OFM

Debugging and Profiling HPC Applications

ATPESC August 7, 2018

> Ryan Hulguin ryan.hulguin@arm.com HPC Applications Engineer

© 2018 Arm Limited

Agenda

- General Debugging and Profiling Advice
- Arm Software for Debugging and Profiling
- Debugging with DDT
- Profiling with MAP
- Theta Specific Settings

Debugging

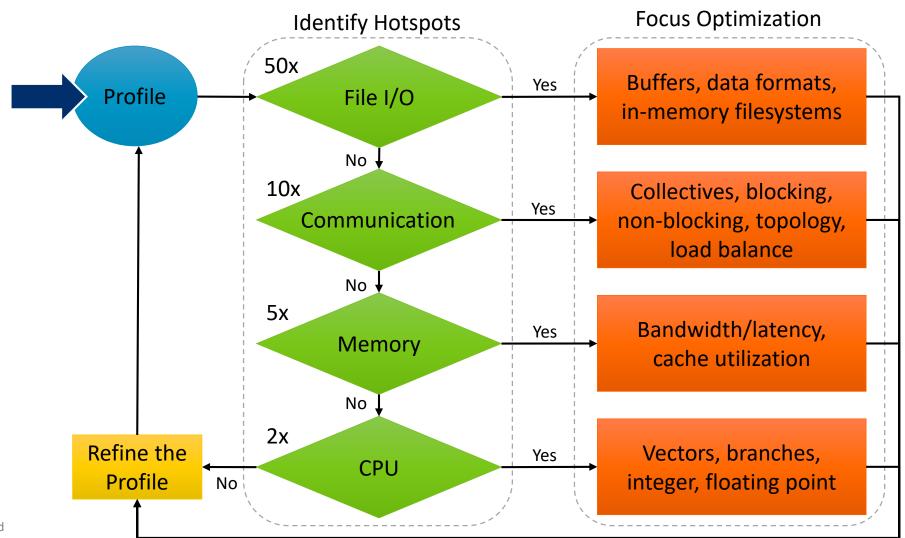
Transforming a broken program to a working one

How? TRAFFIC!

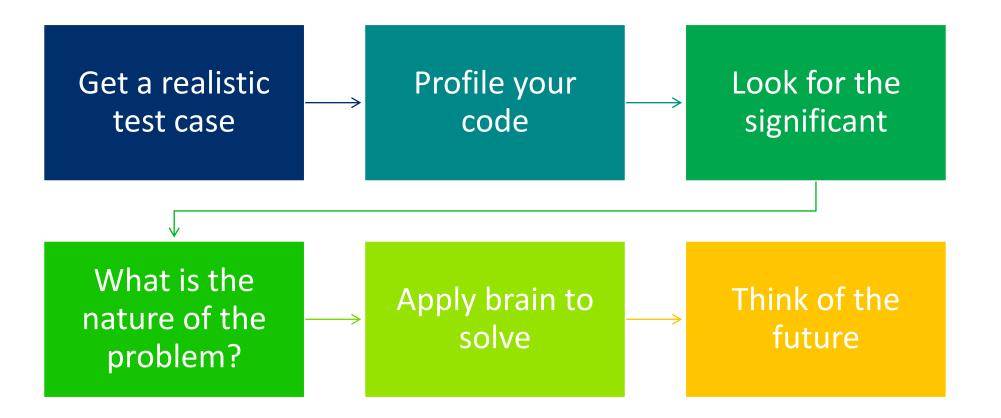
- Track the problem
- **R**eproduce
- Automate (and simplify) the test case
- Find origins where could the "infection" be from?
- Focus examine the origins
- Isolate narrow down the origins
- Correct fix and verify the test case is successful

Profiling

Profiling is central to understanding and improving application performance.



Performance Improvement Workflow





Arm Software



© 2018 Arm Limited

Arm Forge

An interoperable toolkit for debugging and profiling



Commercially supported by Arm





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

Arm Performance Reports

Characterize and understand the performance of HPC application runs



Commercially supported by Arm



Accurate and astute insight



Relevant advice to avoid pitfalls

Gathers a rich set of data

- Analyses metrics around CPU, memory, IO, hardware counters, etc.
- Possibility for users to add their own metrics

Build a culture of application performance & efficiency awareness

- Analyses data and reports the information that matters to users
- Provides simple guidance to help improve workloads' efficiency

Adds value to typical users' workflows

- Define application behaviour and performance expectations
- Integrate outputs to various systems for validation (e.g. continuous integration)
- Can be automated completely (no user intervention)

Run and ensure application correctness

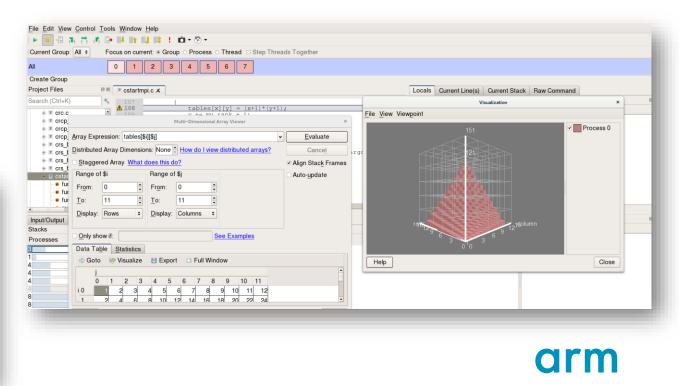
Combination of debugging and re-compilation

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

Examples:

\$> ddt --offline mpirun -n 48 ./example
\$> ddt mpirun -n 48 ./example

15		2:17.256	0-7	P	lay						
16	0	2:18.048	4-7	Р	Process stopped at breakpoint in main (cpi.c:50).						
17				A	dditional I	nformation					Values
					Stacks						mprocs: 8 myid: // from 0 to 7 n: 100
					Processes	Function					umprocs: 8 myid: / from 0 to 7 n: 100
				4	1-7 i	main (cpi.c:50)					numprocs: 8 myid: / from 0 to 7 n: 100
18		2:19.048	n/a	S	elect proc	elect process 4				numprocs: — 8 myid: 🦯 from 0 to 7 n: — 100	
19				A	dditional I	nformation					numprocs: 8 myid: // from 0 to 7 n: 100
					Current S	Stack					numprocs: 8 myid: / from 0 to 7 n: 100
				ľ							numprocs: 8 myid: // from 0 to 7 n: 100
					Locals						numprocs: 8 myid: / from 0 to 7 n: 100
				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
9		© 2018 Arm		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
9		© 2010		13	2:18.325	main (cpi.c:46)	0-3	done: -	– 0 i:	from 97 to 10	0 numprocs: — 8 myid: from 0 to 3 n: — 100



Understand application behaviour

Set a reference for future work

- Choose a representative test case with known behavior
- Analyse performance with Arm Performance Reports

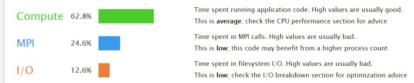
Example:

\$> perf-report mpirun -n 16 mmult_c.exe

Is it performant?

Command:	mpiexec ./mmult_c.exe 7168
Resources:	1 node (28 physical, 56 logical cores per ne
Memory:	125 GiB per node
Tasks:	28 processes
Machine:	r4163
Start time:	Wed May 17 2017 10:25:58 (UTC+10)
Total time:	33 seconds
Full path:	/short/c25/pw9396/allinea_wshop-day1/ 0_charac_performance

Summary: mmult_c.exe is Compute-bound in this configuration



This application run was Compute-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

As little time is spent in MPI calls, this code may also benefit from running at larger scales.

