Arm Debugging and Profiling Tools Tutorial

Hosted By NERSC July 15, 2020

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

© 2018 Arm Limited

Orm

Agenda

- Arm Software for Debugging and Profiling
- Debugging with DDT
- Generating Performance Reports
- Profiling with MAP
- Using Arm tools with Python

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

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

Build a culture of application performance & efficiency awareness

- Analyzes 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 aprun -n 48 ./example
\$> ddt --connect aprun -n 48 ./example

15		2:17.256	0-7	Play	and the second se
16	0	2:18.048	4-7	Process stopped at breakpoint in main (cpi.c:50).	
17				Additional Information	Values
				▼ Stacks	mprocs: 8 myid: // from 0 to 7 n: 100
				Processes Function	umprocs: 8 myid: from 0 to 7 n: 100
				4-7 main (cpi.c:50)	numprocs: 8 myid: // from 0 to 7 n: 100
18		2:19.048	n/a	Select process 4	numprocs: 8 myid: // from 0 to 7 n: 100
19				Additional Information	numprocs: 8 myid: // from 0 to 7 n: 100
				Current Stack	numprocs: 8 myid: / from 0 to 7 n: 100
					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 6	5 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	3 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 8	1 to 88 numprocs: - 8 myid: / from 0 to 7 n: - 100
		@ 201	8 Arm	12 2:18.323 main (cpi.c:46) 0-7 done: — 0 i: 🖍 from 8	9 to 96 numprocs: - 8 myid: / from 0 to 7 n: - 100
5 © 2016 AIII				13 2:18.325 main (cpi.c:46) 0-3 done: 0 i: from 9	7 to 100 numprocs: — 8 myid: from 0 to 3 n: — 100



Visualize the performance of your application

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

Examples: \$> map --profile -n 48 ./example

Profiled: clover_leat on 32 processes, 4 nodes, 32 cores (1 per process) Sampled from: Wed Nov 9 2016 15:28:37 (UTC) for 309.1s	Hide Metrics
Application activity	di penarti di margana di kentere panang meta Matan anti Addisanti at Janahadi, sebatan kanta
Iterations / s ^{Re} 386/8	en and the second s
Grind time 0.00 s	
Step time 0.00 s	and the second
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 📲 🇮 💿
7 hydro.190 🗴	Time spent on line 75 @ 8
7.3 7.3 CALL flux_calc() 51.2% 75 CALL advection() 3.3% 77 CALL rest_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 OpenMP Regions Functions	
OpenMP Stacks Total core time • MPI Overhead Function(s) on line # / clover_leat # hydro 51.2% 8.4% # 0 1% • advection_module_advection 22.2% 0.9% # timesten_module_timesten	8 8

Promea: My_coae.exe on 64 prod	cesses Starteo: Fri Sep 20 14:59:09 2013 Runtime: 355 Time in MPI: 45%	Hide Metrics
Memory usage (M)		
9.4 - 777.9 (454.6 av	g)	
MPI call duration (ms)		~
0 - 5,575.1 (341.0 av	g)	-
0 - 90 (8.2 avg)		
14:59:09-14:59:44 (range 34.7	773s): Mean Memory usage 454.6 M; Mean MPI call duration 341.0 ms; Mean CPU floating-point 8.2 %;	Metrics, Reset
F My code.f90 🔀		
	87	~~~~
	88 🗉 module wall_excitation 🛛 n	
	101 ! MODELE EXCITATION	
	103 + module derivativee	
	140 !	
	141 ! MAIN CODE	
	142	
	143 B program ver vor so rr	
	145 use wall excitation	
	146 implicit none	
	147 include 'mpif.h'	
	double precision :: max omx dt, max omy dt, max omz dt, t, time cal	
	149 Integer :: option, , , , , , , , , , , , , , , , , , ,	
	151	
0.1%	152 call MPI_INIT(ierr)	
	153 call MPI_COMM_SIZE(MPI_COMM_WORLD, npro,ierr)	
put/Output Project Files	Parallel Stack View	
allel Stack View		1
tal Time	MPI Function(s) on line Source	Position
3.0%	= vel vort 3d fp , <un 3d="" fp<="" program="" td="" vel="" vort=""><td>My_code.f90:143 My_code.f90:330</td></un>	My_code.f90:143 My_code.f90:330
16.9%	5.3% @mod_rank_read_file call imod_rank_read_file_all_its_own(str,nn,ios) ! Restart from last checkpoint	My_code.f90:297
12.00/	6.3% 🕸 velocity_solver call velocity_solver	My_code.f90:337
12.8%	The support of the second seco	
1.8%	14% Evel work 24 fe	My, codo (90)-100

