

Matthias Lieber

Center for Information Services and High Performance Computing (ZIH)

# Parallel Debugging with DDT

Parallel Programming with MPI, OpenMP, and Tools  
Dresden, 8-12 February 2021

# Why using a Debugger?

Your program shows incomprehensible behavior, e.g.

- Program terminates abnormally

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

- Program produces wrong results

```
% ./myprog  
Pi = 3.573
```

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

# 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 program execution:

- Stop the program at a specific source code line (**Breakpoints**)
- Stop the program when certain conditions are true (**Conditional Breakpoints** and Watchpoints)
- Stop the program before terminating abnormally
- Execute the program line-by-line (**Stepping**)

# Typical Usage of a Debugger

## Development workflow

- Compile the program with **-g**

```
mpif90 -g myprog.c -o myprog
```

- Run the program under control of the debugger

```
ddt <mpirun command> ./myprog
```

- Use the Debugger to locate the position of the problem and examine variables
- Understand the cause of the problem and correct the source code
- Repeat until problem is solved



### Hints:

Always compile your application with the **-g flag**, especially during developing and testing. It adds **symbolic debug info** to the binary and has no performance impact.

**Optimizations often interfere** with debugging (e.g. functions or variables of interest are “optimized away”). If necessary, compile with the **-O0 flag** to disable optimizations.

# 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 or you do not expect problems

## 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 file
- 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)

# Arm DDT

## Distributed Debugging Tool

- Commercial debugging tool by Arm (Arm acquired Allinea in 2016)
- Languages: C, C++, Fortran
- Parallel Support: Pthreads, OpenMP, MPI, PGAS languages, CUDA, OpenACC
- Available for all common HPC platforms
- Intuitive graphical user interface



- More info:  
<https://developer.arm.com/tools-and-software/server-and-hpc/debug-and-profile/arm-forge/arm-ddt>

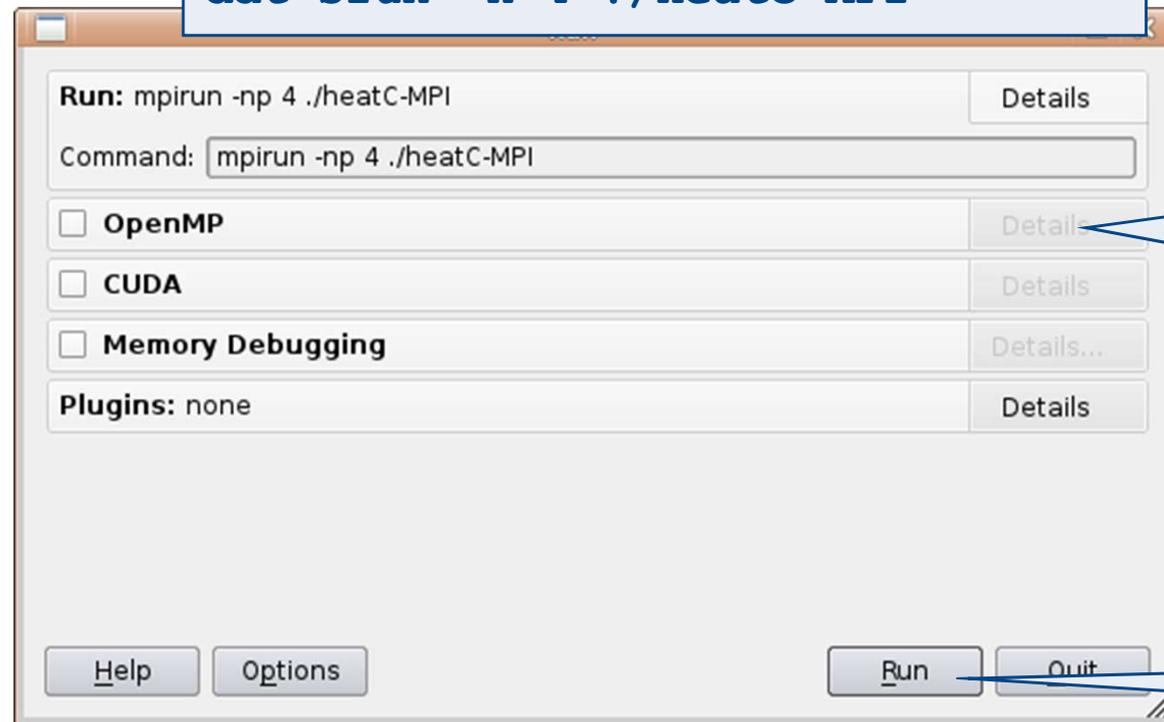
# DDT: Program Start

```
% mpicc -g -O0 heatC-MPI.c -o heatC-MPI  
% ddt mpirun -np 4 ./heatC-MPI
```

Compile with  
Debugging

On Taurus we use srun instead of mpirun:  
`ddt srun -n 4 ./heatC-MPI`

Start DDT: prepend **ddt**  
to mpirun command line



Enable/disable OpenMP  
and set number of  
threads if necessary

Start Program

# DDT: Main Window

The screenshot shows the DDT main window with several callouts pointing to specific features:

- Process and thread selection:** Points to the top toolbar and the 'Process' radio button in the 'Focus on current' section.
- Process control: run, stop, stepping:** Points to the 'Run', 'Stop', and 'Step' buttons in the top toolbar.
- Source file browser:** Points to the 'Project Files' pane on the left side.
- Source view:** Points to the central code editor showing the source code of 'heatC-MPI.c'.
- Variables pane:** Points to the 'Locals' pane on the right, which displays a table of local variables.
- Output, Breakpoints, Watchpoints, Call stack:** Points to the bottom toolbar and the 'Stacks (All)' pane.
- Evaluation window:** Points to the 'Evaluate' pane at the bottom right.

