Debugging with Totalview

Martin Čuma
Center for High Performance Computing University of Utah
mcuma@chpc.utah.edu

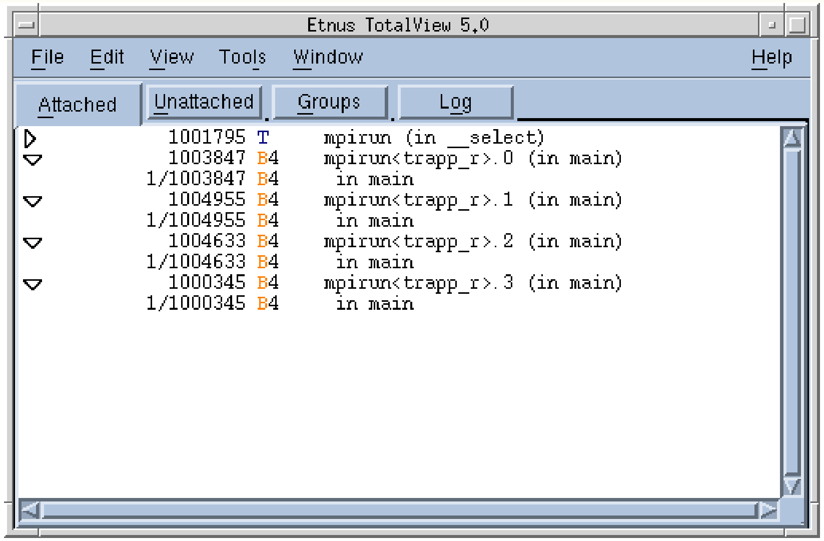
Overview

- Totalview introduction.
- Basic operation.
- Serial debugging.
- Parallel debugging.
  - OpenMP
  - MPI
- Memory debugging.
- Some other useful information

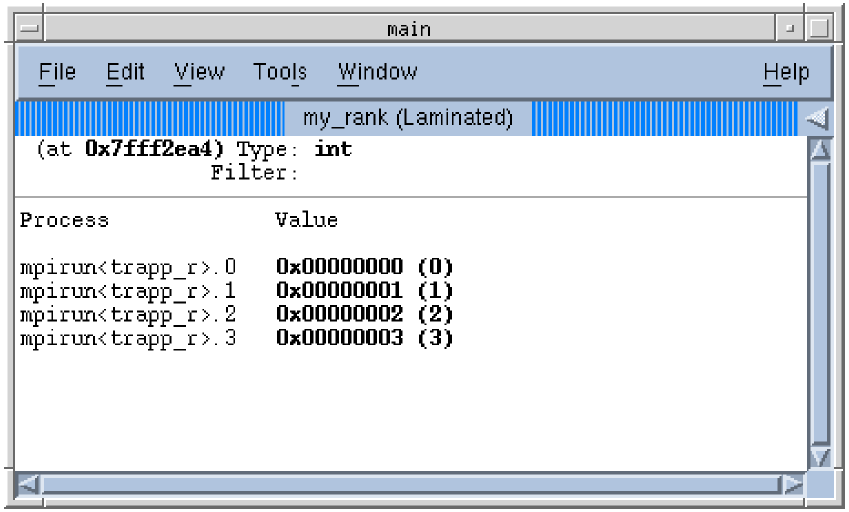
Totalview in a nutshell

- source and machine level debugger
- command line and graphic interface
- serial and parallel debugging support
- supports remote debugging
- runs on variety of platforms

