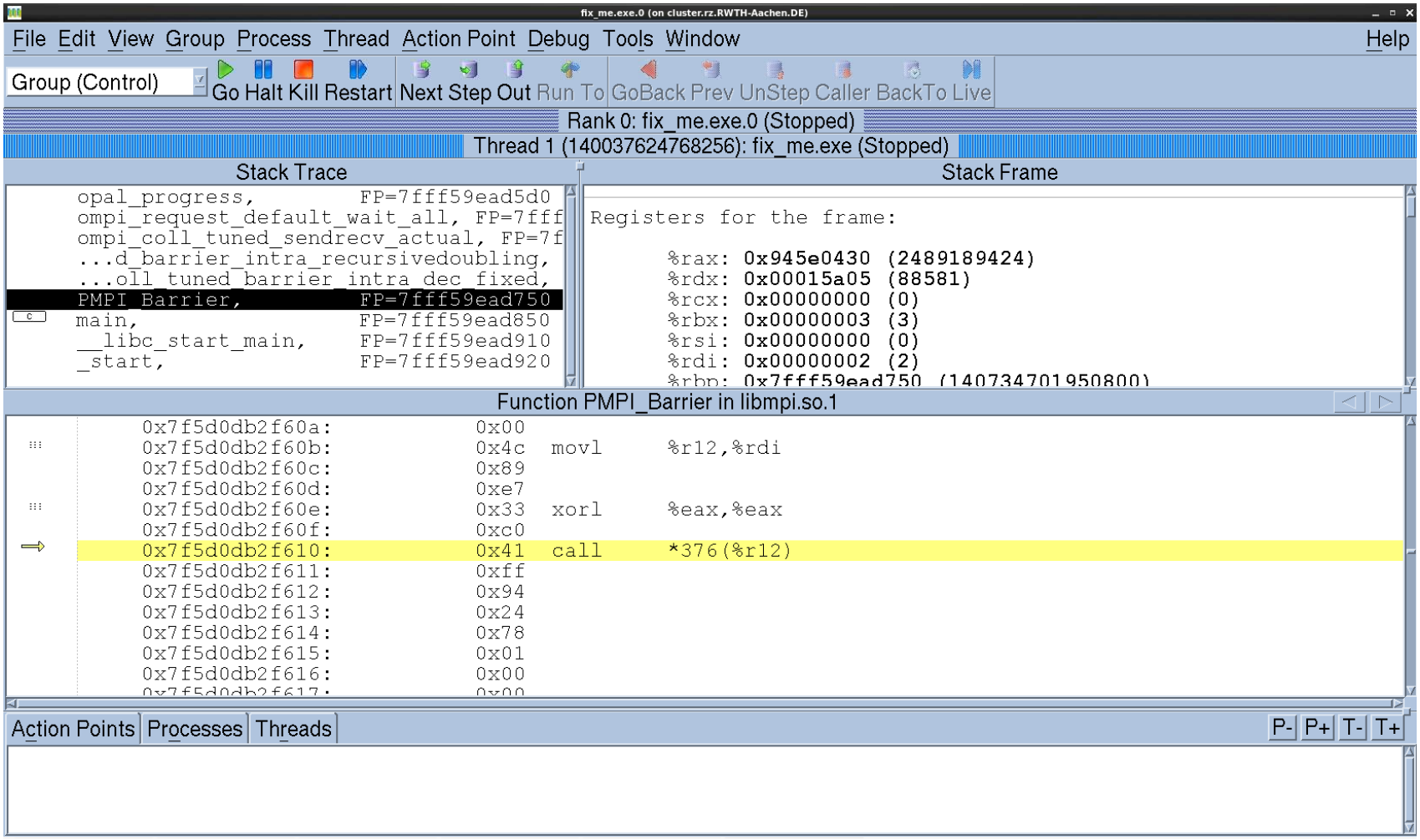


L3: MPI Debugging



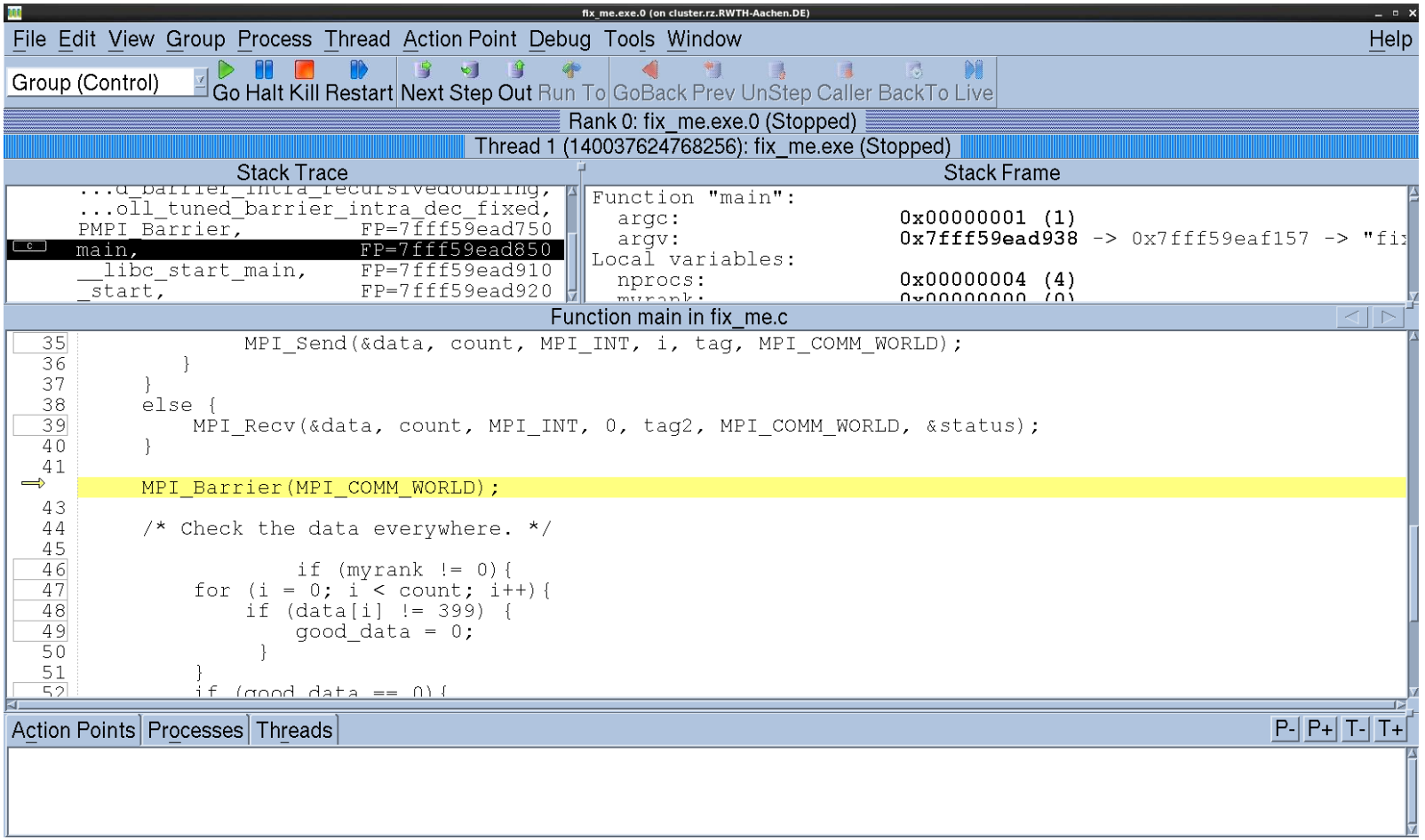
L3: MPI Debugging