The 'Locals' pane shows the following variables:

Variable Name	Value
argc	1
argv	0x7fffffffd98

The 'Stacks (All)' pane shows the following call stack:

Processes	Function
4	main (heatC-MPI.c:429)

# DDT: Process Control & Stepping

The screenshot displays the Arm DDT - Arm Forge 18.2.2 interface. The top menu bar includes File, Edit, View, Control, Tools, Window, and Help. Below the menu is a toolbar with various icons for debugging, including Run, Pause, Step to next code line, Step over function calls, Step out of current function, and a right mouse button icon. A status bar below the toolbar shows 'Current Group: All' and 'Focus on current: Group'. Below the status bar is a group selection area with buttons for '0', '1', '2', and '3'. The main area shows a source code editor with a file named 'MPI.c'. The code includes a main function with a loop. A right mouse button is shown clicking on a line of code, with a callout indicating 'Right mouse button at source code line -> „Run to here“'. Other callouts point to the Run button, Pause button, Step to next code line button, Step over function calls button, Step out of current function button, and the group selection buttons.

Pause

Run

Step to next code line

Step over function calls

Step out of current function

Select group / processes

Commands may affect whole group or single processes / threads

Right mouse button at source code line -> „Run to here“

```
*****  
* Main program and time stepping loop.  
*****  
int main (int argc, char** argv)  
{  
    heatGrid mygrid;  
    double dt, dthetamax, ener  
    t step, nsteps=20;  
    taMPI mympi;
```

# DDT: Segmentation Fault

Processes 2 and 3 crashed

Line where the program crashed is highlighted

Segmentation Fault!

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->num_cells_x;x++)  
154 {  
155     for (y=mympi->start_y; y < mympi->start_y + mympi->num_cells_y; y++)  
156     {  
157         dtheta = ( grid->theta[x-1][y] + grid->theta[x+2][y] - 2*grid->theta[x][y]  
158                 + ( grid->theta[x][y-1] + grid->theta[x][y+1] - 2*grid->theta[x][y])  
159                 grid->thetanew[x][y] = grid->theta[x][y] + grid->k * dtheta * dt;  
160     }  
161     mymax = fmax(fabs(dtheta), mymax); /* save max theta for the next condition
```

Process 2 crashed in `heatTimestep (heatC-MPI.c:157)`

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

# DDT: Breakpoints (1/2)

The screenshot shows the Alinea DDT v3.2.1-27702 interface. The main window displays a source code editor for 'heatF-MPI.F90'. A breakpoint is set at line 189. The 'DDT - Edit Breakpoint' dialog box is open, showing the following settings:

- 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 boxes 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 ...

ds	File	Line	Function	Condition	Start After
	heatF-MPI.F90	189	heatconduction::heattimestep	y==4	0

# DDT: Breakpoints (2/2)

The screenshot shows the Allinea DDT v3.2.1-27702 interface. The main window displays a Fortran source file named 'heatF-MPI.F90'. A conditional breakpoint is set at line 189, which contains the following code:

```
189 dtheta = ( grid%theta(x-1,y) + grid%theta(x+1,y) - 2*grid%theta(x,y) ) / (grid%  
190 + ( grid%theta(x,y-1) + grid%theta(x,y+1) - 2*grid%theta(x,y) ) / (grid%  
191 grid%thetaneew(x,y) = grid%theta(x,y) + grid%k * dtheta * dt
```

A dialog box titled 'Allinea DDT' is displayed, indicating that processes 0 and 2 have stopped at the breakpoint in 'heatconduction::heattimestep' at line 189. The dialog includes a checkbox for 'Always show this window for user-defined breakpoints' and buttons for 'Continue' and 'Pause'.

The 'Locals' window shows the following variable values:

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

At the bottom of the interface, a status bar indicates '2 processes playing'.

Processes 0 and 2  
stopped at conditional  
breakpoint

# DDT Practical 1: Conditional Breakpoints

C:

```
% cd ~/Debugging/c  
% mpicc -g -O0 heatC-MPI.c -o heatC-MPI  
% ddt srun -n 4 ./heatC-MPI
```

Fortran 90:

```
% cd ~/Debugging/f90  
% mpif90 -g -O0 heatF-MPI.F90 -o heatF-MPI  
% ddt srun -n 4 ./heatF-MPI
```

In the DDT run window:  
uncheck OpenMP, CUDA,  
Mem. debugging and hit run

Task A:

- Find out the value of `dthetamax` after step 10 has been computed.
- Hint: Use a conditional breakpoint in the time stepping loop (main program)

Task B (optional):

- Which process contributed the maximum to `dthetamax` at the `MPI_Allreduce` in `heatTimestep` after step 10 has been computed?
- Hint: use an additional breakpoint at the `MPI_Allreduce`, then right click on the variable `mymax` in the variables pane and select “Compare Across Processes”

# DDT Practical 1: Task A Solution

Breakpoint in time stepping loop, condition: `step==11`

`dthetamax = 0,512`