Totalview windows



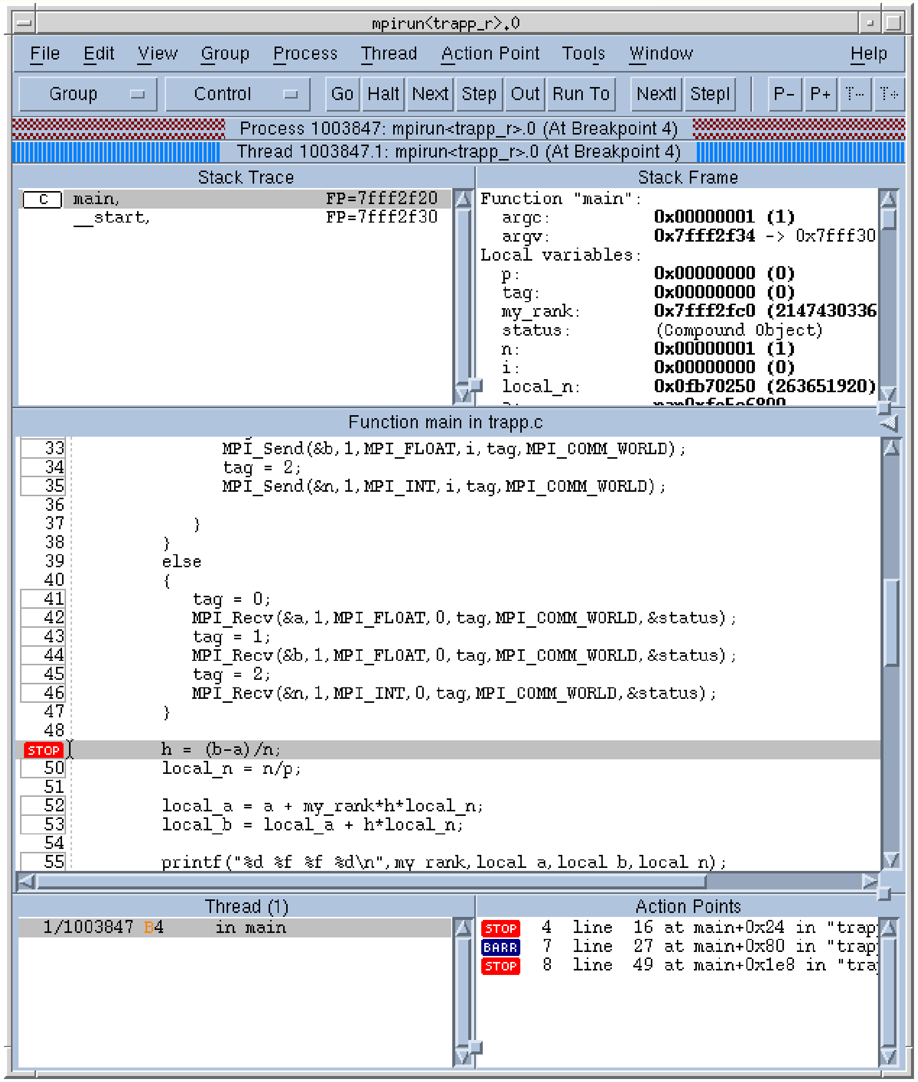
```
▷          1001795 T     mpirun (in __select)
▽          1003847 B4    mpirun<trapp_r>.0 (in main)
         1/1003847 B4     in main
▽          1004955 B4    mpirun<trapp_r>.1 (in main)
         1/1004955 B4     in main
▽          1004633 B4    mpirun<trapp_r>.2 (in main)
         1/1004633 B4     in main
▽          1000345 B4    mpirun<trapp_r>.3 (in main)
         1/1000345 B4     in main
```



```
 (at 0x7fff2ea4) Type: int
                Filter:

Process             Value

mpirun<trapp_r>.0   0x00000000 (0)
mpirun<trapp_r>.1   0x00000001 (1)
mpirun<trapp_r>.2   0x00000002 (2)
mpirun<trapp_r>.3   0x00000003 (3)
```



Process 1003847: mpirun<trapp_r>.0 (At Breakpoint 4)

Thread 1003847.1: mpirun<trapp_r>.0 (At Breakpoint 4)

```
C  main,                 FP=7fff2f20
   __start,              FP=7fff2f30
```

```
Function "main":
  argc:           0x00000001 (1)
  argv:           0x7fff2f34 -> 0x7fff30
Local variables:
  p:              0x00000000 (0)
  tag:            0x00000000 (0)
  my_rank:        0x7fff2fc0 (2147430336
  status:         (Compound Object)
  n:              0x00000001 (1)
  i:              0x00000000 (0)
  local_n:        0x0fb70250 (263651920)
```

```
                MPI_Send(&b,1,MPI_FLOAT,i,tag,MPI_COMM_WORLD);
                tag = 2;
                MPI_Send(&n,1,MPI_INT,i,tag,MPI_COMM_WORLD);

            }
        }
        else
        {
            tag = 0;
            MPI_Recv(&a,1,MPI_FLOAT,0,tag,MPI_COMM_WORLD,&status);
            tag = 1;
            MPI_Recv(&b,1,MPI_FLOAT,0,tag,MPI_COMM_WORLD,&status);
            tag = 2;
            MPI_Recv(&n,1,MPI_INT,0,tag,MPI_COMM_WORLD,&status);
        }

STOP    h = (b-a)/n;
        local_n = n/p;

        local_a = a + my_rank*h*local_n;
        local_b = local_a + h*local_n;

        printf("%d %f %f %d\n",my_rank,local_a,local_b,local_n);
```

```
1/1003847 B4     in main
```

```
STOP  4  line  16 at main+0x24 in "trap
BARR  7  line  27 at main+0x80 in "trap
STOP  8  line  49 at main+0x1e8 in "tra
```

