



arm

Debugging with Arm DDT

ALCF Computational Performance
Workshop
May 6, 2020

Ryan Hulguin
ryan.hulguin@arm.com

Agenda

- Arm Software
- Debugging with DDT
- Theta Specific Settings

Arm Software

Arm Forge

An interoperable toolkit for debugging and profiling



The de-facto standard for HPC development

- Available on the vast majority of the Top500 machines in the world
- Fully supported by Arm on x86, IBM Power, Nvidia GPUs, etc.



State-of-the art debugging and profiling capabilities

- Powerful and in-depth error detection mechanisms (including memory debugging)
- Sampling-based profiler to identify and understand bottlenecks
- Available at any scale (from serial to parallel applications running at petascale)



Easy to use by everyone

- Unique capabilities to simplify remote interactive sessions
- Innovative approach to present quintessential information to users

Run and ensure application correctness

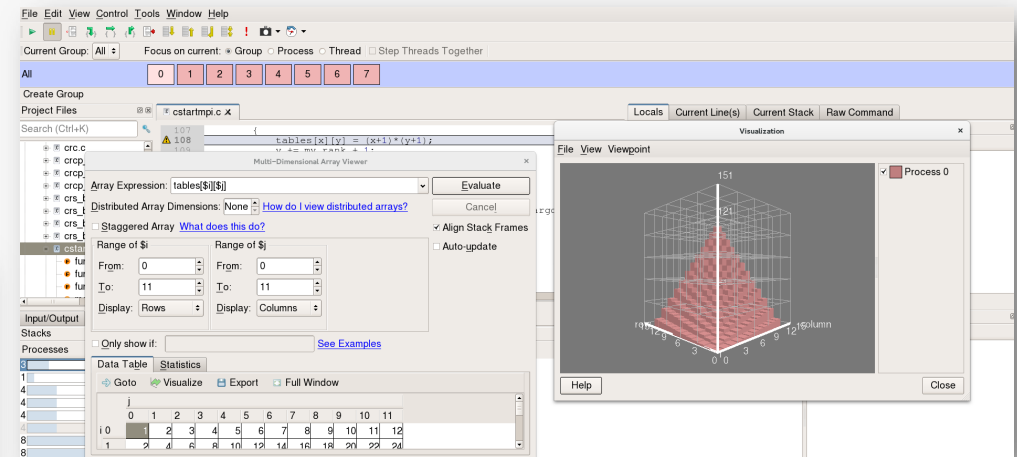
Combination of debugging and re-compilation

- Ensure application correctness with
- Integrate with continuous integration system.
- Use version control to track changes and leverage Forge's built-in VCS support.

Examples:

```
$> ddt --offline aprun -n 48 ./example  
$> ddt --connect aprun -n 48 ./example
```

15	▶	2:17.256	0-7	Play
16	●	2:18.048	4-7	Process stopped at breakpoint in main (cpi.c:50).
17				Additional Information Stacks Processes: Function 4-7 main (cpi.c:50)
18		2:19.048	n/a	Select process 4
19				Additional Information Current Stack Locals
9		2:17.832	main (cpi.c:46) 0-7	done: 0 i: / from 65 to 72 numprocs: 8 myid: / from 0 to 7 n: 100
10		2:17.832	main (cpi.c:46) 0-7	done: 0 i: / from 73 to 80 numprocs: 8 myid: / from 0 to 7 n: 100
11		2:18.323	main (cpi.c:46) 0-7	done: 0 i: / from 81 to 88 numprocs: 8 myid: / from 0 to 7 n: 100
12		2:18.323	main (cpi.c:46) 0-7	done: 0 i: / from 89 to 96 numprocs: 8 myid: / from 0 to 7 n: 100
13		2:18.325	main (cpi.c:46) 0-3	done: 0 i: / from 97 to 100 numprocs: 8 myid: / from 0 to 3 n: 100



Debugging with DDT

Arm DDT – The Debugger

Who had a rogue behaviour ?

– Merges stacks from processes and threads

Where did it happen?

– leaps to source

How did it happen?