```
447
448     /* energy of initial grid */
449     heatTotalEnergy(&mygrid, &energyInitial);
450 }
451
452 /* time stepping loop */
453 for( step=1 ; step<=nsteps ; step++)
454 {
455     heatBoundary(&mygrid, &mympi);
456     heatTimestep(&mygrid, &mympi, dt, &dthetamax );
457 }
458
459 /* Gather data on process 0 for output*/
460 heatMPIGather (&mygrid, &mympi);
461
462 /* Work only for master process*/
463 if (mympi.rank == 0)
464 {
465     /* output of final grid */
466     heatOutputFinal(&mygrid, &mympi);
467 }
```

Variable Name	Value
argc	1
argv	0x0000000000000000
dt	0.5000000000000000
dthetamax	0.51246656139334856
energyFinal	0
energyInitial	113.4835937630376
mygrid	{theta = 0x74cde0, th...
mympi	{rank = 0, cart = 0x74...
nsteps	20
step	11

Processes	Threads	File	Line	Function	Condition	Start After	Trigger E
All	all	heatC-MPI.c	455	main	step==11	0	

# DDT Practical 1: Task B Solution

Not possible with a single breakpoint: variable step is local in main, variable mymax is local in heatTimestep

3: Open context menu for mymax, select „Compare Across Processes“

4: Max. value at processes 0 and 1

1: Run to breakpoint in time stepping loop, condition: step==10

2: Then add breakpoint at MPI\_Allreduce in heatTimestep

DDT - Cross-Process Comparison View

Expression: mymax

Processes in current group (All, 4 procs)

Limit comparison to 1 s.f.

Only show if:  [See Examples](#)

Align stack frames

Compare Cancel

Use as MPI Rank Create Groups Export Full Window

Values	Process(es)
0.27774768663151839	2
0.38032673944589612	3
0.51246656139334856	0-1

Statistics

Count: 4  
 Not shown: 0  
 Errors: 0  
 Aggregate: 0  
 Numerical: 4  
 Sum: 1.68301  
 Minimum: 0.277748  
 Maximum: 0.512467  
 Range: 0.234719

Locals

- dt
- dtheta
- dthetamax
- grid
- mymax** -0.512466561393
- mympi
- x
- y

Type: double

Expression

- 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
- Sort Members Alphabetically

Processes	Threads	File	Line	Function	Condition	Start After	Trigger E
<input checked="" type="checkbox"/> All	all	heatC-MPI.c	455	main	step==10	0	
<input checked="" type="checkbox"/> All	all	heatC-MPI.c	166	heatTimestep		0	

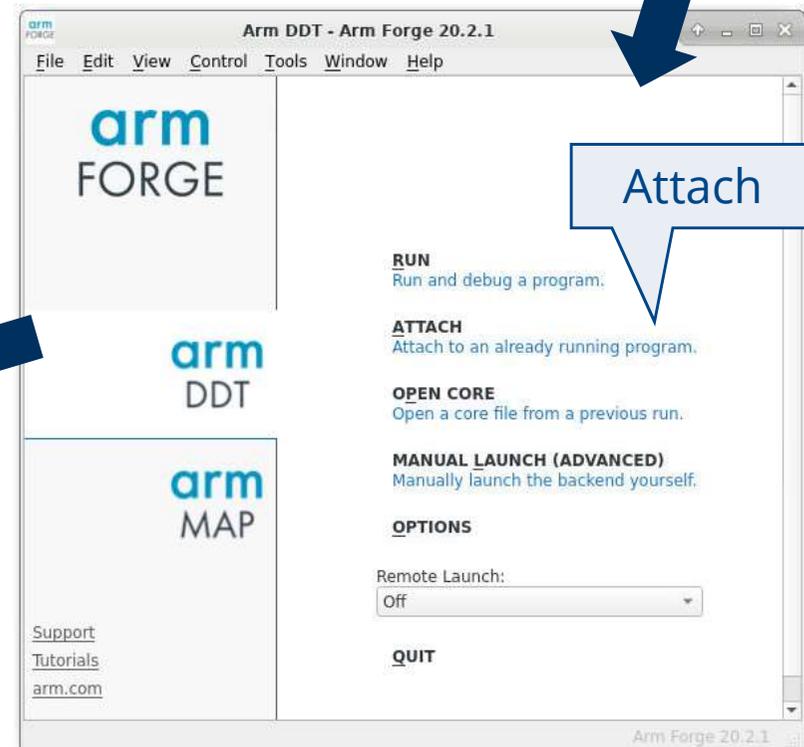
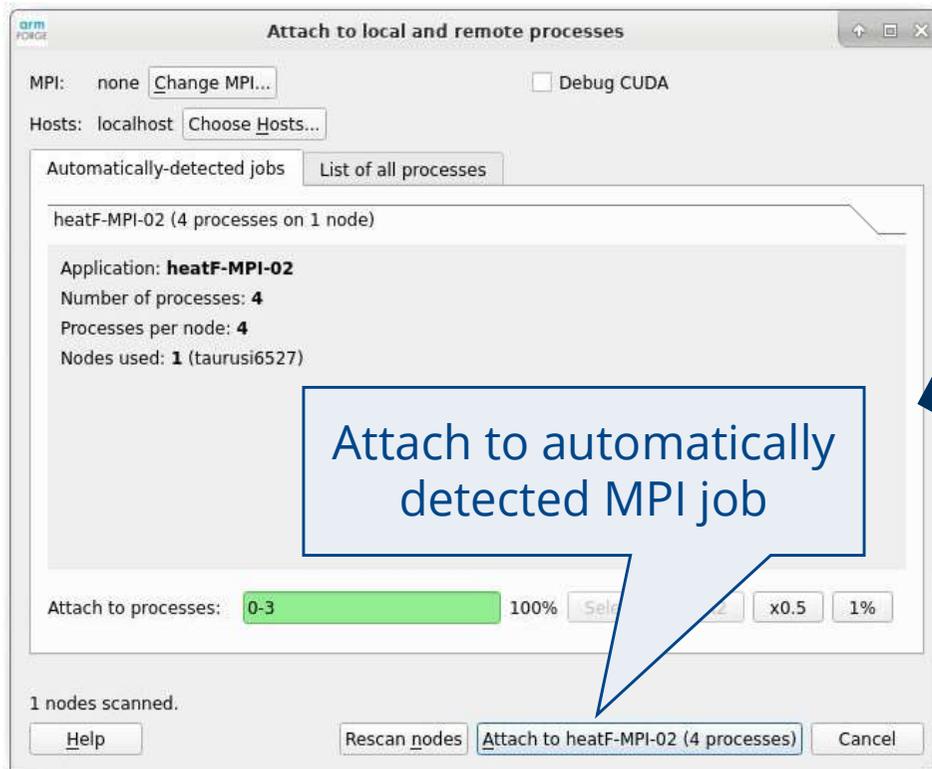
# DDT: Attach to running program