- **Press the “HALT” button**

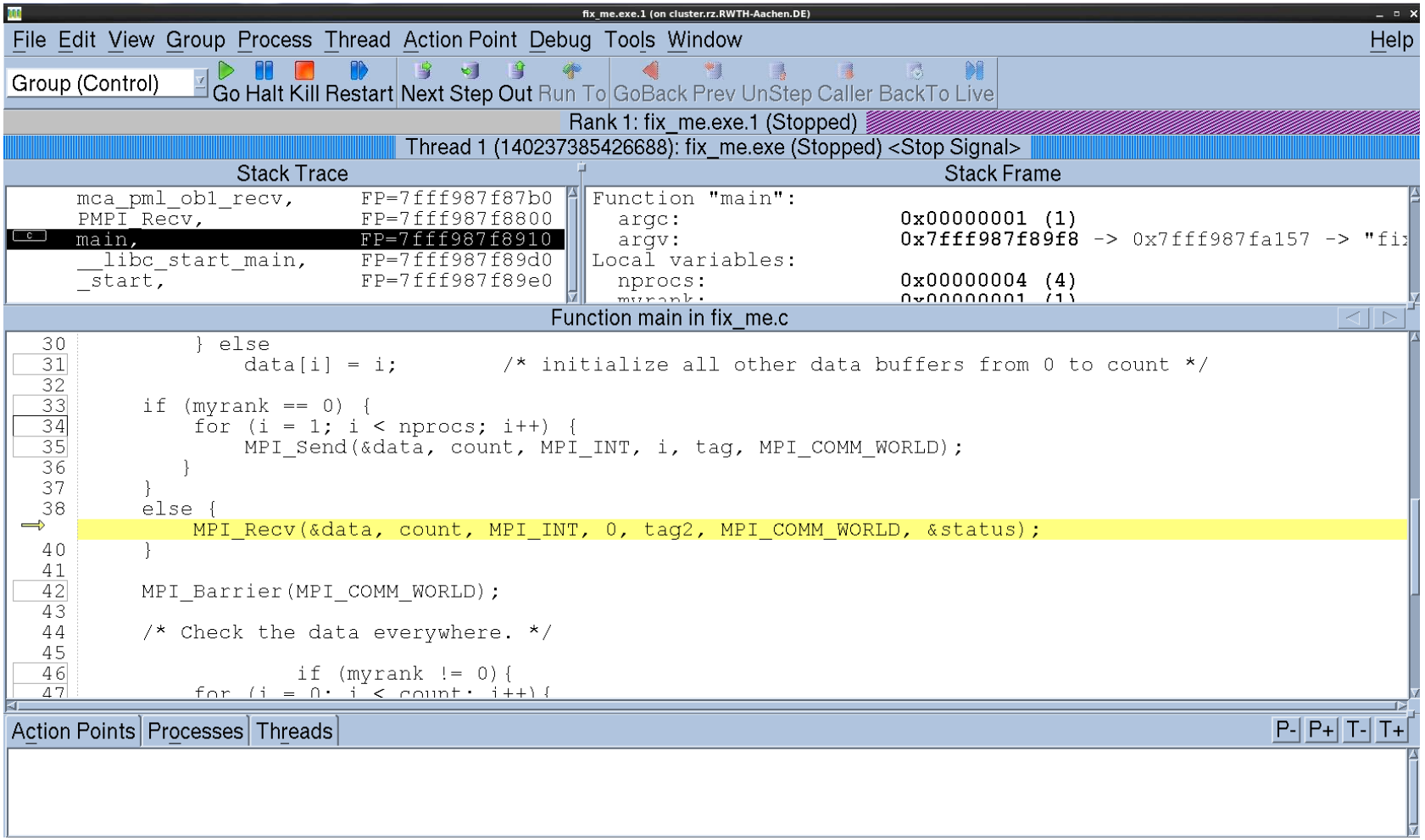


L3: MPI Debugging