CPU

A breakdown of the 62.8% CPU time: Scalar numeric ops 0.2% | Vector numeric ops 13.4% Memory accesses 80.3%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

MPI

A breakdown of the 24.6% MPI time:

 Time in collective calls
 6.3%

 Time in point-to-point calls
 93.7%

Effective process collective rate 0.00 bytes/s

Effective process point-to-point rate 114 MB/s Most of the time is spent in point-to-point calls with an average transfer rate. Using larger messages and overlapping

transfer rate. Using larger messages and overlapping communication and computation may increase the effective transfer rate.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage 448 MiB

Peak process memory usage 1.24 GiB

Peak node memory usage 16.0%

There is significant variation between peak and mean memory usage. This may be a sign of workload imbalance or a memory leak.

The peak node memory usage is very low. Running with fewer MPI processes and more data on each process may be more efficient.

1/0

., -		
A breakdown of the 12.69	% I/O time:	
Time in reads	0.0%	
Time in writes	100.0%	
Effective process read rate	0.00 bytes/s	J.
Effective process write rate	3.56 MB/s	

Most of the time is spent in write operations with a very low effective transfer rate. This may be caused by contention for the filesystem or inefficient access patterns. Use an I/O profiler to investigate which write calls are affected.

Threads

A breakdown of how multiple threads were used:

Computation	0.0%	
Synchronization	0.0%	1
Physical core utilization	99.7%	
System load	101.8%	

No measurable time is spent in multithreaded code.



Optimize the application for Arm

- Measure all performance aspects with Arm MAP parallel profiler
- Identify bottlenecks and rewrite some code for better performance

Examples:

\$> map --profile mpirun -n 48 ./example

Profiled: clover_leaf on 32 processes, 4 nodes, 32 core	s (1 per process) Sampled from: Wed Nov 9 2016 15:28:37 (UTC) for 309.1s	Hide Metrics
Application activity	en benende sternen van senere gebenderske kejsen alweet anderet denin en senere serene serente beten van slocke mete	en en begen en de la sectere par en egen de la feren en de la Malanca de Malanca de Sectere de Sectere de Sect
Iterations / s 3.86 /s	the second shift a second s	
Grind time 0.00 s		
Step time 136+00		and a particular standard and have
15:28:37-15:33:46 (309.138s): Main thread compute 0.2	%, OpenMP 80.0 %, MPI 19.7 %, OpenMP overhead 0.1 %, Sleeping 0.1 %	Zoom 🍕 🇮 🖲
[≆] hydro.f90 ×		Time spent on line 75
51.2%	flux_calc() advection() reset_field()	Breakdown of the 51.2% time spent on this line: Executing instructions 0.0% Calling other functions 100.0%
Input/Output Project Files OpenMP Stacks Ope OpenMP Stacks	nMP Regions Functions	8
	tion(s) on line clover_leaf ydro	
	advection_module=advection limesten_module=timesten	

rofiled: My_code.exe on 64 process	es Started: Fri Sep 20 14:59:0	9 2013 Runtime: 35s Time in MPI: 45 %	Hide Metrics
Memory usage (M)			
9.4 - 777.9 (454.6 avg)			
MPI call duration (ms)			-
0 - 5,575.1 (341.0 avg)			
CPU floating-point (%)		Anna a a a ana ana ana ana ana a	
0 - 90 (8.2 avg)		A CONTRACT AND A CONTRACT	
14:59:09-14:59:44 (range 34.773	s): Mean Memory usage 454.6	M; Mean MPI call duration 341.0 ms; Mean CPU floating-point 8.2 %;	Metrics, Reset
My_code.f90 🗵			
		itationn	
	100 !~~~~~~ 101 !	MODULE EXCITATION	~~~~~
	102 !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	ative [e]	~~~~~~
	140 !		
	141 ! 142 !	MAIN CODE	
	143 🗉 program Vel_V	/ort_3D_FP	
	144 use data_mc 145 use wall exc:	itation	
	146 implicit none	2	
	147 include 'mpi 148 double precis		
		<pre>sion :: max_omx_dt,max_omy_dt,max_omz_dt,t,time_cal ption,i,j,k,nn,fwcnt,count max,counter,ios,next file at,W cnt(1:4)</pre>	
	150 character*30	<pre>:: str,file_type,str_t,num_2_str</pre>	
. 1%	151 152 call MPI INI	T(ierr)	
			>
	allel Stack View		
llel Stack View			8
Time ∇ M	MPI Function(s) on line vel vort 3d fp , <un< td=""><td>Source</td><td>Position</td></un<>	Source	Position
.0%	1.4% Etime integration	. program Vel Vort 3D FP call time integration	My_code.f90:143 My_code.f90:330
.9% 5	.3% I mod rank read file	. call mod rank read file all its own(str,nn,ios) ! Restart from last checkpoint	My_code.f90:297
.8%	5.3% evelocity_solver	call velocity solver	My_code.f90:337
.8%	.4% 🗄 vel vort 3d fp	call cell identifier	My code.f90:190



Debugging with DDT

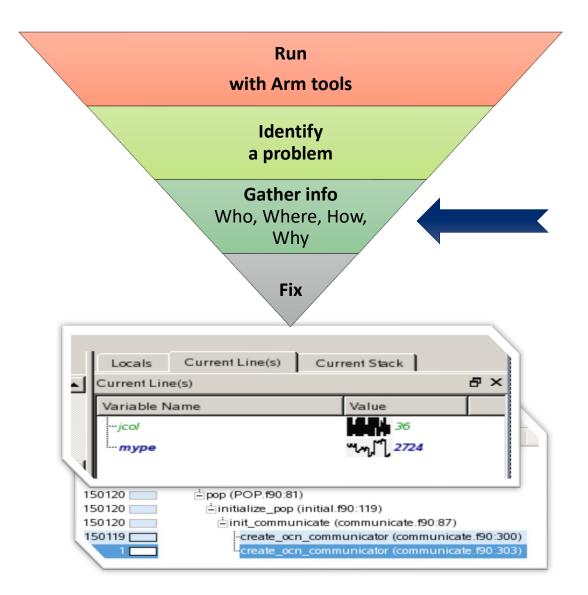


© 2018 Arm Limited

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

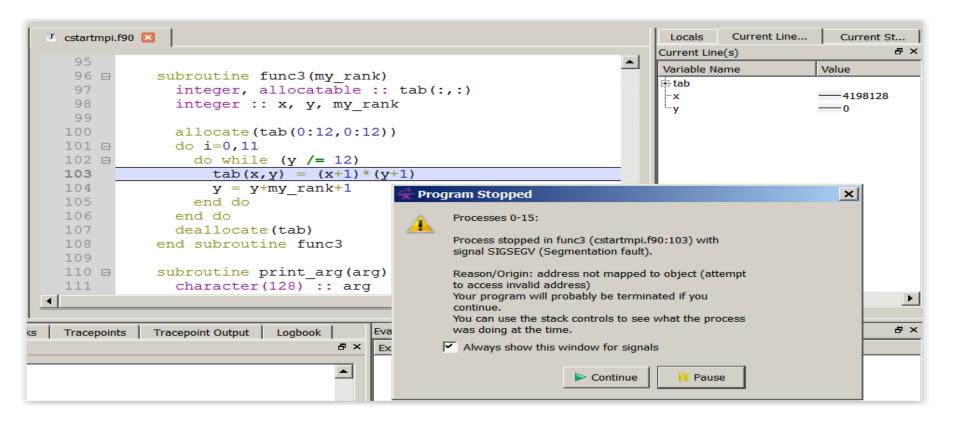
Segmentation Fault

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

E Te	ermina	l - rhulg	uin@ryanlir	ux:/me	nedia/sf_VM_share/Training_Codes/1_2_cstartmpi/f90 🛧 💷 🗙
File	Edit	View	Terminal	Tabs	s Help
	0×7 0×7 0×7 0×4 0×4 0×4 0×7	FEF17 FEF17 FEF16 017EE 014B8 F585E F585E		nc3 tart	at cstartmpi.f90:103 tmpi at cstartmpi.f90:62
rer	note 	machi		ted	orocess rank 12 with PID 18305 on node on signal 11 (Segmentation fault). 90]\$