```
% mpif90 -g heatF-MPI-02.F90 -o heatF-MPI-02
% srun -n 4 ./heatF-MPI-02
. . .
```

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

Start DDT in a 2nd terminal

```
% ddt
```



# DDT: Core Files (1/2)

```
% mpif90 -g -O0 heatF-MPI-01.F90 -o heatF-MPI-01
% ulimit -c
0
% ulimit -Sc 100000
% export FOR_DUMP_CORE_FILE=yes
% srun -n 2 ./heatF-MPI-01
```

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

Intel Fortran only

Segmentation Fault

```
forrtl: severe (174): SIGSEGV, segmentation fault occurred
```

srun realizes crash

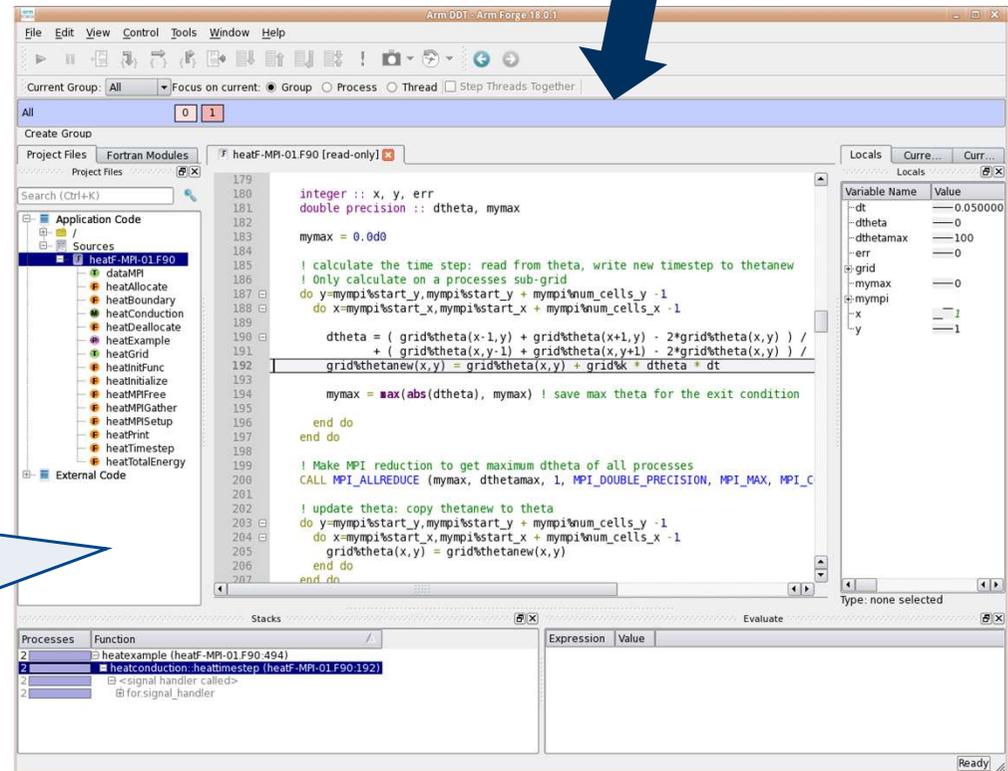
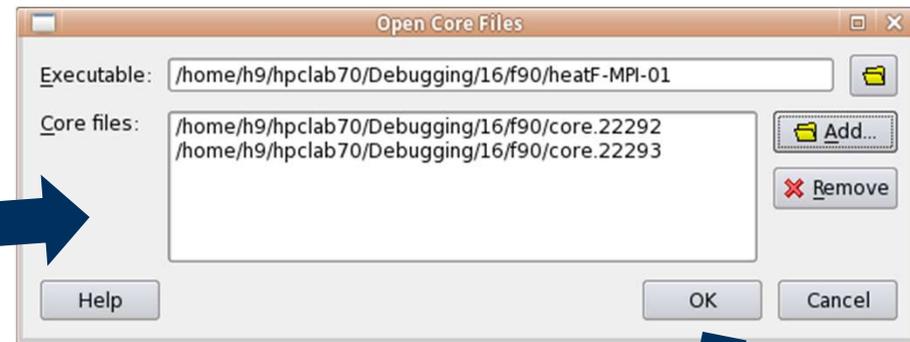
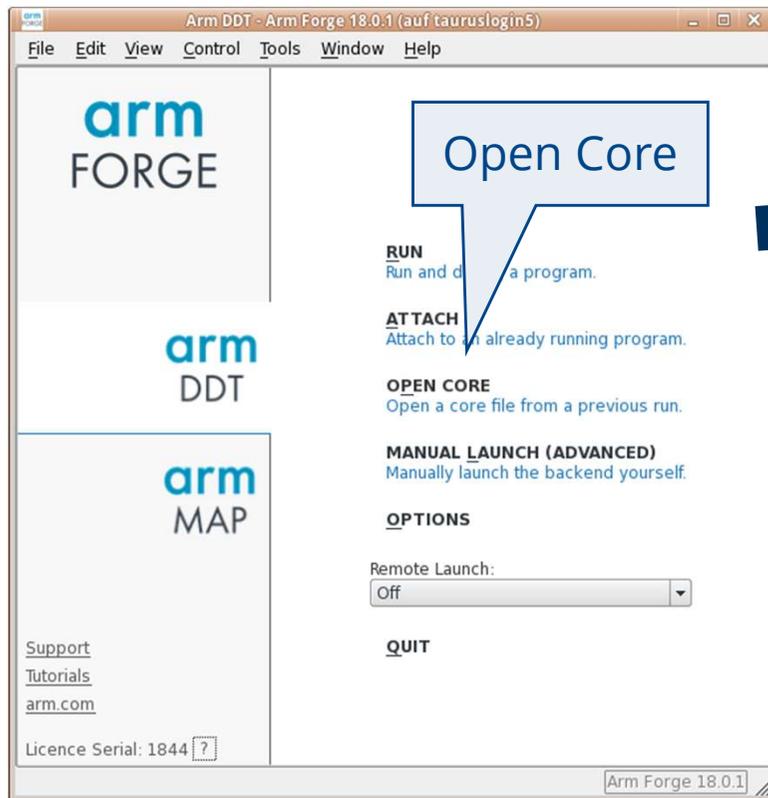
```
srun: error: taurusi6595: tasks 0-1: Exited with exit code 174
```

Per-process core files