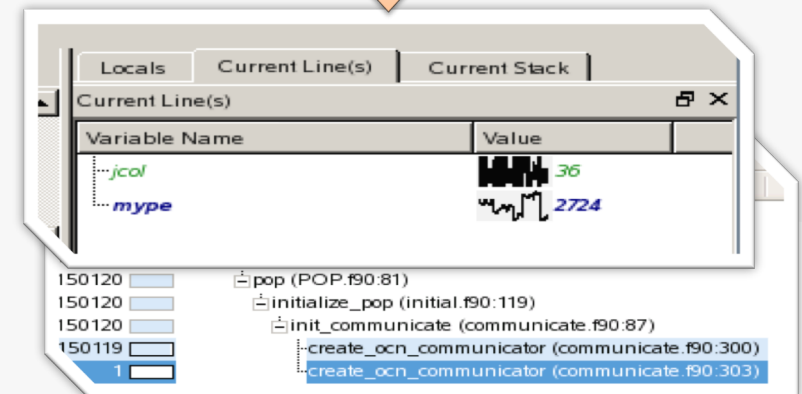
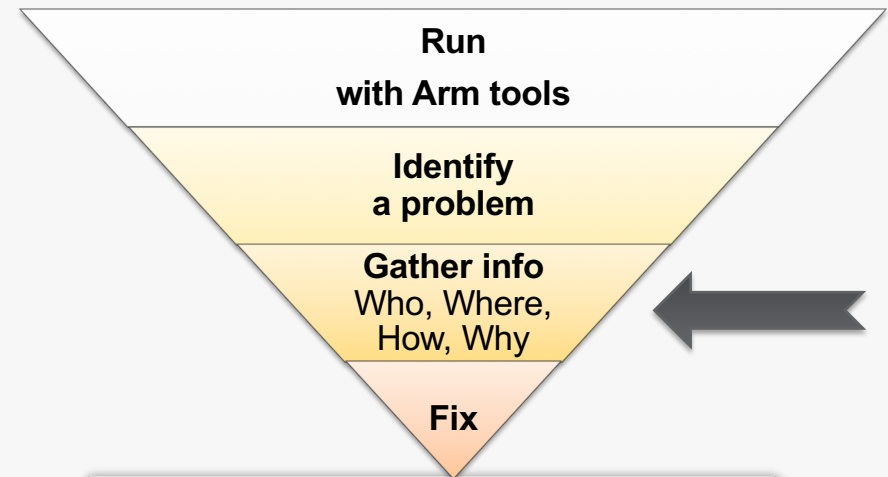
– Diagnostic messages

– Some faults evident instantly from source

Why did it happen?

– Unique “Smart Highlighting”

– Sparklines comparing data across processes



Preparing Code for Use with DDT

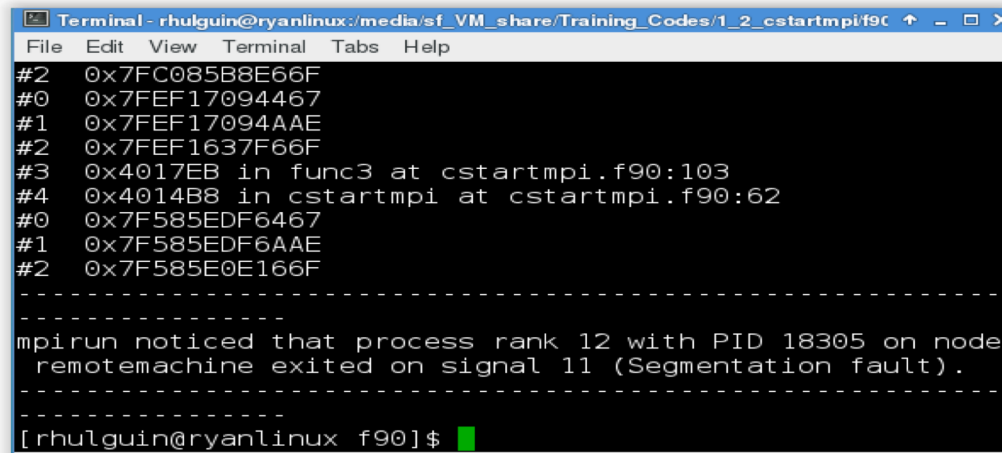
As with any debugger, code must be compiled with the debug flag typically `-g`

It is recommended to turn off optimization flags i.e. `-O0`

Leaving optimizations turned on can cause the compiler to *optimize out* some variables and even functions making it more difficult to debug