Totalview basic operations

- Data examination
  - view data in the variable windows
  - change the values of variables
  - modify display of the variables
  - visualize data
- Action points
  - breakpoints and barriers (static or conditional)
  - watchpoints
  - evaluation of expressions

Multiprocess debugging

- Automatic attachment of child processes
- Create process groups
- Share breakpoints among processes
- Process barrier breakpoints
- Process group single-stepping
- “Laminate” variables
- Display MPI message queue state

How to use Totalview

1. Compile binary with debugging information

- flag -g

```
g77 -g test.f -o test
```

- if use fork() or execve(), link …

```
g77 -g -L/…/totalview/linux-x86-64/lib -ldbfork
test.f -o test
```

2. Run Totalview

- TV + executable

```
totalview executable
```

- TV + core file

```
totalview executable core_file
```

How to use Totalview

- run TV and attach the executable

  - start TV

  - menu New Program Window

  - fill in executable file name

- run TV and attach running program

  - start TV

  - menu New Program Window

  - pullout Attach to an Existing Process

  - choose process ID and fill in executable file name

3. Totalview operation

- left mouse button - select
- right mouse button - menu
- left mouse button double click - dive

Example 1 – Serial code

- CUSP – DFT plane wave atomic simulation
  - atomic nuclei interact classically
  - electrons described by wavefunctions - plane waves
  - interaction between w.f.'s and nuclei define the system potential and kinetic energy
  - iterate electronic and nuclear energy to self consistency to achieve ground state
- Implementation
  - 10+ Fortran source code files
  - simple Makefile w/ flags **-g -byteswapio**

Process view window

- Stack trace – procedure hierarchy
- Stack frame – variables display
- Source code – code + process navigation
- Threads list – in case of multithreaded application
- Action points – list of breakpoints, barriers,…

Running the debugger

- Menu Go/Halt/Next/Step/Hold or shortcuts
- Possible actions (thread,process/group):
- go (g/G)
- halt (h/H)
- step (source line) (s/S)
- step (instruction) (i/I)
- next (source line) (n/N)
- next (instruction) (x/X)
- run (to selection) (r/R)
- return (out of function) (o/O)

Action points

- Breakpoints and barriers
  - toggle location with left mouse (shift for barrier)
  - right-click – Properties for options
- Evaluation points
  - set conditional breakpoints
  - conditionally patch out code
- Watchpoints
  - watch for change in a memory location

Data examination

- Variable view
  - dive (right mouse) on any variable
  - change data type
  - select an array slice, e.g. (3:3,:)
  - filter array values, e.g. .ne. 0
- Variable visualization
  - menu Visualize – only up to 2D arrays

Example 2 – OpenMP code

- Compilation
  - Arches:

```
pathf90 -openmp test.f -g -o test.exe
pgf90 -mp test.f -g -o test.exe
ifort -openmp test.f -g -o test.exe
gcc4 -fopenmp test.c -g -o test.exe
```

- Running
  - set OMP_NUM_THREADS
- Example – saxpy routine
  - simple vector-scalar product

OpenMP specific debugging

- TV automatically attaches all threads
- put breakpoint to OpenMP parallel section to debug threads
- variable lamination - show values from all threads in one window – does not always work
- barrier points – shift-left click
- ambiguous action points – select all

Example 3 – MPI code

- Gravity and magnetic field inversion
  - Just a code that I am working on now
- Implementation
  - 1D domain decomposition
  - Each process has some slices of the 3D domain
  - MPI used for data synchronization, etc

Example 3 – MPI code – Arches

- MPICH2 on interactive nodes
  - good enough for most parallel problems
  - **`source /uufs/chpc.utah.edu/sys/pkg/mpich2/std_pgi/etc/mpich2.csh`**
  - **`mpicc submit.o –g -o submit`**

