



TECHNISCHE
UNIVERSITÄT
DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Parallel Debugging with Allinea DDT

Parallel Programming Course, Dresden, 8.- 12. February 2016

Matthias Lieber (matthias.lieber@tu-dresden.de)



Why using a Debugger?

- Your program...
 - terminates abnormally
 - produces wrong results
 - shows incomprehensible behavior

→ You want to know what your program is (really) doing

- Typical example: your program crashes with a segmentation fault

```
% icc myprog.c -o myprog  
% ./myprog  
Segmentation fault  
%
```

What's going wrong?

What can a Debugger do?

● Observe a running program:

- Print variables (scalars, arrays, structures / derived types, classes)
- Inform about current source code line and function (function call stack)

● Control the program execution:

- Stop the program at a specific source code line (**Breakpoints**)
- Stop the program by evaluating variable expressions (**Conditional Breakpoints** and **Watchpoints**)
- Stop the program before terminating abnormally
- Execute the program line-by-line (**Stepping**)

Typical Usage of a Debugger



- Compile the program with the **-g** compiler flag
 - **gcc -g myprog.c -o myprog**
- Run the program under control of the debugger:
 - **ddt ./myprog**
 - Locate the position of the problem and examine variables
 - Find out why the program shows unexpected behavior
- Edit the source code, modify parameters, etc.
- Repeat until problem is solved

Debugger Operation Modes

● Start program under debugger control

- Most common way to use a debugger
- Not useful if you want to observe what the program does after a long runtime

● Attach to an already running program

- Program was not started under debugger
- Useful if program has been running for a long time

● Core files / core dumps

- Core files are memory state of a crashed program written to disk
- Only static analysis of program's data after termination
- Useful if you don't expect a crash or don't want to wait until a crash happens (probably after long runtime)

Before you start using a Debugger...

- **Use compiler's check capabilities** like `-Wall` etc.
 - Read compiler's manual: `man {gcc|ifort|pgf90|...}`
 - Intel C: `-Wall -Wp64 -Wuninitialized -strict-ansi`
 - Intel Fortran: `-warn all -std95 -C -fpe0 -traceback`
- **Always compile your application with the `-g` flag**, especially during developing and testing
 - Adds symbolic debug info to binary, no performance impact
- Optimizations often interfere with debugging (e.g. functions or variables of interest are "optimized away")
 - If necessary, **compile with `-O0` to disable optimizations**

Allinea DDT (Distributed Debugging Tool)

- Commercial debugging tool by Allinea
- C, C++, Fortran
- Parallel Support: pThreads, OpenMP, MPI, PGAS languages, CUDA, OpenACC, Xeon Phi
- Available for all common HPC platforms
- Intuitive graphical user interface
- Advanced features:
 - Visualization of array contents
 - Memory debugging
 - Modify variables
- More info: <http://www.allinea.com>