```
% ls -lh core*
-rw----- 1 gpu59 1111111 42M Jan 27 13:21 core.taurusi6595.taurus.hrsk.tu-dresden.de.3371
-rw----- 1 gpu59 1111111 42M Jan 27 13:21 core.taurusi6595.taurus.hrsk.tu-dresden.de.3372
% ddt
```

Analyze with DDT

# DDT: Core Files (2/2)



DDT shows position of the crash in the source code and values of variables at the time of the crash. But no running or stepping possible!

# DDT: Multidimensional Array Visualization

Set ranges and "Evaluate"

"Visualize"

Right mouse click on array variable

DDT - Multi-Dimensional Array Viewer

Array Expression: `{*(grid)).theta[$i][$j]}`

Distributed Array Dimensions: None

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

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

Buttons: Evaluate, Cancel, Align Stack Frames, Auto-update

Buttons: Goto, Visualize, Export, Full Window

i	j	11	12	13	14
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.

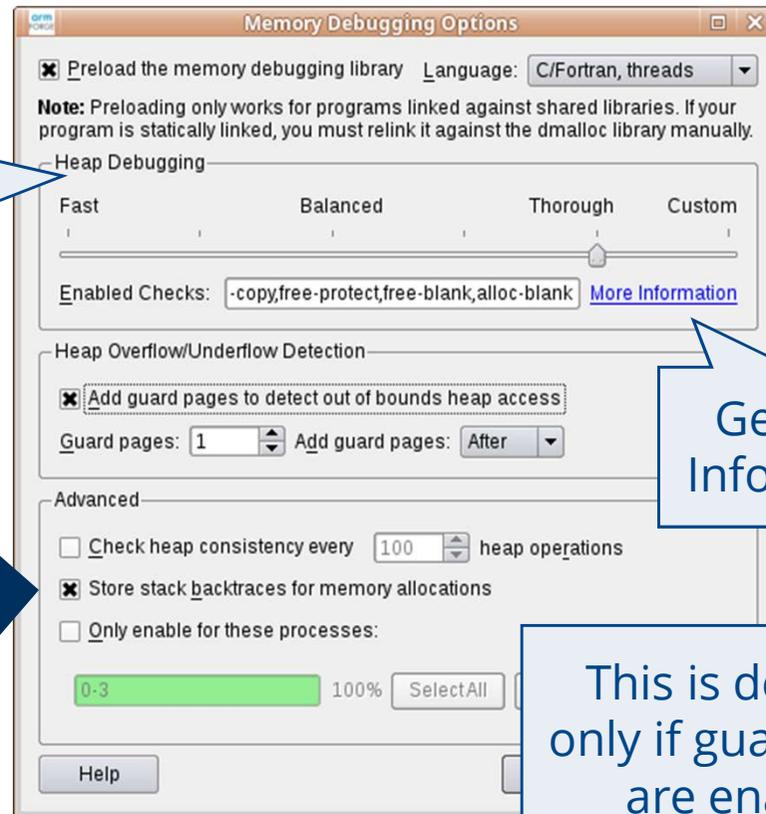
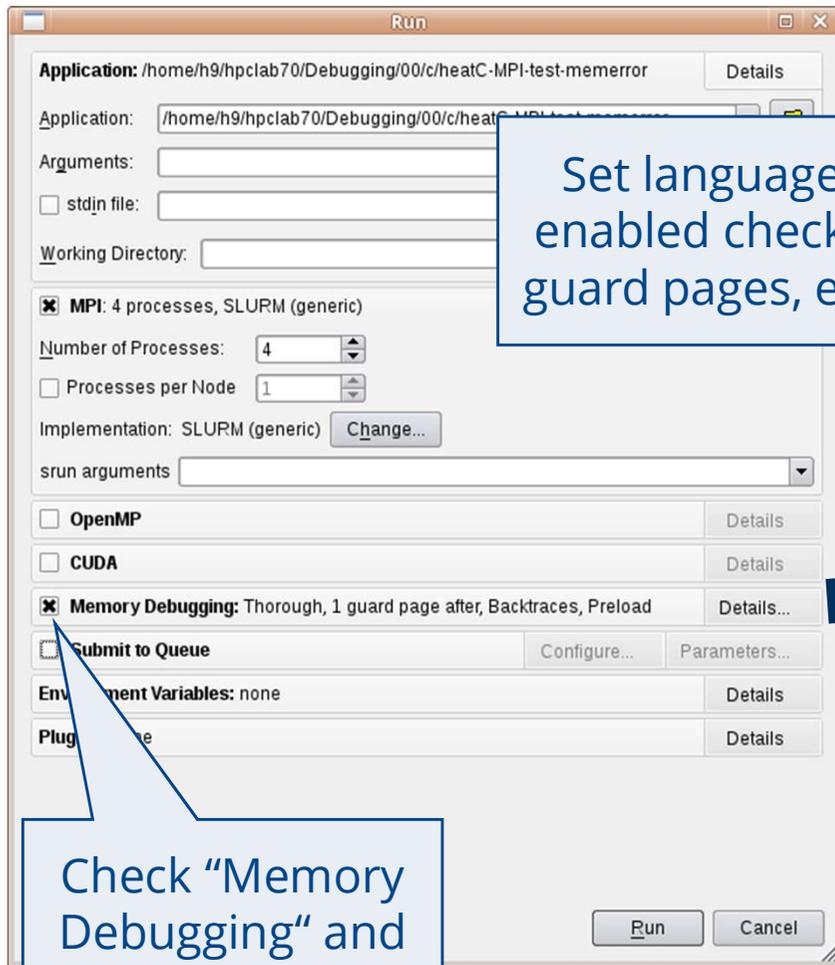
DDT - Visualization

3D visualization of a multidimensional array with axes labeled 'row', 'col', and 'Process 2'.

Context Menu:

- 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: Memory Debugging