Segmentation Fault

In this example, the application crashes with a segmentation error outside of DDT.



```
Terminal - rhulguin@ryanlinux:/media/sf_VM_share/Training_Codes/1_2_cstartmpi.f90
File Edit View Terminal Tabs Help
#2 0x7FC085B8E66F
#0 0x7FEF17094467
#1 0x7FEF17094AAE
#2 0x7FEF1637F66F
#3 0x4017EB in func3 at cstartmpi.f90:103
#4 0x4014B8 in cstartmpi at cstartmpi.f90:62
#0 0x7F585EDF6467
#1 0x7F585EDF6AAE
#2 0x7F585E0E166F
-----
mpirun noticed that process rank 12 with PID 18305 on node
remotemachine exited on signal 11 (Segmentation fault).
-----
[rhulguin@ryanlinux f90]$
```

What happens when it runs under DDT?

Segmentation Fault in DDT

The screenshot shows the DDT debugger interface. The main window displays the source code of a Fortran program named 'cstartmpi.f90'. The code is as follows:

```
95
96  subroutine func3(my_rank)
97    integer, allocatable :: tab(:, :)
98    integer :: x, y, my_rank
99
100    allocate (tab(0:12, 0:12))
101    do i=0, 11
102      do while (y /= 12)
103        tab(x, y) = (x+1) * (y+1)
104        y = y+my_rank+1
105      end do
106    end do
107    deallocate (tab)
108  end subroutine func3
109
110  subroutine print_arg(arg)
111    character(128) :: arg
```

The line number 103 is highlighted, indicating the location of the segmentation fault. A 'Program Stopped' dialog box is open in the foreground, displaying the following information:

Program Stopped

Processes 0-15:

Process stopped in func3 (cstartmpi.f90:103) 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

Buttons: Continue, Pause

The 'Locals' panel on the right shows the current state of variables:

Variable Name	Value
tab	4198128
x	0
y	0

DDT takes you to the exact line where Segmentation fault occurred, and you can pause and investigate

Invalid Memory Access

```
96  subroutine func3(my_rank)
97      integer, allocatable :: tab(:, :)
98      integer :: x, y, my_rank
99
100     allocate(tab(0:12, 0:12))
101     do i=0, 11
102     do while (y /= 12)
103     tab(x, y) = (x+1)*(y+1)
104     y = y+my_rank+1
105     end do
106     end do
107     deallocate(tab)
108     end subroutine func3
109
110     subroutine func4(my_rank)
111     character :: name
```

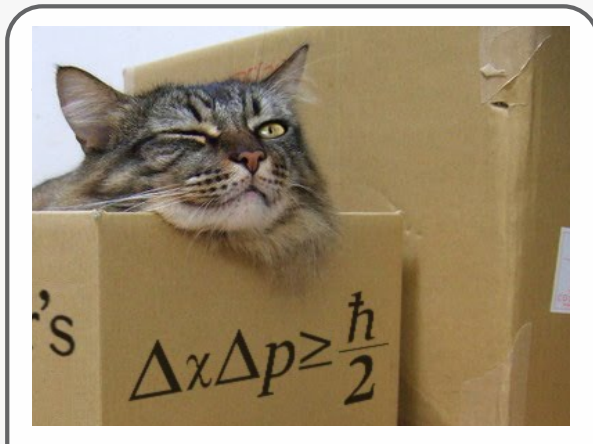
On this line:
16 Processes: ranks 0-15
1 Thread (Rank 0): #1
Name: tab
Type: integer (kind=4), ALLOCATABLE
(0:12, 0:12)

Variable Name	Value
tab	[[[0] = ([0] = -158
x	4198128
y	0

The array `tab` is a 13x13 array, but the application is trying to write a value to `tab(4198128,0)` which causes the segmentation fault.