Allinea DDT: MPI Program Start

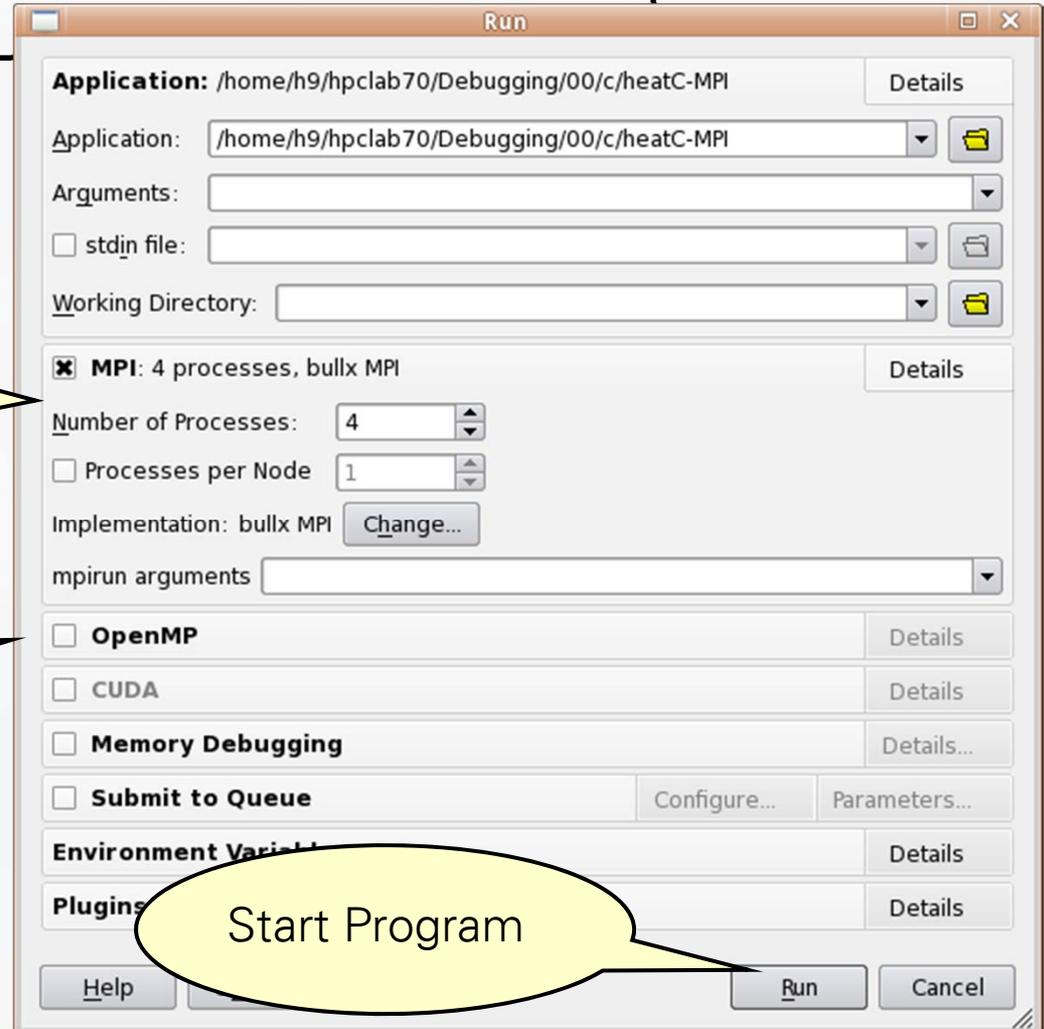
```
% mpicc -g -O0 heatC-MPI.c -o heatC-MPI
% ddt ./heatC-MPI
```

Compile with
Debugging

Start DDT

Set MPI
implementation and
number of MPI
processes

... and / or number
of OpenMP threads



Start Program

Allinea DDT: Main Window

The screenshot shows the Allinea DDT v3.2.1-27702 main window. The interface includes a menu bar (Session, Control, Search, View, Help), a toolbar with various debugging icons, and a status bar. The main area is divided into several panes:

- Source file browser:** Located on the left, showing the file tree for 'heatC-MPI.c'.
- Source view:** The central pane displaying the source code of 'heatC-MPI.c'. The current line is 429, which is highlighted in blue. The code includes comments and function calls like 'MPI_Init' and 'heatMPISetup'.
- Process and thread selection:** A toolbar at the top with buttons for selecting the current group (All), process, thread, or step threads.
- Variables pane:** Located on the right, showing the current line(s) and current stack. It displays variables like 'argc' with value 1 and 'argv' with value 0x7ffffffa948.
- Output, Breakpoints, Watchpoints, Call stack:** A pane at the bottom left showing the current stack of processes and threads. The current process is 'main (heatC-MPI.c:429)'.
- Evaluation window:** A pane at the bottom right for evaluating expressions and viewing their values.

Callouts in yellow speech bubbles point to these features:

- Process control: run, stop, stepping
- Source file browser
- Process and thread selection
- Source view
- Variables pane
- Output, Breakpoints, Watchpoints, Call stack
- Evaluation window

Allinea DDT: Process Control & Stepping

The screenshot shows the Allinea DDT interface with several callouts explaining its features:

- Run**: Points to the green play button in the toolbar.
- Pause**: Points to the yellow pause button in the toolbar.
- Step to next code line**: Points to the green arrow with a right-pointing arrowhead.
- Step over function calls**: Points to the green arrow with a right-pointing arrowhead and a small downward-pointing arrowhead.
- Step out of current function**: Points to the green arrow with a right-pointing arrowhead and a small upward-pointing arrowhead.
- Right mouse button at source code line -> „Run to here“**: Points to the source code editor.
- Select group / processes**: Points to the "All" group selection in the "Current Group" dropdown.
- Commands may affect whole group or single processes / threads**: Points to the "Focus on current" radio buttons (Group, Process, Thread).

The source code editor shows the following code:

```
418 /*****  
419 * Main program  
420 *****/  
int main(int argc, char *argv)
```

Allinea DDT: Segmentation Fault

Processes 2 and 3 crashed

Segmentation Fault!