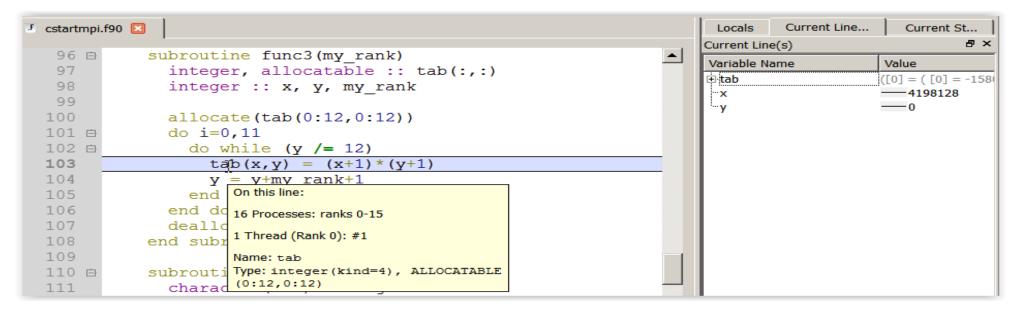
What happens when it runs under DDT?

Segmentation Fault in DDT



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

Invalid Memory Access



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.

is not used, and x and y are not initialized

It works... Well, most of the time



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

		✓ Preload the memory debugging library Language: C/Fortran, no threads \$
Run		Note: Preloading only works for programs linked against shared libraries. If your program is statically linked, you must relink it against the dmalloc library manually.
Run: mpirun -n 4 examples/wave_c	Details	Heap Debugging
Command: mpirun -n 4 examples/wave_c		Fast Balanced Thorough Custom
OpenMP		
CUDA		Enabled Checks: basic More Information
Memory Debugging	Details	
Plugins: none	Details	Heap Overflow/Underflow Detection
		Add guard pages to detect out of bounds heap access
		<u>G</u> uard pages: 1 Add guard pages: After
		Advanced
Help Options	<u>R</u> un Quit	Check heap consistency every 100 + heap operations
		✓ Store stack backtraces for memory allocations
		Only enable for these processes:
		0 100% Select All x2 x0.5 1%
		Help OK Cancel

Memory Debugging Options

Heap debugging options available



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 goodiness

•Memory usage, statistics, etc.

Balanced

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

alloc-blank

free-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 reallocating a memory allocation (e.g. due to realloc)

Thorough

•Check to see if space that was blanked when a pointer was

check-blank

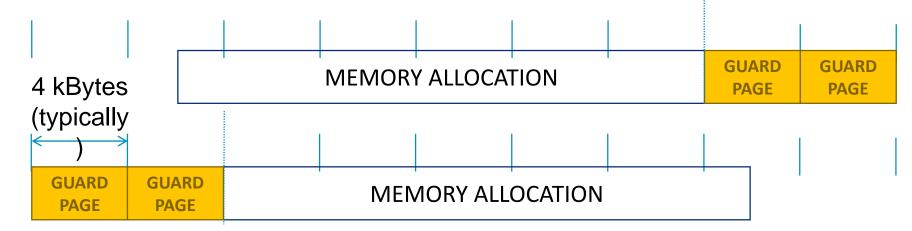
allocated/freed has been overwritten.

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")



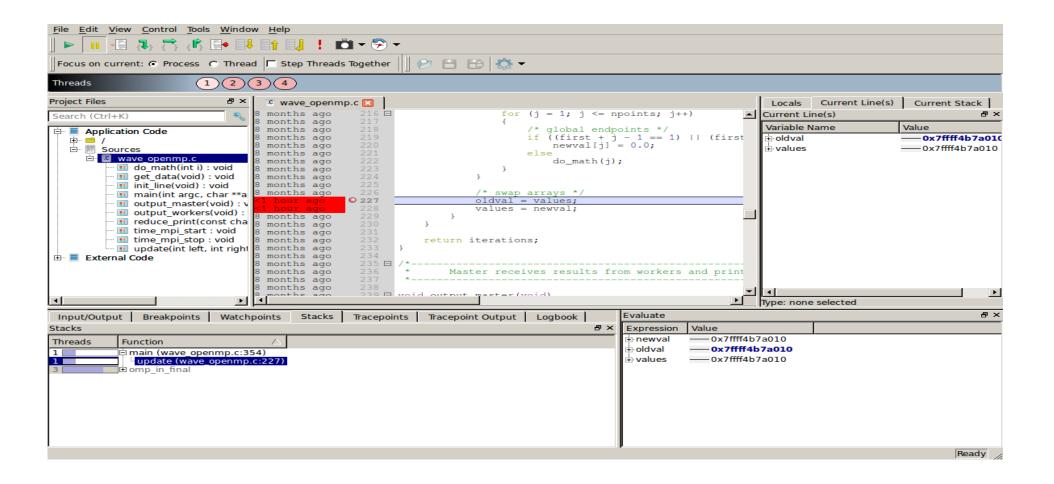
- A powerful feature...:
 - Forbids read/write on guard pages throughout the whole execution

(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

Allinea DDT 4.2.1-36484	_ = ×
<u>File view Control Search Tools window H</u> elp	
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 944	31)
Create Group	
Project Files 💿 🗵 🔤 MpiEnvironment.cc 💥 💽 xyzpart.c 💥	Locals Current Line(s) Current Stack
Search (Ctrl+K) 🔍 551 ikvsortii(ntsamples, allpicks);	Current Line(s)
552	Variable Name Value
Image: Interpretation of the second seco	allpicks 0x2aab8055e010 0x2aab8055e010
weird.c 555 for (i=1: i <npes: i++)<="" td=""><td></td></npes:>	
	-npes
Writer.cc 557 mypicks[0].key = IDX_MIN; wspace.c 557 mypicks[0].key = IDX_MIN;	ntsamples — 1818550
The m XdrEileWrite 558 mypicks[npes].key = IDX_MAX;	
XdrMemRea 559 Feo	
■ W XdrMemWrit 560 Solution	
WCOREPOP; /* Tree allpicks */	
m XmiAbstract 563 STOPTIMER(ctrl, ctrl->AuxTmr2);	
🔹 💽 xyzpart.c 🗧 564 STARTTIMER(ctrl, ctrl->AuxTmr3);	
External Code 565	
	Type: none selected
Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Logbook	Evaluate Ø 🗵
Stacks	•
Processes Threads Function	i * ntsamples
	e: int
17223 17223 SimulationMaster: Initialise (SimulationMaster cc:154)	nge: from -2147259746 to -12282046
17223 17223 energyReader.cc:188)	7223 processes equal
17223 17223 ichemelb::geometry::GeometryReader::OptimiseDomainDecomposition (GeometryReader 17223 17223 ichemelb::geometry::decomposition::OptimisedDecomposition::OptimisedDecomposition	
17223 17223 hemelb::geometry::decomposition::OptimisedDecomposition::CallParmetis (Optimis	
17223 I7223 17223 I7223 17223 I7223 I7223 I7223 I7223 I7223	
17223 17223 17223 Ibparmetis_PseudoSampleSort (xyzpart.c:556)	
	7353 processes playing
Computer III titan-ext7 Allinea DDT 4.2.1-36484	📢) 🚺 Sun Aug 10, 7:50 PM 🔛