`i` is not used, and `x` and `y` are not initialized

It works... Well, most of the time



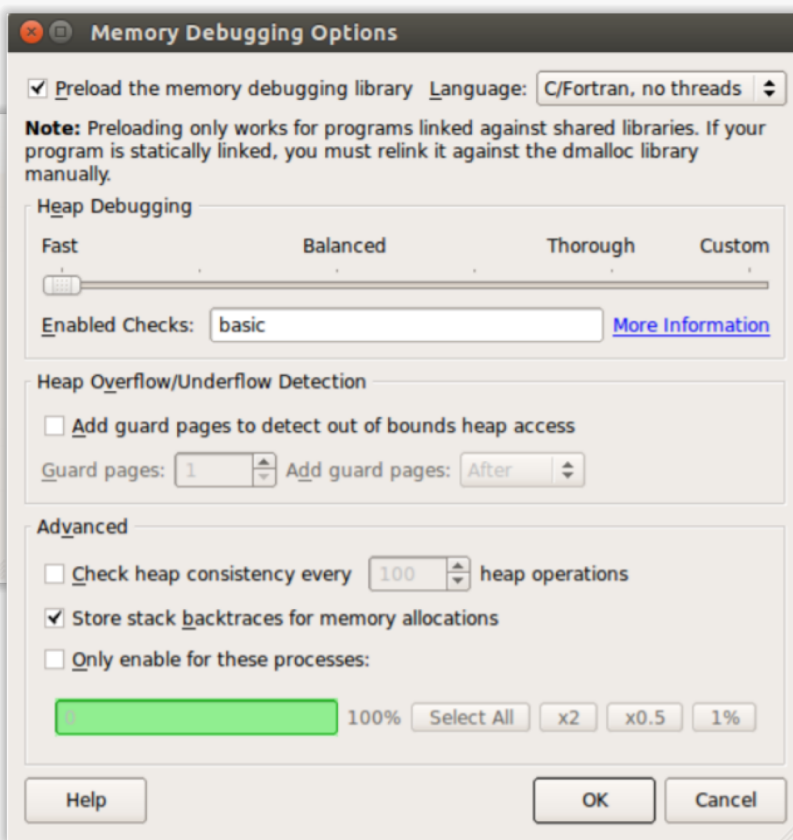
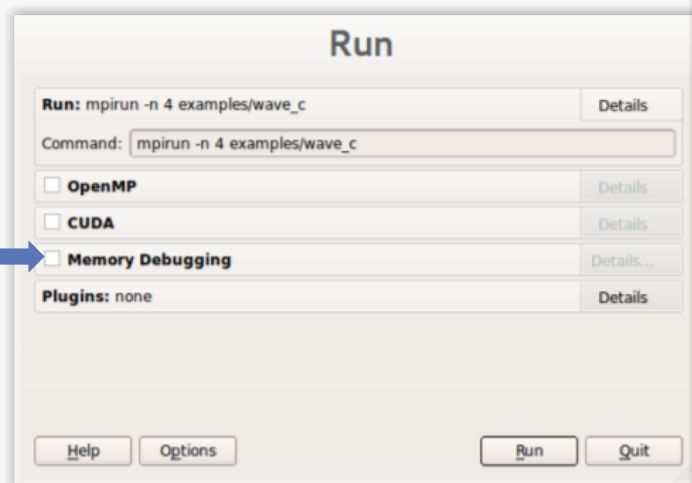
**SCHRODIN
BUG**



A strange behaviour where the application “sometimes” crashes is a typical sign of a memory bug

Arm DDT is able to force the crash to happen

Advanced Memory Debugging



Heap debugging options available

Fast

basic

- Detect invalid pointers passed to memory functions (e.g. malloc, free, ALLOCATE, DEALLOCATE,...)

check-fence

- Check the end of an allocation has not been overwritten when it is freed.

free-protect

- Protect freed memory (using hardware memory protection) so subsequent read/writes cause a fatal error.

Added goodness

- Memory usage, statistics, etc.

Balanced

free-blank

- Overwrite the bytes of freed memory with a known value.

alloc-blank

- Initialise the bytes of new allocations with a known value.

check-heap

- Check for heap corruption (e.g. due to writes to invalid memory addresses).

realloc-copy

- Always copy data to a new pointer when re-allocating a memory allocation (e.g. due to realloc)

Thorough

check-blank

- Check to see if space that was blanked when a pointer was allocated/freed has been overwritten.