This is the line where the program crashed

Hit "Pause" to stop the program

```
146 double dtheta;
147 double mymax = 0.0;
148
149 *dthetamax = 0.0;
150
151 /* calculate the time step: read from theta, write new timestep to thetanew */
152 /* Only calculate on a processes sub-grid */
153 for (x=mympi->start_x; x < mympi->start_x + mympi->subgrid_x; x++)
154 {
155     for (y=mympi->start_y; y < mympi->start_y + mympi->subgrid_y; y++)
156     {
157         dtheta = ( grid->theta[x-1][y] +
158                 grid->theta[x][y-1] +
159                 grid->theta[x][y] +
160                 grid->theta[x+1][y] ) / 4.0;
161         mymax = fmax(fabs(dtheta) - mymax); /* save the max */
162     }
163 }
164
165 /* Make MPI reduction to get max
166 MPI_Allreduce (&mymax, &dthetamax, 1, MPI_DOUBLE, MPI_MIN, MPI_COMM_WORLD);
167
168 /* update theta: copy thetanew to theta
169 for (x=mympi->start_x; x < mympi->start_x + mympi->subgrid_x; x++)
```

Process	Thread	Function
2	2	btl_openib_async_thread
2	2	main (heatC-MPI.c:456)
2	2	heatTimestep (heatC-MPI.c:157)
2	2	service_thread_start

Variable Name	Value
dtheta	0
grid	0x7fffffffa7e8
grid->dx	1
grid->theta	0x74cdf0
	20
	1

Processes 2-3:
Process stopped in heatTimestep (heatC-MPI.c:157) with signal SIGSEGV (Segmentation fault).
Reason/Origin: address not mapped to object (attempt to access invalid address)
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.
 Always show this window for signals

Continue Pause

2 processes playing

Allinea DDT: Breakpoints (1)

The screenshot shows the Allinea DDT interface with a Fortran source code editor. A breakpoint is set on line 189. The 'DDT - Edit Breakpoint' dialog box is open, showing the location and condition 'y==4'. The dialog box has the following fields:

- Location: Line, File: /home/h9/hpclab70/Debugging/00/heatF-MPI.F90, Line Number: 189
- Applies To: Process Group: All, Process: All, Thread: All
- Hit Limits: Start on the n-th pass: 0, Trigger every n-th pass: 1, Stop after n hits: Never
- Condition: Condition: y==4
- Language: Auto

Callout bubbles provide instructions:

- Click to the margin left of the line number
- Or open context menu on source code line
- Edit breakpoint, e.g. to add condition
- Then hit run ...

Allinea DDT: Breakpoints (2)

Processes 0 and 2 stopped at conditional breakpoint

```
178
179 integer :: x, y, err
180 double precision :: dtheta,
181
182 mymax = 0.0d0
183
184 ! calculate the time step: read
185 ! Only calculate on a processes sub
186 do y=mympi%start_y,mympi%start_y +
187 do x=mympi%start_x,mympi%start_x
188
189 dtheta = ( grid%theta(x-1,y) + 2*grid%theta(x,y) + grid%theta(x+1,y) - 2*grid%theta(x,y) ) / (grid%
190 * ( grid%theta(x,y) + 2*grid%theta(x,y) + grid%theta(x+1,y) - 2*grid%theta(x,y) ) / (grid%
191
192
193
194
195
196
197
198
199
200
201
```

Process stopped at breakpoint in heatconduction::heattimestep (heatF-MPI.F90:189).

Always show this window for user-defined breakpoints

Processes	Threads	File	Line	Function	Condition	Start After	Trigger Ev
<input checked="" type="checkbox"/> All	all	heatF-MPI.F90	189	heatconduction::heattimestep	y==4	0	1

Variable Name | Value

dt	0.0500000000
dtheta	0
dthetamax	100
err	0
grid	
mymax	0
mympi	(rank = 0, ca
x	1
y	4

Type: none selected

Expression | Value

2 processes playing

Allinea DDT: Array Visualization

DDT - Multi-Dimensional Array Viewer

Array Expression: `(*grid).theta[$i][$j]` Evaluate Cancel

Distributed Array Dimensions: None [How do I view distributed arrays?](#)

Range of \$i: From: 0 To: 21 Display: Rows

Range of \$j: From: 0 To: 21 Display: Columns

Align Stack Frames Auto-update

Only show if: [See Examples](#)