New Bugs from Latest Changes



Track Your Changes in a Logbook

😣 🖻 🗉 Allinea DDT - Allinea Forge 7.0 [Trial Version]				
<u>File Edit View Control Tools Window H</u> elp				
📔 🕨 🖪 🚓 🚓 🦚 📪 📑 📑 📑 📫 🔹 🕈 🗖 🕶				
Current Group: All 🗘 Focus on current: Group O Process O Thread Departments Together				
All 0 1 2 3				
Create Group				
Project Files 💿 🗵 💽 cstartmpi.c 🗙	Locals	Current	Curren	Raw Co
Search (Ctrl+K) 91	Current I	Line(s)		0 🗙
92 MPI_Init(&argc, &argv);	Variable	Name	Value	
Image: Second state Image: Second st	ⁱ p			
P Sources 95				
96 97				
e funcl(): void 98 dynamicArray = malloc(sizeof(int)*100000)				
F func2(): int	Type: non	e selected		
	.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,			
Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Logbook		Evaluate		0 8
Logbook	0 X	Expression V	/alue	
Time Ranks Message	8			
Launching program /home/bpaisley/demo/ddt/cstartmpi/cstartmpi.exe				
- 0:00 0-3 (1) at Wed Mar 1 10:59:59 2017 Executable modified on Tue Feb 21 10:53:10 2017				
- 0:05 0-3 (i) Startup complete.				
- 0:05 n/a Select process group All				
- 0:05 0-3 Add tracepoint for cstartmpi.c:113				
- 0:05 0-3 Add breakpoint for cstartmpi.c:102	¥			
- 0:05 0-3 Add breakpoint for cstartmpi.c:171				
0:28 0-3 Step Over				
□ 0:28 0-3 Process stopped.				
B Stacks				
⊕- Current Stack — — ⊖- Locals				
argc — 1				
argv — 0x7ffffffcea8 beingWatched — 0				
bigArray				
dest 0 dynamicArray 0x0				
environ — 0x7ffffffceb8				
				Deede
				Ready

Inspect AVX Registers

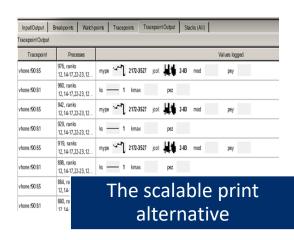
arm PRODE	Arm DDT - Arm Forge 18.2 [Trial Version]	^ _ □ × P :
File Edit View Control Tools Window Help		
Ĩ► 🔲 🗄 🖧 😤 🕼 📴 🖬 🔛 ! 🛍 ד 🕏 ד		2
Focus on current: 💿 Process 🔿 Thread 🗌 Step Threads Together 🛛 📀 💿		
Threads 64 threads (1-64) Paused: 64 Playing: 0 Finished: 0		
Show threads Currently selected: 1 LWP 18108		
Proj 🖉 🗷 🕅 mmatestl.c 🕱	Registers Locals Current Line(s) Current Stack Raw Command	
Sea		Ø 🕷
Sum = 0.0; 135 136 \square 135 136 \square 135 136 \square 135 136 \square	Variable Name Value	
⊕ ● / 137 ⊖ ▶ S ● 138 sum += pA[p*i+k] * pB[n*	-k	
1 1 3 9	+ pA	
Disassembly	□ × → pC 0x7fffe3bd6040	
Eunction: 'main'	mble	
Address Offset Bytes Instruction	TimeS) / (double [) LOOP_COUNT);	
0x4016a2 <+562> 4d 8d 14 0f lea (%r15,%rcx,1),%r10		
0x4016a6 <+566> 4d 8d 0c 0e lea (%r14,%rcx,1),%r9 0x4016aa <+570> 49 89 d8 mov %rbx,%r8		
0x4016ad <+573> 62 al 7c 48 28 d0 vmovaps %zmm16,%zmm18 0x4016b3 <+579> 33 ff xor %edi,%edi		
0x4016b5 <+581> 4d 89 c5 mov %r8,%r13	Type: none selected	
0x4016be <+590> 62 e1 7c 48 28 c8 vmovaps %zmm0,%zmm17	Evaluate	@ (X)
Imp 0x4016c4 <th< th=""> <th< th=""> <t< th=""><th>Expression Value</th><th></th></t<></th<></th<>	Expression Value	
0x4016d0 <+608> 62 c1 7c 48 10 1c ba vmovups (%r10,%rdi,4),%zmm19 0x4016d7 <+615> 62 c1 7c 48 10 64 ba 01 vmovups 0x40(%r10,%rdi,4),%zmm20	fte 🗄 \$zmm18	
0x4016df <+623> 62 d2 65 40 b8 85 00 00 vfmadd231ps 0x4000(%r13), %zmm19, %zmm0 0x4016e9 <+633> 62 c1 7c 48 10 6c ba 02 vmovups 0x80(%r10, %rdi, 4), %zmm11	Image: style float float [[0] = 4096, [1] = 0, [2] = 2048, [3] = 0, [4] = 0, [5] = 0, [6] = 0, [7] = 0, [8] = 0, [9] = 0, [10] = 0, [11] = 0, [12] = 0, [7] = 0, value double [[0] = 5.760887010628376e-315, [1] = 5.7194417803360855e-315, [2] = 0, [3] = 0, [4] = 0, [5] = 0, [6] = 0, [7] =	
0x4016f1 <+641> 62 d2 5d 40 b8 85 40 40 00 00 vfmadd231ps 0x4040(%r13),%zmm20,%zmm0	1 v 64 int8 {[0] = 0, [1] = 0, [2] = -128, [3] = 69, [4] = 0, [5] = 0, [6] = 0, [7] = 0, [8] = 0, [9] = 0, [10] = 0, [11] = 69, [12] = 0 v v32 int16 {[0] = 0, [1] = 17792, [2] = 0, [3] = 0, [4] = 0, [5] = 17664, [6] = 0, [7] = 0, [8] = 0, [9] = 0, [10] = 0, [11] = 0, [12] = 0, [
0x4016fb <+651> 62 c2 65 00 vfmadd231ps 0x0(%r13),%zmm19,%zmm18 0x401702 <+658> 62 c2 5d 40 b8 4d 01 vfmadd231ps 0x40(%r13),%zmm20,%zmm17	$ \oplus \sqrt{32 \text{ muld}} \{[0] = 0, [1] = 17792, [2] = 0, [3] = 0, [4] = 0, [5] = 17804, [0] = 0, [0] = 0, [0] = 0, [10] = 0, [11] = 0, [12] = 0, [12] = 0, [13] = 1166016512, [1] = 0, [2] = 1157627904, [3] = 0, [4] = 0, [5] = 0, [6] = 0, [7] = 0, [8] = 0, [9] = 0, [10] = 0, [12] = 0, [12] = 0, [13] = 0, [14] = 0, [15] = 0, [15] = 0, [15] = 0, [15] = 0, [16] $	
0x401709 <+665> 62 d2 55 40 b8 85 80 40 00 00 vfmadd231ps 0x4080(%r13),%zmm21,%zmm0 0x401713 <+675> 62 c1 7c 48 10 74 ba 03 vmovups 0xc0(%r10,%rdi,4),%zmm22	→ v8_int64 {[0] = 1166016512, [1] = 1157627904, [2] = 0, [3] = 0, [4] = 0, [5] = 0, [6] = 0, [7] = 0} ⊕ v4_int128 {[0] = 0x00000004500000000045800000, [1] = 0x0000000000000000000000000000000000	200000000000000000000000000000000000000
0x40171b <+683> 48 83 c7 40 add \$0x40,%rdi 0x40171f <+687> 62 c2 55 40 b8 55 02 vfmadd231ps 0x80(%r13),%zmm21,%zmm18		
0x401726 <+694> 62 c2 4d 40 b8 4d 03 vfmadd231ps 0xc0(%13),%zmm22,%zmm17 0x40172d <+701> 62 d2 4d 40 b8 85 c0 40 00 00 vfmadd231ps 0x40c0(%13),%zmm22,%zmm0		
0x401737 <+711> 49 81 c5 00 01 00 00 add \$0x100,%r13		
0x40173e <+718> 48 81 ff 00 10 00 00 cmp \$0x1000,%rdi 0x401745 <+725> 72 89 jb 0x4016d0 <main+608></main+608>		
0x401747 <+727> 49 81 c0 00 80 00 00 add \$0x8000,%r8		
<u>H</u> elp <u>C</u> los	e	
25 © 2018 Arm Limited		arm

