

# Tutorial on Parallel Debugging

## Victor Eijkhout

### SSiASC 2016

# Compiling for debug

Enable debug mode

```
mpicc -g -O2 yourprogram
```

Debug option can be used with any optimization level, but use

```
mpicc -g -O0 yourprogram
```

to prevent confusion.

(Compiler 'optimizes away' code at higher levels.)

# Your minimal debugger

```
mpirun -np 4 xterm -e gdb yourprogram
```

Pops up 4 xterms.

Great for debugging on your laptop.

Not great at scale.

# The DDT debugger

Load the module:

```
module load ddt
```

Call the debugger:

```
ddt yourprogram
```


Make sure you have an X forwarding connection:

```
ssh -X you@stampede.tacc.utexas.edu
```

or use VNC.

# Setup

**Job Submission Settings**

Submission template file:  

Submit command:

Regexp for job id:

Cancel command:

Display command:

☒ Quick Restart [What is Quick Restart?](#)

Wall Clock Limit:

Queue:

Project:

- Project: for this class, but later of your own
- Queue: development often quickest

# Run parameters

**Application:** /work/00434/eijkhout/pcse\_instructors/Labs2016/DDT/c/h/ Details

Application: /work/00434/eijkhout/pcse\_instructors/Labs2016/DDT/c/hang

Arguments:

☐ stdin file:

Working Directory:

☒ **MPI:** 12 processes, 1 node, MVAPICH 2 Details

Number of Processes: 12

Number of Nodes: 1

Implementation: MVAPICH 2 Change...

ibrun arguments:

☐ **OpenMP** Details

☐ **CUDA** Details

☒ **Memory Debugging:** Thorough, 1 guard page after, Backtraces, Ir Details...

☒ **Submit to Queue:** Wall Clock Limit=00:30:00 Configure... Parameters...

**Environment Variables:** none Details

**Plugins:** none Details

Help Options Submit Cancel

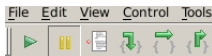
- MPI or OpenMP? Processes, nodes, threads.
- Memory debugging
- Commandline arguments

```

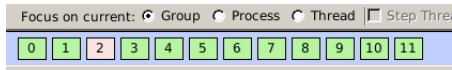
37  int main(int argc, char **argv) {
38      MPI_Comm comm;
39
40      MPI_Init(&argc, &argv);
41      comm = MPI_COMM_WORLD;
42
43      loop_for_awhile(comm);
44
45      MPI_Finalize();
46      return 0;
47  }
48

```

- Program starts at `MPI_Init`
- Use run controls



# Hanging processes



- Red: stopped at an interrupt or breakpoint
- Green: still running.  
All green but 'nothing happening': probably hanging program.
- Combination: some processes are not getting to the breakpoint: probably deadlocked.

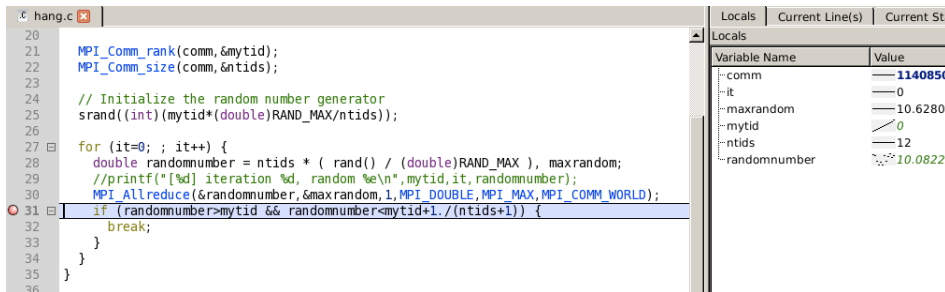


# Call stacks

Input/Output	Breakpoints	Watchpoints	Stacks	Tr
Stacks				
Processes	Function			
12	main (hang.c:43)			
11	loop_for_await (hang.c:30)			
1	loop_for_await (hang.c:35)			

- Hit the pause button, go to 'stacks' panel.
- Not every process is in the same source line.
- Click on process number to see what it's doing.

# Breakpoints



```
hang.c
20
21 MPI_Comm_rank(comm,&mytid);
22 MPI_Comm_size(comm,&ntids);
23
24 // Initialize the random number generator
25 srand((int)(mytid*(double)RAND_MAX/ntids));
26
27 for (it=0; ; it++) {
28     double randomnumber = ntids * ( rand() / (double)RAND_MAX ), maxrandom;
29     //printf("[%d] iteration %d, random %e\n",mytid,it,randomnumber);
30     MPI_Allreduce(&randomnumber,&maxrandom,1,MPI_DOUBLE,MPI_MAX,MPI_COMM_WORLD);
31     if (randomnumber>mytid && randomnumber<mytid+1./(ntids+1)) {
32         break;
33     }
34 }
35
36
```

Variable Name	Value
comm	114085
it	0
maxrandom	10.6280
mytid	0
ntids	12
randomnumber	10.0822

- Values display: everyone the same it
- value of mytid linearly increasing
- value of randomnumber all over the place.

# Exercise 1 (hang0)

- 1 `make hang0`
- 2 `ddt hang0`
- 3 Every process outputs a different value. Why?
- 4 Use stepping or breakpoints

## Exercise 2 (hang1)

- 1 the author of `hang0.c` has attempted to fix the program
- 2 `make hang1`
- 3 in DDT: `File > New Session`
- 4 run the program: only one process prints output.  
what are the others doing?
- 5 fix the program

## Exercise 3 (broadcast)

The author of `broadcast.c` was confused about the SPMD model.

- 1 The program as is will abort. Find the reason and fix.
- 2 Now the program will hang. Find out why and fix it.