Data Table Statistics

Goto Visualize Export Full Window

	j	11	12	13	14
i 0	0	0	0	0	0
1	0	0.88681159560070588	1.3048188290809211	1.4239999999999999	1.
2	5908699	1.7179608286283654	1.9112312334248853	1.952	1.
3	6283654	1.9804744622314747	1.9900310562001513	1.968	1.
4	4248853	1.9900310562001513	1.8932952390670059	1.8560000000000001	1.
5	1.952	1.968	1.8560000000000001		

Close

- Add To Evaluations
- Add Watchpoint
- Edit Type/Language...
- Copy Value
- View As
- View Array**
- Compare Across Processes
- Compare Across Threads
- View Pointer Details
- Find Variable In Files
- Show variables from control statements
- Sort Members Alphabetically

DDT - Visualization

File View Viewpoint

Close

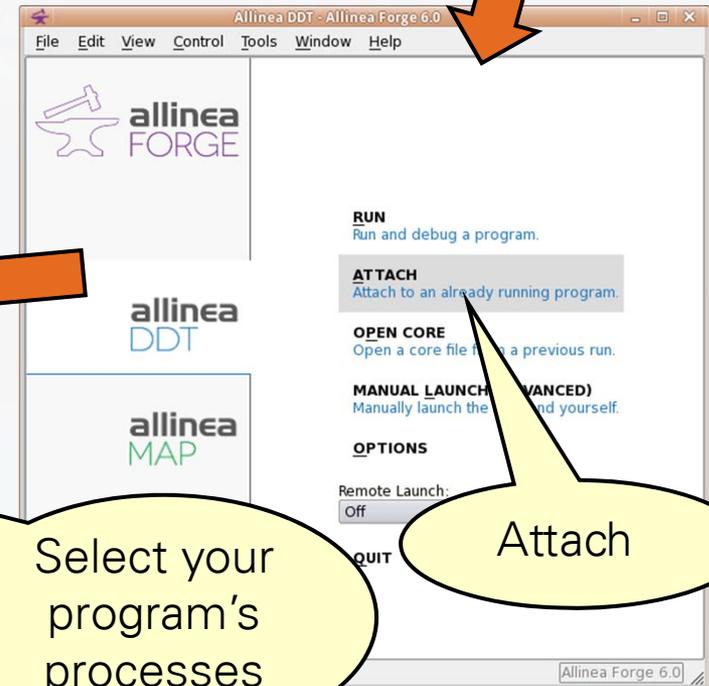
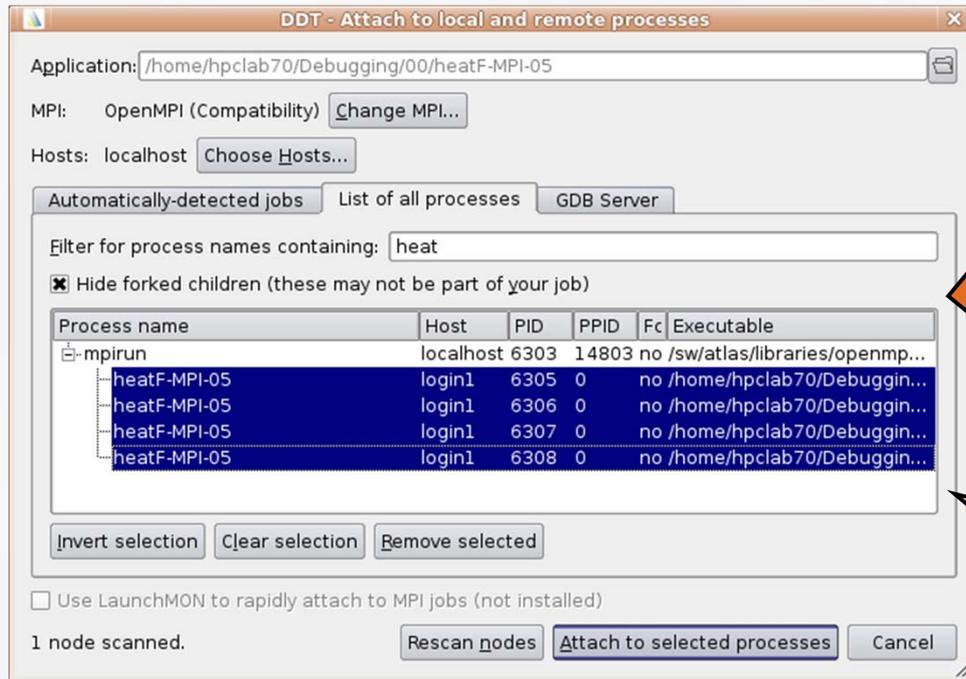
Allinea DDT: Attach to running program

```
% mpif90 -g heatF-MPI-05.F90 -o heatF-MPI-05
% mpirun -np 4 ./heatF-MPI-05
. . .
```

Programs runs – you want to know what it is doing?