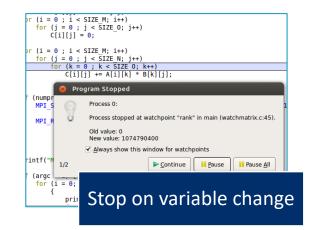
Arm DDT Demo

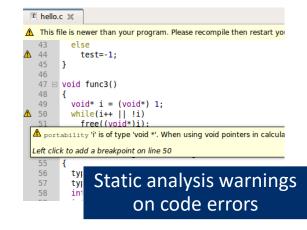


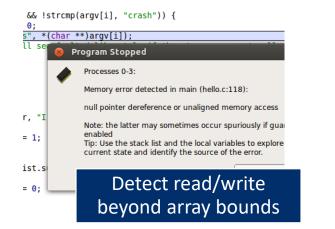
© 2018 Arm Limited

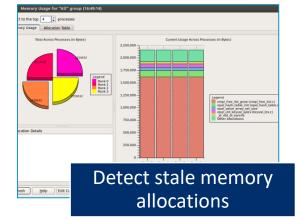
Five great things to try with Allinea DDT













Arm DDT cheat sheet

Load the environment module

• \$ module load forge/18.2.1

Prepare the code

• \$ cc -OO -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

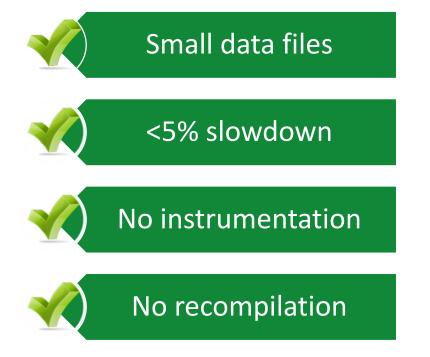
- On the login node:
 - \$ ddt &
- (or use the remote client) <- Preferred method
- Then, edit the job script to run the following command and submit:
 - **ddt --connect** aprun -n 8 ./myapp.exe arg1 arg2

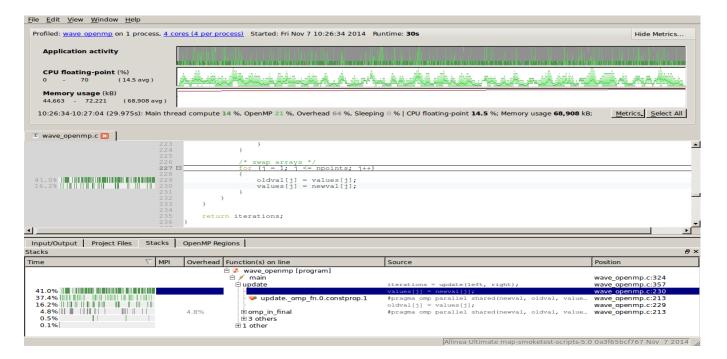
Profiling with MAP



© 2018 Arm Limited

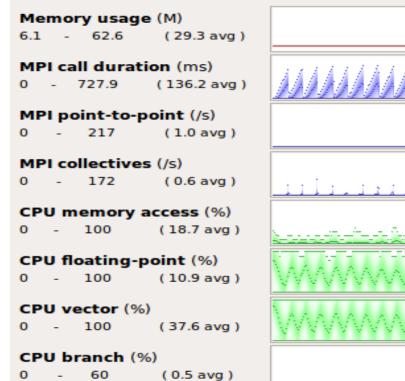
Arm MAP – The Profiler

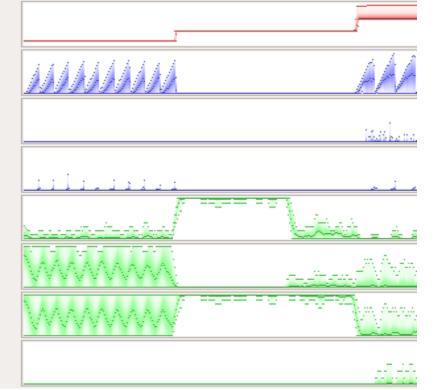




orm

Glean Deep Insight from our Source-Level Profiler





Track memory usage across the entire application over time

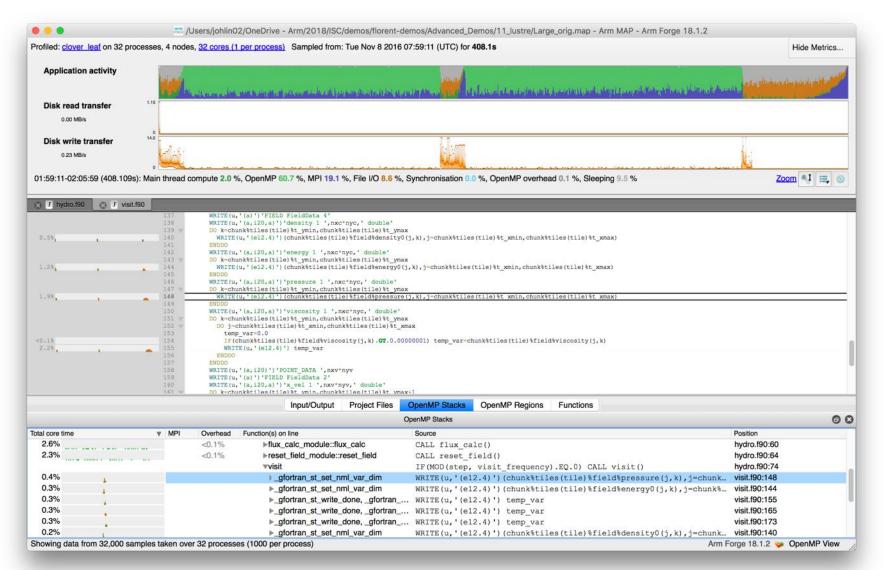
Spot MPI and OpenMP imbalance and overhead