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.

🔳 Terminal -	hulguin@ryanlinux:/media/sf_VM_share/Training_Codes/1_2_cstartmpi/f9(🛧 🗕 🗆 🗙
File Edit V	iew Terminal Tabs Help
#2 0x7F0	085B8E66F
#0 0x7FE	F17094467
#1 0x7FE	F17094AAE
#2 0x7FE	F1637F66F
#3 0x401	.7EB in func3 at cstartmpi.f90:103
#4 0x401	4B8 in cstartmpi at cstartmpi.f90:62
#0 0x7F5	85EDF6467
#1 0x7F5	85EDF6AAE
#2 0x7F5	85E0E166F
mpirun no	ticed that process rank 12 with PID 18305 on node
remotema	chine exited on signal 11 (Segmentation fault).
[rhulguir	@ryanlinux f90]\$

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

Track Your Changes in a Logbook

Se a contra do 1 - Allinea Forge 7.0 [Trial Version]									
File Fait Alex Coutrol Tools Millaon Helb									
· · · · · · · · · · · · · · · · · · ·									
Current Group: All 🗘 Focus on current: Group O Process O Thread D Step Threads Together									
All 0 1 2 3									
Create Group									
Project Files 💿 🗷 💽 cstartmpi.c 🗙	Locals	Current	Curren	Raw Co					
Search (Ctrl+K)	Current I	Line(s)		6 🕱					
92 MPI_Init (&argc, &argv);	Variable	Name	Value						
Application Code → 93 MPI_Comm_size(MPI_COMM_WORLD, &my_rank);	^L p								
G D R Sources									
cstartmpi.c									
e funcl(): void 98 dynamicArray = malloc(sizeof(int)*100000)									
	Type: non	e selected							
	.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	Evaluate							
input/Output Breakpoints Watchpoints Stacks iracepoints iracepoint Output Logbook		Evaluate		신 🗶					
Logbook	0 ×	Expression V	alue						
Time Ranks Message									
Launching program /home/bpaisley/demo/ddt/cstartmpi/cstartmpi.exe									
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									
- 0:05 0-3 - Add breaknoint for cstartmni c:102	Ŷ								
0:05 0-3									
H 0:05 n/a Add Expression to Evaluate: my rank									
0.28 0-3 Step Over									
E ⊡ 0:28 0-3 Process stopped.									
B Stacks									
Grief Stack Filter Stack Fi									
argc — 1									
argv — 0x/itffffcea8									
bigArray									
dest0									
environ 0x7ffffffceb8									
				Doortu					
				кеаду					