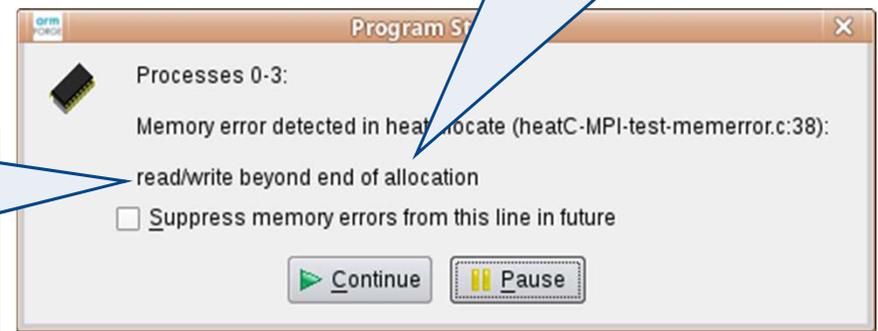
Check "Memory Debugging" and click "Details..."

Set language, enabled checks, guard pages, etc.

Get more Information

This is detected only if guard pages are enabled

Message when memory error is detected, pause to inspect the error



# DDT Practical 2: Find the Bugs!

Find the bug in each of the three programs!

- Compile and first run normally (4 processes) to observe the behavior, then use DDT to find the bug
- If the program stops in MPI, DDT may complain about missing source files: ignore and select an application function in the call stack view

## **heatC-MPI-01 / heatF-MPI-01**

- Produce core dumps (with up to 4 MPI processes) and open with DDT
- You already know this example from the Intro's practical

## **heatC-MPI-02 / heatF-MPI-02**

- Run without DDT and then attach DDT (use a second terminal session)
- In case of trouble when attaching: ensure that Options - System - Debugger is set to GNU 7.6.2

## **heatC-MPI-03 / heatF-MPI-03**

- Hint: compare arguments of send and receive call

Optional OpenMP task on next slide

# DDT Practical 3: Breakpoints with OpenMP (optional)

C:

```
% cd ~/Debugging/c
% icc -g -O0 -fopenmp heatC-omp.c -o heatC-omp
% ddt ./heatC-omp
```

Fortran 90:

```
% cd ~/Debugging/f90
% ifort -g -O0 -fopenmp heatF-omp.F90 -o heatF-omp
% ddt ./heatF-omp
```

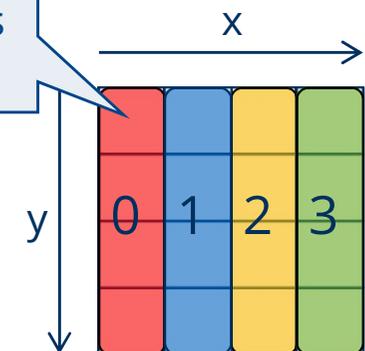
Task:

- Run with 4 threads in DDT and find out which thread computes which part of the 20 x 20 grid

Hints:

- Use a breakpoint in the inner compute loop in heatTimestep and examine loop variable  $x$ , use condition  $y==1$  to jump to the next iteration of the  $x$ -loop
- Fortran:  $x$  and  $y$  are interchanged compared to C,  $y$ -loop is parallelized: need to examine  $y$
- Breakpoints in OpenMP parallel regions sometimes behave unexpected - it helps to run a single thread, while pausing all others: select "Focus on current thread" and select the thread below

C loop order,  
Fortran order is  
interchanged!



# DDT Practical 2 Task 01: Solution (C)

Segmentation fault writing grid->thetanew[0] in heatAllocate  
Check if the array has been allocated

```
25
26 grid->xsize = xsize;
27 grid->ysize = ysize;
28 grid->theta = (double**) malloc (sizeof(double*)*(ysize+2));
29 grid->theta [0] = (double*) malloc (sizeof(double)*(xsize+2)*(ysize+2))
30
31 for (i = 0; i < xsize+2; i++)
32 {
33     grid->theta [i] = grid->theta [0]+i*(ysize+2);
34     grid->thetanew[i] = grid->thetanew[0]+i*(ysize+2);
35
36     for (j = 0; j < ysize+2; j++)
37     {
38         grid->theta [i][j] = 0.0;
39         grid->thetanew[i][j] = 0.0;
40     }
41 }
42
43 grid->dx = 1.0;
44 grid->dy = 1.0;
45 grid->sk = 1.0;
```

Variable Name	Value
grid	0x7fffffff7858
*(grid)	{theta = 0x6164b0, thetanew = 0x6164b0}
theta	0x6164b0
thetanew	0x0
*(thetanew)	<Cannot access memory at add
xsize	20
ysize	20
dx	6.9533558061358138e-310
dy	2.079947693866876e-317
k	1.2414507464325953e-312
i	0
j	1
xsize	20
ysize	20

Processes	Function
4	main (heatC-MPI-01.c:432)
4	heatAllocate (heatC-MPI-01.c:34)

# DDT Practical 2 Task 01: Solution (Fortran)

Segmentation fault writing  
grid%thetaneu(1,1) in heatTimestep  
Check if the array has been allocated

```
182  
183 mymax = 0.0d0  
184  
185 ! calculate the time step: read from theta, write to thetaneu  
186 ! Only calculate on a processes sub-grid  
187 do y=mympi%start_y, mympi%start_y + mympi%num_cells_y - 1  
188 do x=mympi%start_x, mympi%start_x + mympi%num_cells_x - 1  
189  
190 dtheta = ( grid%theta(x-1,y) + grid%theta(x+1,y) - 2*grid%theta(x,y) ) / (grid%  
191 + ( grid%theta(x,y-1) + grid%theta(x,y+1) - 2*grid%theta(x,y) ) / (grid%  
192 grid%thetaneu(x,y) = grid%theta(x,y) + grid%k * dtheta * dt  
193  
194 mymax = max(abs(dtheta), mymax) ! save max theta for the exit condition  
195  
196 end do  
197 end do  
198  
199 ! Make MPI reduction to get maximum dtheta of all processes  
200 CALL MPI_ALLREDUCE (mymax, dthetamax, 1, MPI_DOUBLE_PRECISION, MPI_MAX, MPI_COMM_W  
201  
202 ! update theta: copy thetaneu to theta  
203 do y=mympi%start_y, mympi%start_y + mympi%num_cells_y - 1
```

Variable Name	Value
dt	0.0500000000000000
dtheta	0
dthetamax	100
err	0
grid	
theta	
thetaneu	<not associated>
xsize	20
ysize	20
dx	1
dy	1
k	1
mymax	0
mympi	
x	1
y	1

Processes Function

4	heatexample (heatF-MPI-01.F90:494)
4	heatconduction::heattimestep (heatF-MPI-01.F90:192)

Fortran version  
crashes later than  
C version

# DDT Practical 2 Task 02: Solution

Arm DDT - Arm Forge 18.2.2

File Edit View Control Tools Window Help

Current Group: All Focus on current:  Group  Process  Thread  Step Threads Together

All 0 1 2 3

Create Group

Proj... Fortran ...

Project Files

Search (Ctrl+K)

Application Code

Headers

Sources

heatC-MPI-02.c