Optimize CPU memory and vectorization in loops

Detect and diagnose I/O bottlenecks at real scale

Initial profile of CloverLeaf shows surprisingly unequal I/O

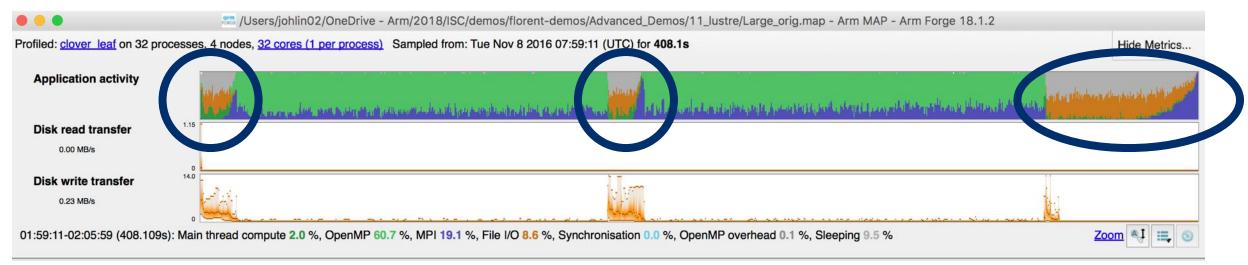
Each I/O operation should take about the same time, but it's not the case.



arm

Symptoms and causes of the I/O issues

Sub-optimal file format and surprise buffering.



- Write rate is less than 14MB/s.
- Writing an ASCII output file.
- Writes not being flushed until buffer is full.
 - Some ranks have much less buffered data than others.
 - Ranks with small buffers wait in barrier for other ranks to finish flushing their buffers.

Solution: use HDF5 to write binary files

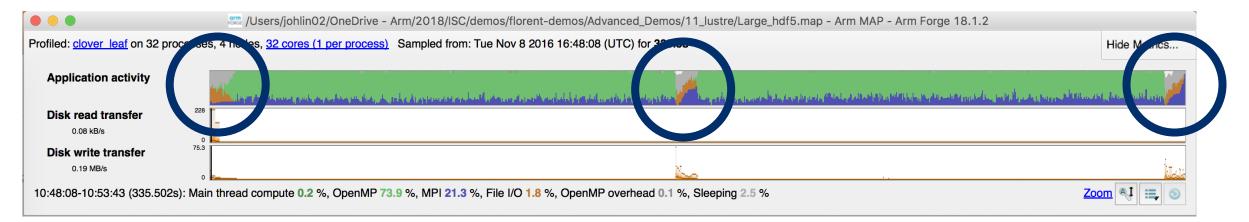
Using a library optimized for HPC I/O improves performance and portability.

• • •	sta /	Jsers/johlin02/OneDrive - Arm/2018/ISC/demos/florent-demos/Advanced_Demos/11_lu	ustre/Large_hdf5.map - Arm í	MAP - Arm Forge 18.1.2	
Profiled: <u>clover leaf</u> on 32 proce	esses, 4 node	s, <u>32 cores (1 per process)</u> Sampled from: Tue Nov 8 2016 16:48:08 (UTC) for 335.5s			Hide Metrics
Application activity		Alatena en anteren en la si el si sur mandrillaren alcunteri di basidaren si interenteri di basidaren alcunter	والمراجع والمراجع والمراجع والمراجع والمراجع والمراجع	Rasials additions in the matter particular, and the	and the second state of the second second
CPU floating-point	100	n en en la servició de la constructiva de la construcción de la construcción de la construcción de la construcc	And the second	ممر ممر محمد محمد مرتدر .	
37.8 %					
Memory usage	164				
151 MB	0				
10:48:08-10:53:43 (335.502s):	Main thread	compute 0.2 %, OpenMP 73.9 %, MPI 21.3 %, File I/O 1.8 %, OpenMP overhead 0.1 %, Slee	eping 2.5 %		Zoom 🔍 🚍 💿
S I hydro.f90 S I visit.f90				Time spent	on line 237 💿 S
	224 225	: CALL h5screate_simple_f(2, dims2d, space, hdferr)		Breakdown of the 0.3% tir	me spent on this line:
	226 227	! ! Create the dataset. We will use all default properties for this		Executing instructions	0.0%
	228 229	example.			00.0%
	230	dataset='pres'			00.0 %
	231 232	CALL h5dcreate_f(file, dataset, H5T_IEEE_F64LE, space, dset, hdferr)			
	233 234				
	235	! Write the data to the dataset.			
0.3%	236 237	! CALL h5dwrite f(dset, H5T NATIVE DOUBLE, chunk%tiles(tile)%field%pressure, dims2d, hdfe	err)		
	238 239				
	240	! Close and release resources.			
	241 242	! CALL h5dclose_f(dset , hdferr)			
	243 244	CALL h5sclose_f (space, hdferr)			
	245				
	246 247	dims2d(1)=chunk%tiles(tile)%t xmax - chunk%tiles(tile)%t xmin + 1			
	248	<pre>dims2d(2)=chunk%tiles(tile)%t_ymax - chunk%tiles(tile)%t_ymin + 1</pre>			
	249 250	1			
	251	! Create dataspace. Setting size to be the current size.			
		Input/Output Project Files OpenMP Stacks OpenMP	Regions Functions		
		OpenMP Stacks			0 8
Total core time	▼ MPI	Overhead Function(s) on line Source			Position
		▼7 others			
0.7%	0.7%		r(kernel_total,totals)		hydro.f90:111
			requency).EQ.0) CALL vi		hydro.f90:74
0.1%				hunk%tiles(tile)%field	
0.1% <0.1%				hunk%tiles(tile)%field	
<0.1%		_		hunk%tiles(tile)%field	
<0.1% <0.1%				hunk%tiles(tile)%field	
<0.1%		<pre>>_h5_dble_interface_MOD_h5dw CALL h5dwrite_f(dset, >h5d_MOD_h5dclose_f CALL h5dclose f(dset</pre>		hunk%tiles(tile)%field	
					visit.f90:130

arm

Solution: use HDF5 to write binary files

Using a library optimized for HPC I/O improves performance and portability.



- Replace Fortran write statements with HDF5 library calls.
 - Binary format reduces write volume and can improve data precision.
 - Maximum transfer rate now 75.3 MB/s, over 5x faster.
- Note MPI costs (blue) in the I/O region, so room for improvement.