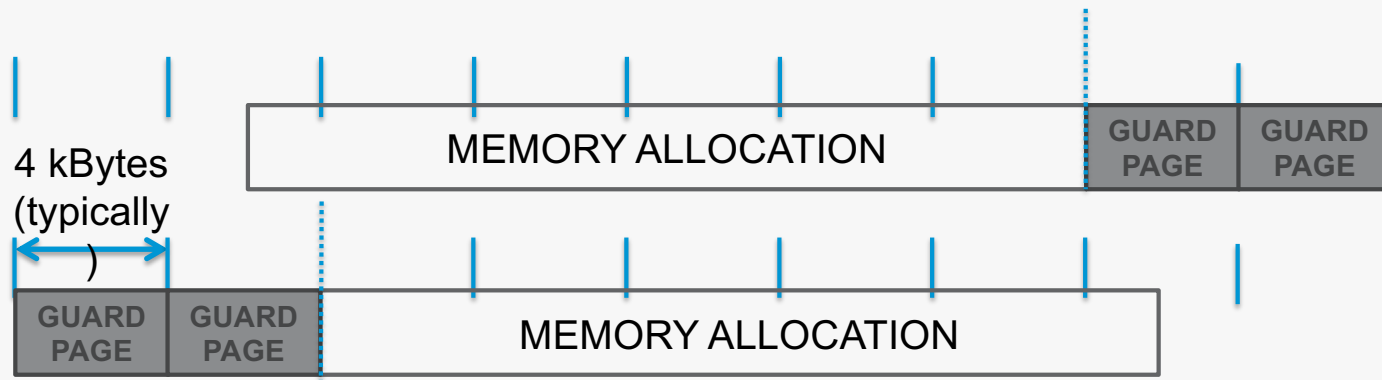
check-funcs

- Check the arguments of addition functions (mostly string operations) for invalid pointers.

See user-guide:

Chapter 12.3.2

Guard pages (aka “Electric Fences”)



- **A powerful feature....:**
 - Forbids read/write on guard pages throughout the whole execution
(because it overrides C Standard Memory Management library)
- **... to be used carefully:**
 - Kernel limitation: up to 32k guard pages max (“mprotect fails” error)
 - Beware the additional memory usage cost



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

All 24576 processes (0-24575) Paused: 17223 Playing: 7353 Finished: 0
Currently selected: 260 (on nid00194, pid 9481, main thread IWP 9481)

Create Group

Project Files

Search (Ctrl+K)

- VolumeTrav
- wave.c
- weird.c
- WholeGeon
- Writer.cc
- wspace.c
- XdrFileWrite
- XdrMemRead
- XdrMemWrite
- XdrReader.c
- XdrWriter.cc
- XmiAbstract
- xyzpart.c

```
MpiEnvironment.cc xyzpart.c  
551 ikvsortii(ntsamples, allpicks);  
552  
553  
554 /* Select the final splitters. Set the boundaries to s  
555 for (i=1; i<nps; i++)  
556 mypicks[i] = allpicks[i*ntsamples/nps];  
557 mypicks[0].key = IDX_MIN;  
558 mypicks[nps].key = IDX_MAX;  
559  
560  
561 WCOREPOP; /* free allpicks */  
562  
563 STOPTIMER(ctrl, ctrl->AuxTmr2);  
564 STARTTIMER(ctrl, ctrl->AuxTmr3);  
565
```

Locals Current Line(s) Current Stack

Current Line(s)

Variable Name	Value
allpicks	0x2aab8055e070
i	2245
mypicks	0x2a6f8f0
nps	24575
ntsamples	1818550

Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Logbook

Stacks

Processes	Threads	Function
17223	17223	main (main.cc:37)
17223	17223	SimulationMaster::SimulationMaster (SimulationMaster.cc:63)
17223	17223	SimulationMaster::Initialise (SimulationMaster.cc:154)
17223	17223	hemelb::geometry::GeometryReader::LoadAndDecompose (GeometryReader.cc:188)
17223	17223	hemelb::geometry::GeometryReader::OptimiseDomainDecomposition (GeometryReader.c
17223	17223	hemelb::geometry::decomposition::OptimisedDecomposition::OptimisedDecomposition
17223	17223	hemelb::geometry::decomposition::OptimisedDecomposition::CallParmetis (Optimised
17223	17223	PairMETIS_V3_PartGeomKway (gkmetis.c:90)
17223	17223	libpametis_Coordinate_Partition (xyzpart.c:58)
17223	17223	libpametis_PseudoSampleSort (xyzpart.c:556)