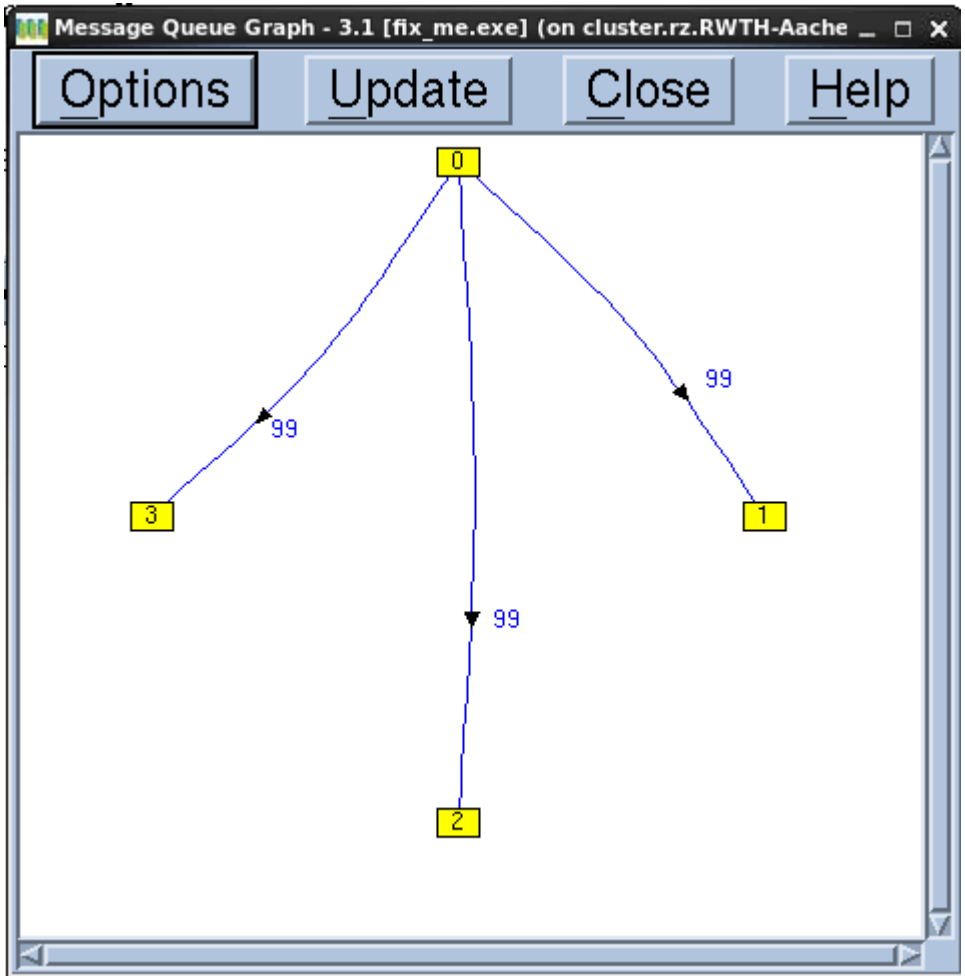
- **Rank 0 is waiting in barrier**



Rank 1 still waits for data



- **