Arm MAP cheat sheet

Load the environment module (manually specify version)

• \$ module load forge/18.2.1

Generate the wrapper libraries (static is default on Theta)

• \$ make-profiler-libraries --lib-type=static

Unload Darshan module (It wraps MPI calls which cannot be used with MAP)

• \$ module unload darshan

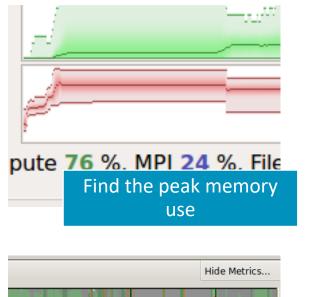
Follow the instructions displayed to prepare the code

- \$ cc -O3 -g myapp.c -o myapp.exe -Wl,@/path/to/profiler_wrapper_libraries/allinea-profiler.ld
- Edit the job script to run Arm MAP in "profile" mode
- \$ map --profile aprun -n 8 ./myapp.exe arg1 arg2

Open the results

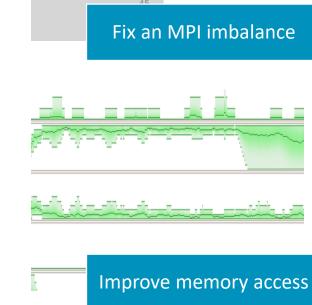
- On the login node:
 - \$ map myapp_Xp_Yn_YYYY-MM-DD_HH-MM.map
- (or load the corresponding file using the remote client connected to the remote system or locally)

Six Great Things to Try with Allinea MAP



Make sure OpenMP

regions make sense



30

31

32

33 34

36

38

39

40

41

42

43

! late to the party

do j=1,20*nprocs; a

do from=1,nprocs-1
 call MPI RECV(b,

call MPI BARRIER(MPI CO

do j=1,50; b=sqrt

print *, "Answer f

if (pe /= 0) then
 call MPI SEND(a, si

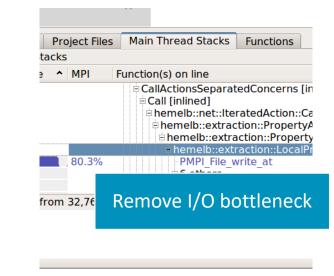
end if

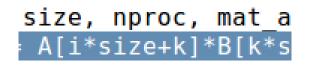
else

end do

end if

end do







, Sleeping

Theta Specific Settings



© 2018 Arm Limited

Configure the remote client

Install the Arm Remote Client

 Go to : <u>https://developer.arm.com/products/software-development-</u> 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/ddt

ALCF Documentation available at

https://tinyurl.com/debugging-cpw-2018-05

Static Linking Extra Steps

To enable advanced memory debugging features, you must link explicitly against our memory libraries

Simply add the link flags to your Makefile, or however appropriate

Iflags = -L/soft/debuggers/ddt/lib/64 -WI,--undefined=malloc -Idmalloc -WI,--allow-multipledefinition

In order to profile, static profiler libraries must be created with the command make-profiler-libraries --lib-type=static

Instructions to link the libraries will be provided after running the above command

Sample usage Commands

Theta

rpn=64 ddt aprun -n \$((COBALT_JOBSIZE*rpn)) -N \$rpn -d \$depth -j 1 -cc depth ./myProgram.exe

map aprun -n \$((COBALT_JOBSIZE*rpn)) -N \$rpn -d \$depth -j 1 -cc depth ./myProgram.exe

Questions?



© 2018 Arm Limited

Thank You! Danke! Merci! 谢谢! ありがとう! **Gracias!** Kiitos! 감사합니다 धन्यवाद



Arm Forge Hands-on Examples



Hands-on files

The files for the examples that follow can be obtained on theta at the following location

/projects/Comp_Perf_Workshop/allinea/allinea_handson.tgz

This extracts 2 directories: demonstrations and allinea_examples

The demonstrations are there for you to play with and ask questions

The examples are more like guided exercises

Launch Remote client

Be sure to launch the remote client first

Using a remote launch on your local machine is preferred

Alternatively you can forward X11 when connecting to the login node of theta, and launch it there

```
module load forge/18.2.1
forge &
```

If you accidentally close this window (easy to do), you will have to start it again

Hands-on Examples

These examples are meant to be run on Theta in an interactive session

```
qsub -I -q training -t 120 -n 1 --proccount 64
```

Once a session has been allocated, load the Forge module

```
module load forge/18.2.1
```

Before Generating MAP profiles

Static profiler libraries need to be created before MAP profiles can be generated

Go to the allinea_examples/wrapper directory

Run

make-profiler-libraries --lib-type=static

The Makefiles for the examples have already been modified to look for the profiler libraries in this directory

Go to exercise 1 – [Bug] Solver is not converging

Exercise objectives

- Familiarize with DDT user interface
- · Inspect values of u using multidimensional array viewer
- Set watchpoint for diffnorm_global
- Set breakpoint at line 89

• Typical run commands to use:

- \$> cd allinea_examples/1_debug/
- \$> make

• Key DDT commands

On the login node:

\$> forge &

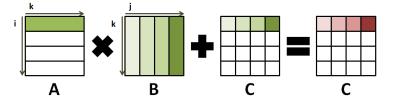
In a submission file/interactive job:

\$> ddt --connect -n 4 ./jacobi.exe

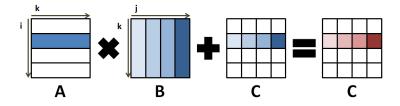
Matrix Multiplication Example

 $C = A \times B + C$

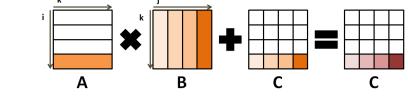




Slave process 1







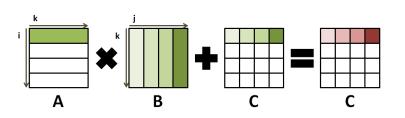
Slave process n-1

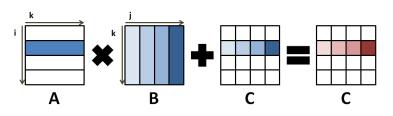


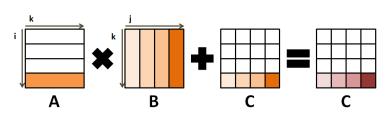
Matrix Multiplication Example Continued

 $C = A \times B + C$

- The "Master" process initializes matrices A, B and C
- The "Master" process sends the whole matrix B along with slices of A and C to the "Slave" processes
- The "Master" and "Slave" processes perform the matrix multiplication function on the domain that has been given to them and everyone computes a slice of C
- The "Master" process retrieves all slices of C and puts the result matrix C together