- Running
  - Start MPD daemon on the interactive node

    `mpdboot –n 1`

    if have problems, create ~/.mpd.conf file
  - Run:

    open Totalview (**`totalview`**), then in New Program Window enter:
    - program name
    - MPICH2 for parallel sysem
    - number of processors

Example 3 – MPI code – Arches

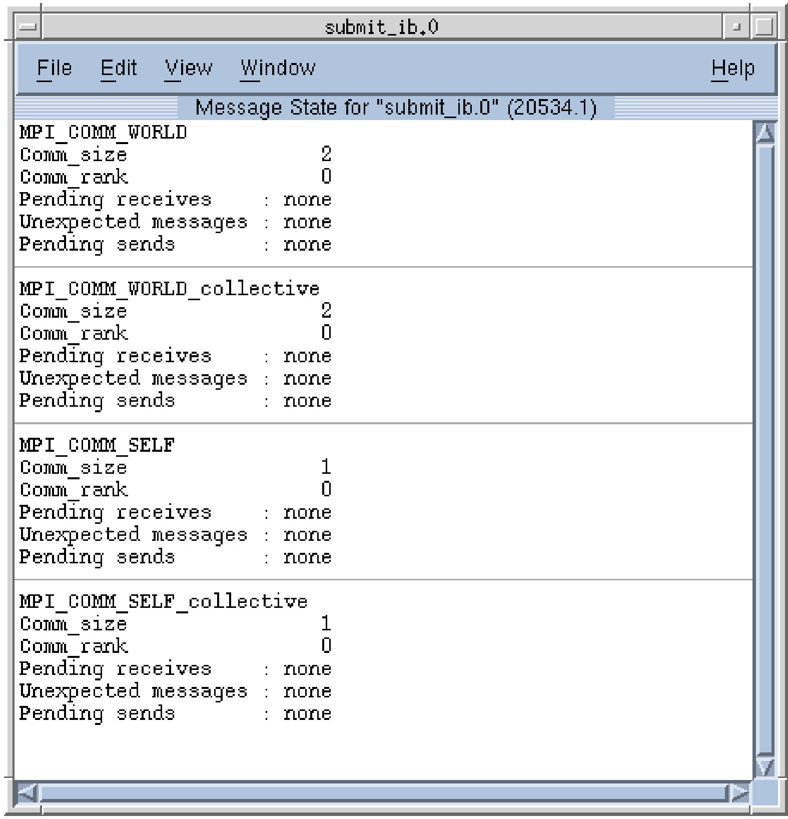
- Running on compute nodes
  - use larger number of processors or other MPI than MPICH2 is needed
  - `source .../etc/mpi*.csh`
  - `mpicc -g -mp submit.c`
  - `mpicc submit.o -o submit`

- Running on compute nodes (large # of processors)
  - CHPC has 32 process Totalview license
  - interactive PBS:

```
qsub -I –X –l nodes=4:ppn=4,walltime=2:00:00
```

MPI specific debugging

- Process synchronization – program groups
- Barrier points
- Message queue state graph and display

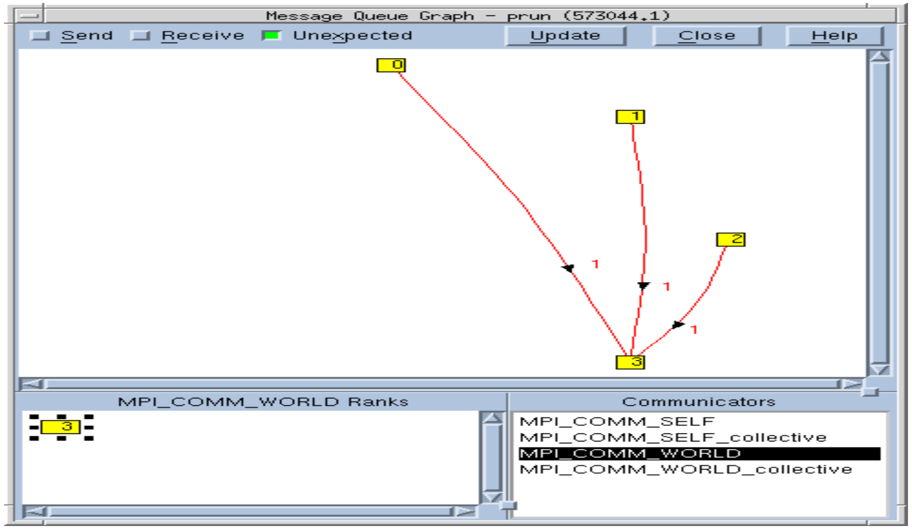


```
Message State for "submit_ib.0" (20534.1)
MPI_COMM_WORLD
Comm_size                  2
Comm_rank                  0
Pending receives     :  none
Unexpected messages  :  none
Pending sends        :  none

MPI_COMM_WORLD_collective
Comm_size                  2
Comm_rank                  0
Pending receives     :  none
Unexpected messages  :  none
Pending sends        :  none

MPI_COMM_SELF
Comm_size                  1
Comm_rank                  0
Pending receives     :  none
Unexpected messages  :  none
Pending sends        :  none

MPI_COMM_SELF_collective
Comm_size                  1
Comm_rank                  0
Pending receives     :  none
Unexpected messages  :  none
Pending sends        :  none
```



```
MPI_COMM_SELF
MPI_COMM_SELF_collective
MPI_COMM_WORLD
MPI_COMM_WORLD_collective
```