Type: none selected

Evaluate

Expression	Value
i + ntsamples	-212322546

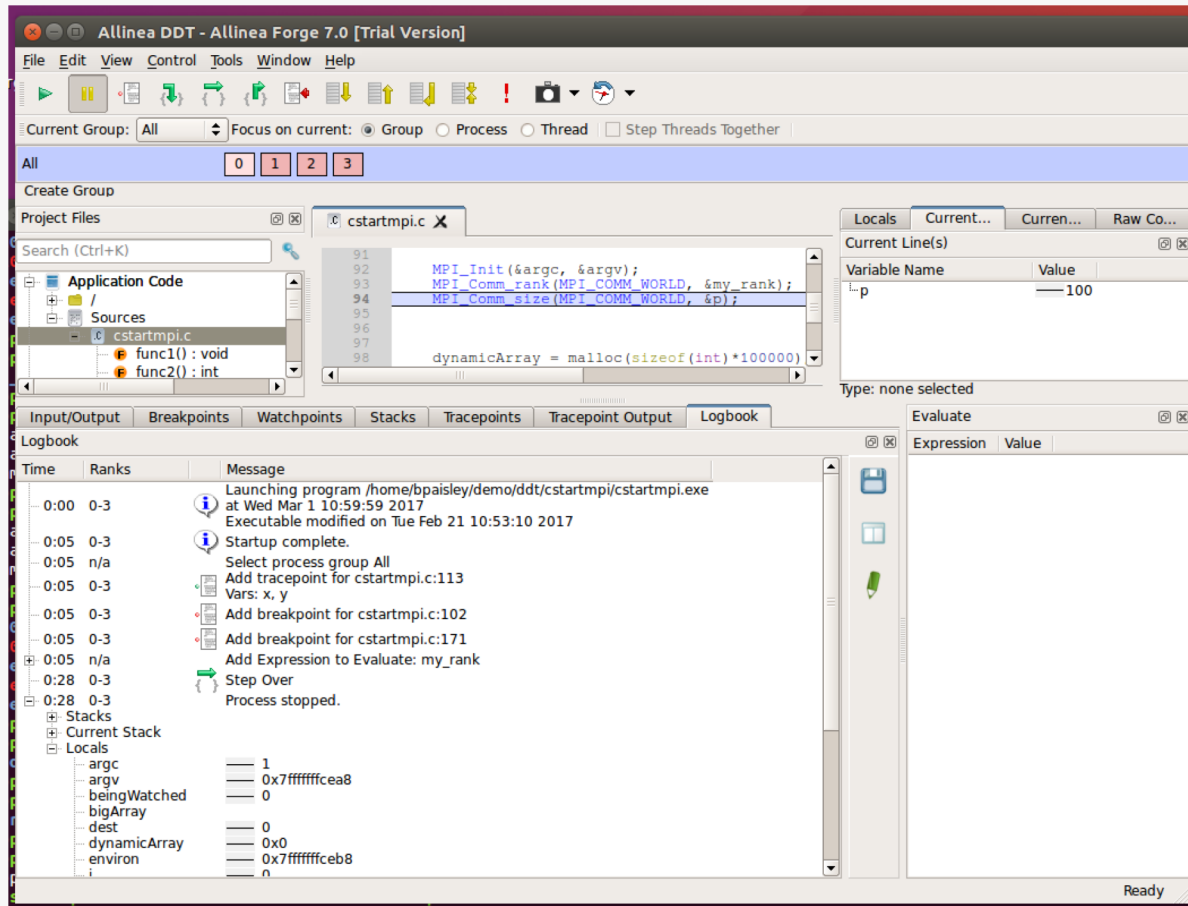
Type: int
Range: from -2147259746 to -12282046
49/17223 processes equal

New Bugs from Latest Changes

The screenshot shows a debugger window with the following components:

- Project Files:** A tree view showing the source code structure, including `wave_openmp.c`.
- Source Code:** The main code window displays the C source code for `wave_openmp.c`. A red box highlights line 227, which is `oldval = values;` inside a swap function. A green dashed line indicates a comment: `Master receives results from workers and print`.
- Threads:** A panel at the top shows four threads. Thread 1 is the main thread, and thread 3 is a worker thread. Both are currently at line 227.
- Stacks:** A panel at the bottom left shows the call stack for the selected thread. The stack includes `main (wave_openmp.c:354)` and `update (wave_openmp.c:227)`.
- Evaluate:** A panel at the bottom right shows the current values of variables: `newval` is `0x7ffff4b7a010`, `oldval` is `0x7ffff4b7a010`, and `values` is `0x7ffff4b7a010`.