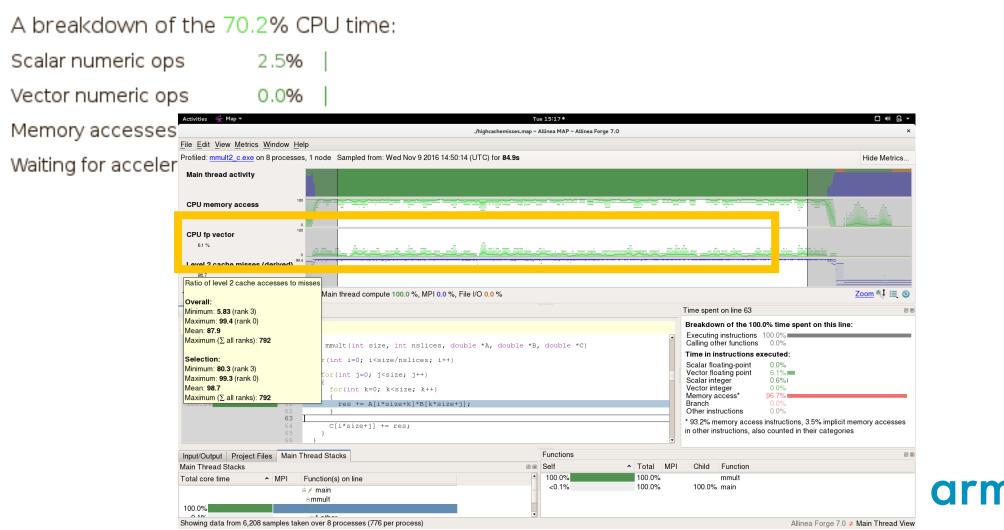






Use Allinea Forge to vectorize your codes

CPU



Use Forge to optimize memory access

Activities 😤 Map 🗸	Tue 15:17 ●						
Jhighcachemisses.map – Allinea MAP – Allinea Forge 7.0 X							
File Edit View Metrics Window Help							
Profiled: mmult2_c.exe on 8 processes, 1 r					Hide Metrics		
Main thread activity							
CPU memory access							
61% Level 2 cache misses (derived) Ratio of level 2 cache accesses to misses							
Main thread compute 100.0 %, File I/O 0.0 %							
Overall: Minimum: 5.83 (rank 3)	Time spent on line 63 Allocation Details						
Maximum: 99.4 (rank 0)				down of the 1	⊡ 3,145,728 b	vtes, 2 alloca	ations
Mean: 87.9 Maximum (Σ all ranks): 792			Brouk		🖻 main (m	emkind stats	.c:21) (1.048.576 bytes, 1 allocation)
Maximum (2, an ranks). 732	mmult(int size, int nslices, double *A, double *B, double			Cur	At: 0)	7fffedde600	0, size: 1,048,576 bytes
Selection: Minimum: 80.3 (rank 3)	r(int i=0; i <size i++)<="" nslices;="" th=""><th>6,500,000</th><th></th><th></th><th>⊡ main (m</th><th>emkind_stats</th><th>c:22) (2,097,152 bytes, 1 allocation) 0, size: 2,097,152 bytes, kind: memkind_hbw</th></size>	6,500,000			⊡ main (m	emkind_stats	c:22) (2,097,152 bytes, 1 allocation) 0, size: 2,097,152 bytes, kind: memkind_hbw
Maximum: 99.3 (rank 0)	for(int j=0; j <size; j++)<="" th=""><th>E and and</th><th></th><th></th><th>····· AL: 0)</th><th>k/iiidd20000</th><th>o, size: 2,097,152 bytes, kind: memkind_nbw</th></size;>	E and and			····· AL: 0)	k/iiidd20000	o, size: 2,097,152 bytes, kind: memkind_nbw
Mean: 98.7	<pre>for(int k=0; k<size; <="" k++)="" pre=""></size;></pre>	6,000,000		-			
Maximum (Σ all ranks): 792	res += A[i*size+k]*B[k*size+j];	5,500,000				-	
63 64 65 66 }	<pre>C[i*size+j] += res; }</pre>	5,000,000					
Input/Output Project Files Main Thread Stacks Function		4,500,000					
Main Thread Stacks B Self Total core time ^ MPI Function(s) on line ^ 100.0%		E					
		4,000,000	-				
emmult		Ξ					
100.0%	3,500,000					Legend	
Showing data from 6,208 samples taken over 8 processes (776 per process)		=		-			main (memkind_stats.c)
		3,000,000					ompi_free_list_grow hwloc_nolibxml_backend_init
		2,500,000					opal_hwloc191_hwloc_bitmap_alloc
		2,500,000					opal_hash_table_init2
		2,000,000					Other allocations
		2,000,000					
		1,500,000					
		1					
	1,000,000						
		500,000					
		500,000		-			
		6.E					
nited			Rank 0	Rank 1	Rank 2	Rank 3	

arm

Go to exercise 2

Exercise objectives

- Generate initial baseline profile
- Ensure the matrices are stored in the MCDRAM (if applicable)
- Fix the inefficient memory access issues
- Further enable vectorization with Intel compiler flag -xMIC-AVX512
- Generate profile with MAP after applying changes

Typical run commands to use:

\$> cd allinea_examples/2_memory_accesses/

\$> make

Key Map commands

On the login node:

\$> forge &

In a submission file/interactive job:

\$> map --profile -n 64 ./mmult2_c.exe

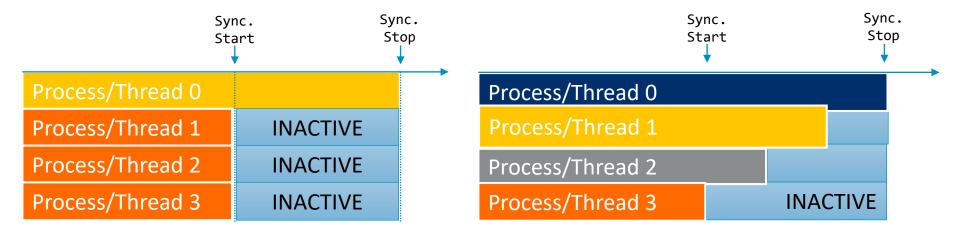
\$> map --connect ./mmult2_c_*.map

How to identify load balancing issues?

Problem: "one or some process(es) have too much work"

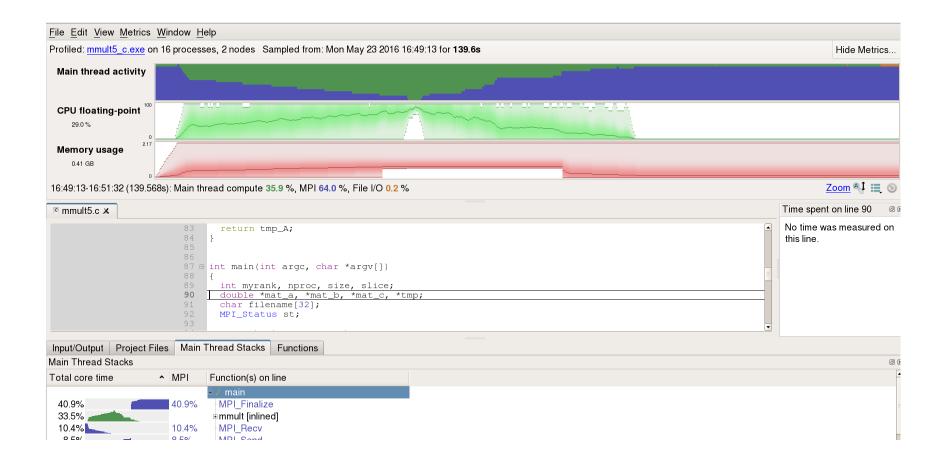
Clues visible in synchronization

- MPI Collective calls (MPI_Barrier, _Broadcast, etc.) with no actual data transfer
- · Idle cores where threads are stuck in locks/mutexes



Total runtime: 100 sec Total CPU time available: 400 sec Total CPU time actually used: 250 sec Efficiency: 62.5% of the machine time Total runtime: 100 sec Total CPU time available: 400 sec Total CPU time actually used: 300 sec Efficiency: 75% of the machine time

Use Allinea MAP to balance your workloads



Go to exercise 3

Exercise objectives

- Expose workload imbalance issues in the code (preferably on 2 nodes)
- Make suggestions to fix the problem

• Typical run commands to use:

\$> cd allinea_examples/3_imbalance/

\$> make

Key Map commands

\$> map --profile -n 64 ./mmult4_c.exe

\$> map --connect mmult4_c_*.map

Go to exercise 4

Sometimes optimizations introduce bugs of their own

- Exercise objectives
 - Use ddt in offline mode to detect memory leaks
 - Examine the debug_report.txt file
 - Fix the leak
 - Generate new report
- Typical run commands to use:
- \$> cd allinea_examples/4_memory_leak/
- \$> make

• Key ddt commands

\$> ddt --offline --mem-debug --output=debug_report.txt -n 64 ./mmult6_c.exe

Solutions to exercises

Solutions to these exercises can be found in the **.solution** directory in each of the exercises