```
249 /*Send right column to right neighbor*/
250 MPI_Bsend (&(grid->theta[mympi->start_x+mympi->num_cells_x-1][mympi->start_y]),
251           mympi->num_cells_y, MPI_DOUBLE, mympi->right, 123, mympi->cart);
252 /*Receive left column from left neighbor*/
253 MPI_Recv (&(grid->theta[mympi->start_x-1][mympi->start_y]),
254          mympi->num_cells_y, MPI_DOUBLE, mympi->left, 123, mympi->cart, &status);
255
256 /*Send upper row to top neighbor*/
257 MPI_Bsend (&(grid->theta[mympi->start_x][mympi->start_y]),
258           1, mympi->rowtype, mympi->up, 123, mympi->cart);
259 /*Receive lower border row from bottom neighbor*/
260 MPI_Recv (&(grid->theta[mympi->start_x][mympi->start_y+mympi->num_cells_y]),
261          1, mympi->rowtype, mympi->down, 122, mympi->cart, &status);
262
263 /*Send lower row to bottom neighbor*/
264 MPI_Bsend (&(grid->theta[mympi->start_x][mympi->start_y+mympi->num_cells_y-1]),
265           1, mympi->rowtype, mympi->down, 123, mympi->cart);
266 /*Receive upper border row from top neighbor*/
267 MPI_Recv (&(grid->theta[mympi->start_x][mympi->start_y-1]),
268          1, mympi->rowtype, mympi->up, 123, mympi->cart, &status);
269 }
270
```

Locals Current Line(s) Current Stack

Variable Name	Value
grid	0x7ffffff2a68
grid->theta	0x618660
mympi	0x7ffffff2a08
mympi->num_cells_y	10
mympi->start_x	1
mympi->start_y	1

Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook

Stacks (All)

Processes	Function
4	main (heatC-MPI-02.c:457)
4	heatBoundary (heatC-MPI-02.c:260)
4	PMPI_Recv (recv.c:173)

Ready

All processes are waiting at this MPI\_Recv in heatBoundary Reason: Tags are not matching

# DDT Practical 2 Task 03: Solution

Arm DDT - Arm Forge 18.2.2

File Edit View Control Tools Window Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

All 0 1 2 3

Create Group

Proj... Fortran ...

Project Files

Search (Ctrl+K)

heatC-MPI-03.c

```
386 MPI_DOUBLE, /* old type */
387 &blocktype /* new type */);
388 MPI_Type_commit (&blocktype);
389
390 MPI_Send (block_size, 4, MPI_DOUBLE, 0, 50, MPI_COMM_WORLD);
391 MPI_Send (&grid->theta[mympi->start_x][mympi->start_y], 1, blocktype, 0, 51, MPI_COMM_WORLD);
392
393 MPI_Type_free (&blocktype);
394 }
395 else
396 /*Master Receives data*/
397 {
398 MPI_Comm_size (MPI_COMM_WORLD, &size);
399 for (i = 1; i < size; i++)
400 {
401 /*Receive Block Info*/
402 MPI_Recv (block_size, 4, MPI_INT, 1, 50, MPI_COMM_WORLD, &status);
403
404 /* Create datatype to communicate one block*/
405 MPI_Type_vector (
406
407
```

Variable Name Value

block_size	
i	1
status	{count_lo = 1

Program Stopped

Process 0:

Program stopped at MPID\_Abort.

Always show this window for default breakpoints

Continue Pause

Caution: If ranks 1-3 would send 4 MPI\_FLOAT, MPI would not abort because the buffer size fits! Only MUST could detect this error.

# DDT Practical 3: Solution

1: Run to a breakpoint in OpenMP parallel for loop

```
151 /* calculate the time step: ... eta, write new timestep to th
152 /* OpenMP 3.1: New reduction ... min and max were added for C a
153 #pragma omp parallel for private(c ... , y) reduction(max:dthetamax_re
154 for (x=1; x <= grid->xsize;x++
155 {
156     for (y=1; y <= grid->ysize; y++)
157     {
158         dtheta = ( grid->theta[x-1][y] + grid->theta[x+1][y] - 2*grid
159             + ( grid->theta[x][y-1] + grid->theta[x][y+1] - 2*grid
160 grid->thetaneu[x][y] = grid->theta[x][y] + grid->k * dtheta *
161
162         dthetamax_reduction = fmax(fabs(dtheta), dthetamax_reduction)
```

3: Compare x across threads (via context menu)

Thread 0 computes x=1, ..., x=5  
Thread 1 computes x=6, ..., x=10  
etc.

2: Select bottom call stacks

Values	Thread(s)	OpenMP thread(s)
1	0	0
6	1	1
11	2	2
16	3	3