<u>Hie view Control Search Tools window H</u> elp											
Current Group:	Current Group: All 🗣 Focus on current: 💿 Group 🔿 Process 🔿 Thread 📃 Step Threads Together										
All	24576 processes (0-24575) Paused: 17223 Playing: 7353 Finished: 0 II Currently selected: 260 (on nid00194, pid 9481, main thread IWP 9481)										
Create Group	Create Group										
Project Files	Ø×	MpiEnviror	nment.cc 🔀	.C xyzpart.c 💥				Locals Cur	rent Line(s)	Current Stack	L
Search (Ctrl+K)	IumeTrav ave.c aird.c holeGeon riter.cc space.c IrFileWrite IrMemRea IrMemWrit IrReader.c IrWriter.cc nIAbstract spart.c	551 552 553 554 7 555 6 556 557 r 558 r 559 560 561 V 562 563 ST 564 ST	ikvsortii(/* Select for (i=1; : mypicks[0] mypicks[0] wCOREPOP; WCOREPOP; OPTIMER(ct ARTTIMER(ct	htsamples, all the final spli i <npes; i++)<br=""><u>i] = allpicks</u> .key = IDX es].key = IDX /* free allpi rl, ctrl->Aux trl, ctrl->Aux</npes;>	lpicks); itters. Set i <u>[i*ntsamples,</u> _MIN; _MAX; icks */ icks */ Tmr2); xTmr3);	the bounda /npes];	aries to :	Current Line(s) Variable Name		Value 0x2aab805 2245 0x2a6f8f0 24575 1818550	@ X 5e010
	Þ	•						Type: none selec	ted		
Input/Output	Breakpoints	Watchpoints	Stacks	Tracepoints Tra	acepoint Output	Logbook		Evaluate			0 ×
Stacks							6	Expression	Value		
Processes Th	nreads F	unction						i * ntsample	es -212322	2546	
17223 17 17223 17 17223 17 17223 17 17223 17 17223 17 17223 17 17223 17 17223 17 17223 17	223	i main (main.cc:3 È SimulationMa È SimulationM È hemelb∷g È hemelb: È hemel È hemel	37) ister::Simulatio Aaster::Initialis eometry::Geon ::geometry::Geo b::geometry::di elb::geometry::	nMaster (Simulation e (SimulationMaster netryReader::LoadA ometryReader::Optin ecomposition::Optir :decomposition::Optir	nMaster.cc:63) r.cc:154) AndDecompose (C miseDomainDeco misedDecomposi timisedDecomposi	GeometryRea Omposition (G tion::Optimise sition::CallPa	der.cc:188) der.cc:188) GeometryRead edDecomposit armetis (Optim	pe: int ange: from -2147259 9/17223 processes e ler.c tion isec	9746 to -122820 equal	146 ×	
17223 17223 EParMETIS_V3_PartGeomKway (gkmetis.c:90)											
17223 17	223		libparmetis	PseudoSampleSo	rt (xyzpart.c:556)						
•											
										7353 processe	s playing
📃 Computer	🛛 🖬 ti	itan-ext7		Allinea DDT 4.2.1	-36484			()»	Su	n Aug 10, 7:50	РМ 🖾

New Bugs from Latest Changes



Arm DDT Demo



© 2018 Arm Limited

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 Overnow/Undernow Detection
		Add guard pages to detect out of bounds heap access
		Guard pages: 1 Add guard pages: After
		Advanced
Help Options	<u>R</u> un <u>Q</u> uit	Check heap consistency every
		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

Five great things to try with Allinea DDT













Arm DDT cheat sheet

Load the environment module

• \$ module load allinea-forge

Prepare the code

• \$ cc -OO -g myapp.c -o myapp.exe

Start Arm DDT in interactive mode

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

Generating Performance Reports



© 2018 Arm Limited

Profiling

Profiling is central to understanding and improving application performance.



Arm Performance Reports

High-level view of application performance shows low write rate.

	Command:	mpiexec.hydra -host node-1,node-2 -map-by socket -n 16 -ppn 8 ./Bin/low_freq///Src//hydro	1/0			
orm Performance		-ı ./Bin/low_freq////Input/input_250x125_corner.nml	A breakdown of the 16.2	2% I/O time:		
REPORTS	Resources:	2 nodes (8 physical, 8 logical cores per node)	Time in reads	0.0%		
	Tasks:	16 processes, OMP_NUM_THREADS was 1	Time in writes	100.0%		
	Machine: Start time:	node-1 Thu Jul 9 2015 10:32:13	Effective process read rate	0.00 bytes/s		
	Total time:	165 seconds (about 3 minutes)	Effective process write rate	1.38 MB/s		
	Full path:	BIN//Src				

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.