Track Your Changes in a Logbook



caption

Arm DDT Demo

Five great things to try with Alinea DDT

Tracepoint	Process	Values logged
vhone 9005	900 ranks 12, 14-17, 22-23, 12	mtype 210-3527 jcol 2-40 mod pey
vhone 9001	900 ranks 12, 14-17, 22-23, 12	ls 1 kmax pec
vhone 9005	900 ranks 12, 14-17, 22-23, 12	mtype 210-3527 jcol 2-40 mod pey
vhone 9001	900 ranks 12, 14-17, 22-23, 12	ls 1 kmax pec
vhone 9005	900 ranks 12, 14-17, 22-23, 12	mtype 210-3527 jcol 2-40 mod pey
vhone 9001	900 ranks 12, 14-17, 22-23, 12	ls 1 kmax pec
vhone 9005	884, 10 12, 14-17, 22-23, 12	
vhone 9001	880, 10 17, 14	

The scalable print alternative

```

r (i = 0 ; i < SIZE M; i++)
for (j = 0 ; j < SIZE_0; j++)
  C[i][j] = 0;

r (i = 0 ; i < SIZE M; i++)
for (j = 0 ; j < SIZE N; j++)
  for (k = 0 ; k < SIZE_0; k++)
    C[i][j] += A[i][k] * B[k][j];

intf("M 1/2")
(argc
for (i = 0;
{ pri
  
```

Stop on variable change

```

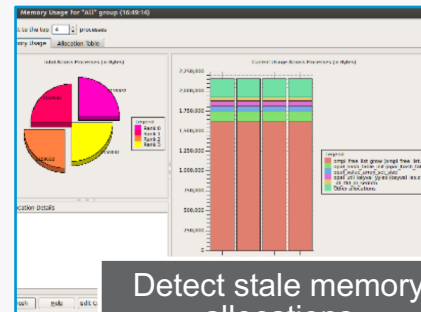
hello.c
43
44 test=-1;
45 }
46
47 void func3()
48 {
49 void* i = (void*) 1;
50 while(i++ || i)
51 free((void*)i);
52
53 portability 'i' is of type 'void *'. When using void pointers in calcula
54
55 {
56 ty
57 ty
58 in
  
```

Static analysis warnings on code errors

```

&& strcmp(argv[i], "crash")) {
0;
}
*(char **)argv[i]);
ll se
Program Stopped
Processes 0-3:
Memory error detected in main (hello.c:118):
null pointer dereference or unaligned memory access
Note: the latter may sometimes occur spuriously if guard
enabled
Tip: Use the stack list and the local variables to explore
current state and identify the source of the error.
r, "I
= 1;
ist.s
= 0;
  
```

Detect read/write beyond array bounds



Detect stale memory allocations

Arm DDT cheat sheet

Load the environment module

- \$ module load **forge/19.1.2**
- \$ module unload **xalt**

Prepare the code

- \$ cc **-O0 -g** myapp.c -o myapp.exe

Start Arm DDT in interactive mode

- \$ **ddt** aprun -n 8 ./myapp.exe arg1 arg2

Or use the reverse connect mechanism

- On the login node:
 - \$ ddt &
- (or use the remote client) **<- Preferred method**
- Then, edit the job script to run the following command and submit:
 - **ddt --connect** aprun -n 8 ./myapp.exe arg1 arg2

Theta Specific Settings

Configure the remote client

Install the Arm Remote Client

- Go to : <https://developer.arm.com/products/software-development-tools/hpc/downloads/download-arm-forge>

Connect to the cluster with the remote client

- Open your Remote Client
- Create a new connection: Remote Launch → Configure → Add
 - Hostname: <username>@theta.alcf.anl.gov
 - Remote installation directory:
`/soft/debuggers/forge-19.1.2-2019-08-06`

Static Linking Extra Steps

To enable advanced memory debugging features in DDT, you must link explicitly against our memory libraries
Simply add the link flags to your Makefile, or however appropriate

```
lflags = -L/soft/debuggers/ddt/lib/64 -Wl,--undefined=malloc -ldmalloc -Wl,--allow-multiple-definition
```

Questions?

Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

감사합니다

धन्यवाद

arm