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"

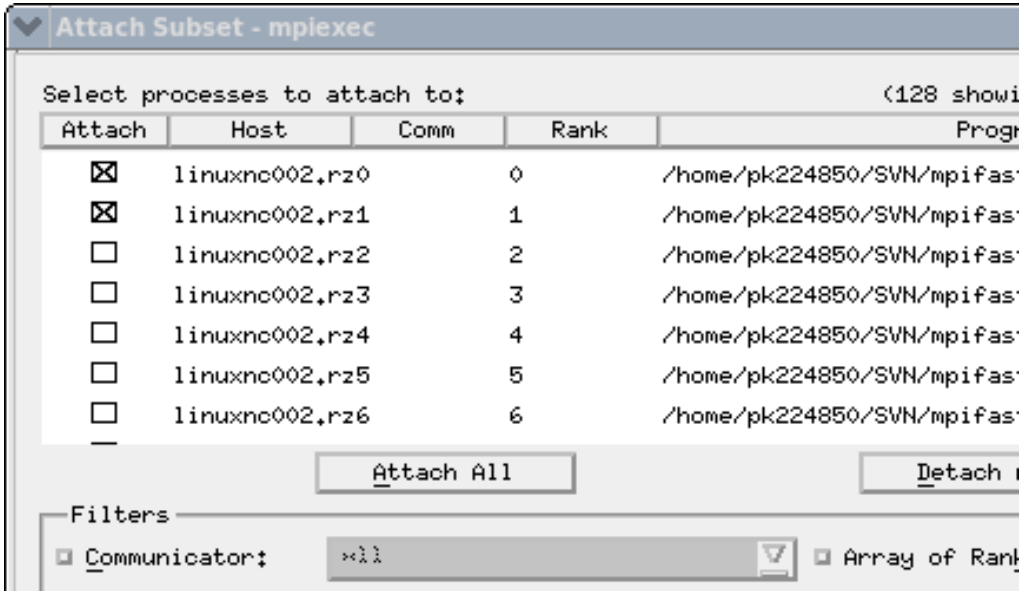
    "When a job goes parallel" menu set:

    [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

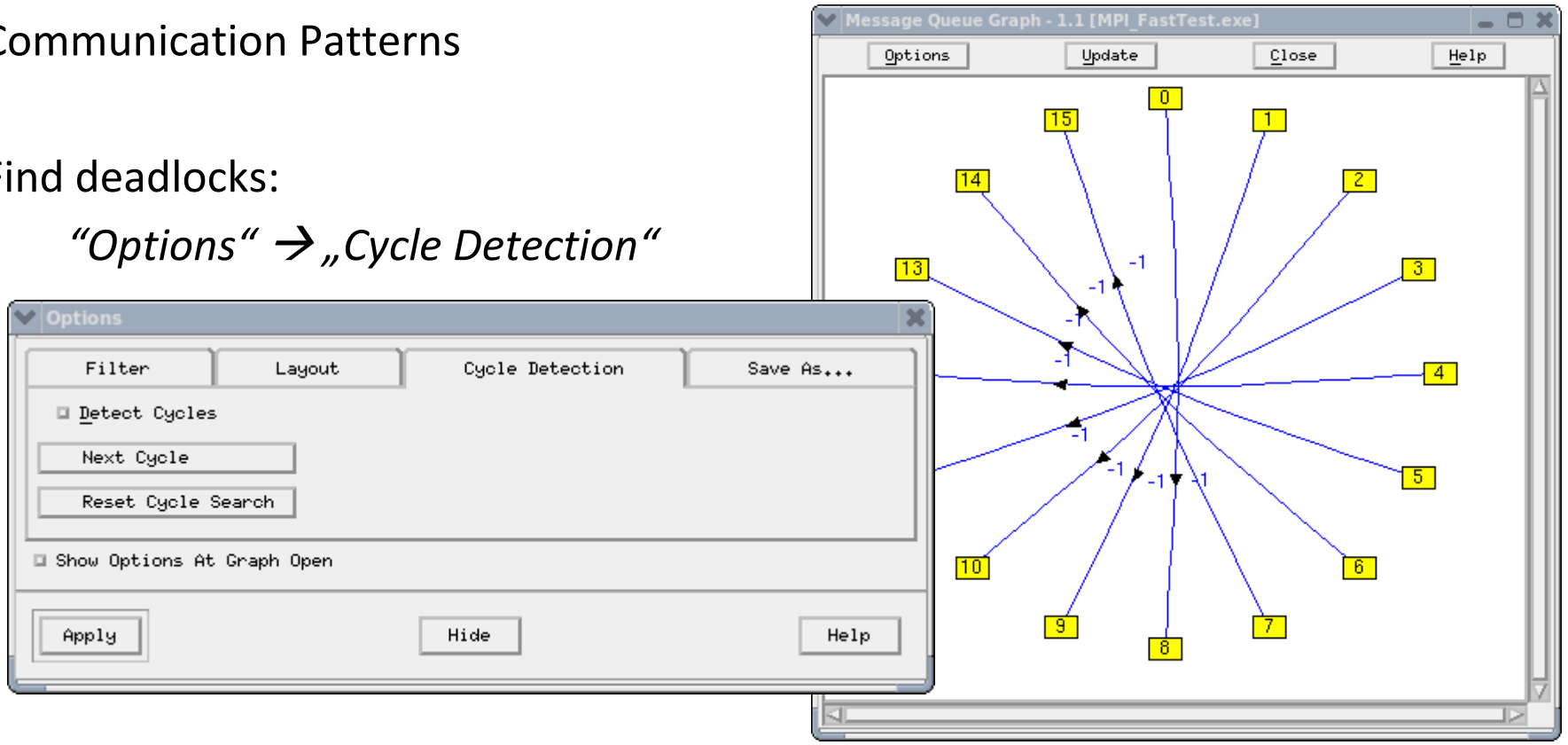
Green: Pending Sends

Blue: Pending Receives

Red: Unexpected Messages

Find deadlocks:

“Options“ → „Cycle Detection“



The End

Thank you for your attention!