MemoryScape

- Dynamic memory debugging tool
- display memory status
- paint allocated and deallocated blocks
- find memory leaks
- identify dangling pointers
- enable with `Tools > Memory Debugger > Enable memory debugging` checkbox

Replay Engine

- Allows to reversely debug the code
- Must be turned on at the start of debugging session
- Run to the error, then backtrack to the source of the problem
- Helps to capture race conditions and other hard to reproduce bugs

CUDA debugging

- Debug Nvidia CUDA on GPU
- We haven’t tested it but according to come users it works

Some useful resources

- TotalviewTech webpage

  http://www.roguewave.com/products/totalview

- Location of Totalview

  Arches:**/uufs/chpc.utah.edu/sys/pkg/totalview/std**

  Some group desktops: **inquire at CHPC**

- Documentation

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

http://www.chpc.utah.edu/software/docs/par_devel.html

http://www.chpc.utah.edu/software/docs/totalview.html

http://www.chpc.utah.edu/short_courses/Totalview

Totalview Student Edition

- Free for students
- Limited to one computer, 4 processes
- To sign up, e-mail mcuma@chpc.utah.edu:
  - name
  - e-mail
  - university ID
  - anticipated year of graduation
- More details

`http://www.totalviewtech.com/academia/totalview_express.html`

Code checkers

- compilers check for syntax errors
  - some compiler flags help too (-C)
- memory checking tools - many errors are due to bad memory management
  - valgrind – easy to use
  - purify – harder to use

Thread checking

- Good for finding OpenMP errors
  - race conditions
  - privatization
- Intel thread checker (ITC)
  - OpenMP
  - pthreads

Intel software development products

- We have a free classroom license
- Tools for all stages of development
  - Compilers and libraries
  - Verification tools
  - Profilers
- More info

http://software.intel.com/en-us/intel-sdp-home/

Intel Inspector

- Thread checking
  - Data races and deadlocks
- Memory checker
  - Like leaks or corruption
- Standalone or GUI integration
- More info

http://software.intel.com/en-us/articles/intel-inspector-xe/

Intel Inspector

- Source the environment

```
source /uufs/chpc.utah.edu/sys/pkg/intel/
composerxe-2011.1.107/bin/compilervars.csh intel64
addpath "/uufs/chpc.utah.edu/sys/pkg/intel/inspector_xe_2011/
bin64:$PATH"
```

- Compile with `–tcheck -g`

```
ifort -openmp -tcheck -g trap.f
```

- Run tcheck

  `inspxe-gui` – graphical user interface

  `inspxe-cl` – command line

- Tutorial

`http://software.intel.com/sites/products/documentation/hpc/inspectorxe/en-us/lin/start/index.htm`

Intel VTune Amplifier

- Serial and thread profiler
- Built in hardware counters
  - Find hardware related slowdowns
- Source the environment

```
addpath "/uufs/chpc.utah.edu/sys/pkg/intel/
vtune_amplifier_xe_2011/bin64:$PATH"
```

- Run Vtune

  `amplxe-gui` – graphical user interface

  `amplxe-cl` – command line

- More info

```
http://software.intel.com/en-us/articles/intel-vtune-
amplifier-xe/
```

Intel tools summary

- We have license for more-less everything
- Installed in the /uufs/chpc.utah.edu branch
- Visible on all clusters and Linux desktops
- See the products description and documentation

http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/

- Let us know if you are interested in hands on tutorial