Start DDT in a 2nd terminal:

```
% ddt
```



Select your program's processes

Attach

Allinea DDT: Core Files (1)

```
% mpif90 -g heatF-MPI-01.F90 -o heatF-MPI-01
% ulimit -c
1
% ulimit -Sc 100000
% export decfort_dump_flag=yes
% mpirun -np 2 ./heatF-MPI-01
. . .
```

Check core file size limit (reports kB) and increase if required (sets to 100 MB)

Intel Fortran only

Run program

Program crashes

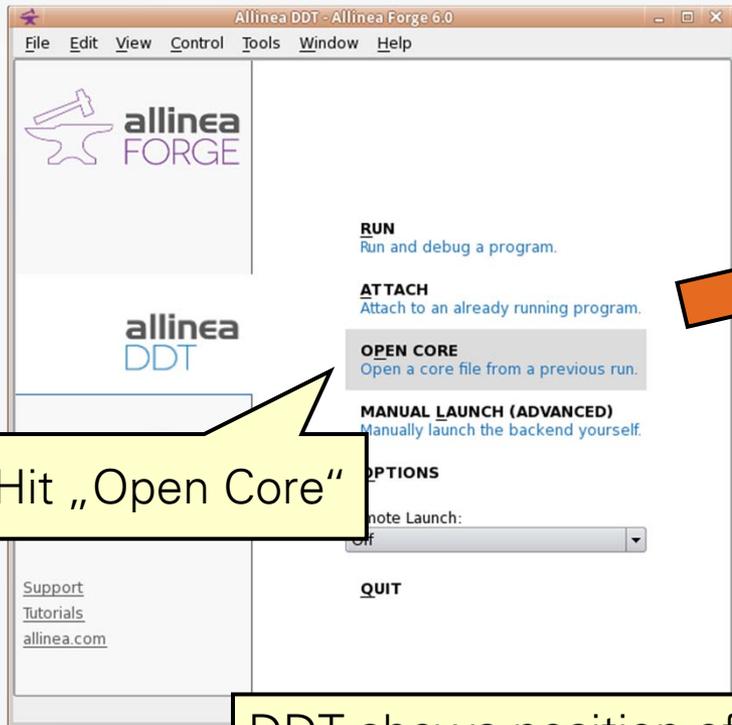
```
-----
mpirun noticed that process rank 0 with PID 27934 on node login1
exited on signal 11 (Segmentation fault).
-----
```

Corefiles created

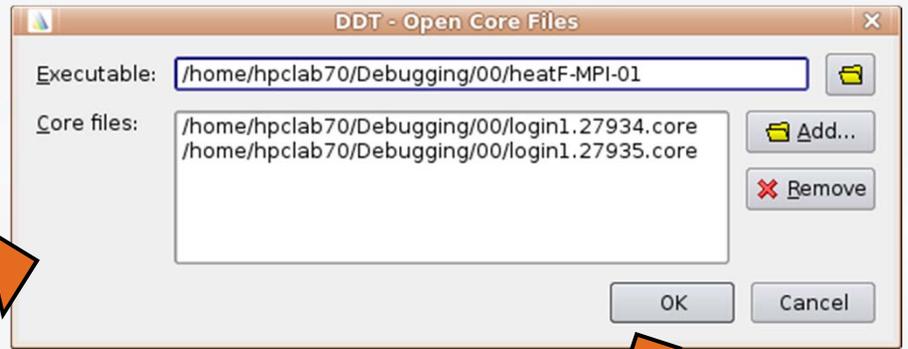
```
% ls -l *.core
-rw----- 1 hpclab70 zih-hpclab 76M 10. Feb 11:03 login1.27934.core
-rw----- 1 hpclab70 zih-hpclab 76M 10. Feb 11:03 login1.27935.core
% ddt
```

Analyze with DDT

Allinea DDT: Core Files (2)



Hit „Open Core“



DDT shows position of crash in source code
DDT shows variables at the time of the crash
But no stepping possible!