Summary: hydro is MPI-bound in this configuration



Time spent running application code. High values are usually good. This is **very low**; focus on improving MPI or I/O performance first

Time spent in MPI calls. High values are usually bad. This is **high**; check the MPI breakdown for advice on reducing it

Time spent in filesystem I/O. High values are usually bad. This is **average**; check the I/O breakdown section for optimization advice

After the fix, write rate has improved 41.6x

Eliminating file open/close bottleneck has dramatically improved I/O performance.





Time spent running application code. High values are usually good. This is **very low**; focus on improving MPI or I/O performance first Time spent in MPI calls. High values are usually bad. This is **very high**; check the MPI breakdown for advice on reducing it Time spent in filesystem I/O. High values are usually bad. This is **very low**; however single-process I/O may cause MPI wait times

arm

Performance Reports Demo



© 2018 Arm Limited

LAMMPS IO Performance Report Suggests Using MPI Profiler

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

CPU

A breakdown of the 22.1% CPU time:

Single-core code	96.2%	
OpenMP regions	3.8%	I.
Scalar numeric ops	34.4%	
Vector numeric ops	0.0%	
Memory accesses	65.6%	

Per-process performance is dominated by serial sections of computation. Use a profiler to find these or run with fewer threads and more processes.

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 76.7% MPI time:					
Time in collective calls	38.2%				
Time in point-to-point calls	61.8%				
Effective process collective rate	293 kB/s				
Effective process point-to-point rate	80.8 MB/s				

Most of the time is spent in point-to-point calls with a low transfer rate. This can be caused by inefficient message sizes such as many small messages, or by imbalanced workloads causing processes to wait.

The collective transfer rate is very low. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate.

Built-in Timers vs Arm MAP

10000		_ , 0 0	~	<u> </u>		orm FORGE	lammps	io 32.map - Arm MAP - Arm Forge 20.1	
20000	1	-2.256537	Θ	-2.182	27175	File Edit View Metrics	Window Help		
21000	1	-2.1586807	Θ	-2.084	48611	Profiled: Imp on 32 processe	s, 1 node Sampled from:	Thu Jul 16 07:55:20 2020 for 24.2s	
22000	1	-2.0923113	Θ	-2.018	84917	Main thread activity			
23000	1	-2.2482562	Θ	-2.174	44366		🔺 Assu i tradicitadicat	an a	
24000	1	-2.2003511	0	-2.126	65316				
25000	1	-2 1005479	0 O	-2 026	67283	CPU floating-point	100	,	
Loop time	e of 21.7979 o	on 32 procs fo	or 25000 steps	s with 80	9331 ato	9.2 %		بغيب والمرابعة والمرابعة والمنابعة والمرابعة والمرابعة والمحافظ والمحافظ والمحافظ والمحافظ والمحافظ والمحافظ والمحافظ	
			ľ			Memory usage	126		
Performar	nce: 297276_4($00 \pm au/day$ 11	46 900 timest	ens/s		124 MB			
94.1% CPL	J use with 32	MPI tasks x 1	OpenMP threa	ads		0 07:55:20-07:55:44 (24.231s): Main thread compute 21.2%, MPI 76.7%, File I/O 1.2%, Python interpreter 0.1%			
MPI task	timing break	down:				Missing Debugging Informat	ion 🗙	Time spent on line 92	
Section	min time	avg time	max time	%varavg	¦ %total	Missing debuggin	g information	A	
Pair	2.8332	2.9174	3.177	6.3	13.38	Normally MAP will display your been possible. There are a fe 1. You didn't compile your Normally this means pot evecutable does not c	source files here. This ti w possible explanations program with debugging assing an extra -g flag to optain debugging informs	ne, that hasn't for this: information. the compiler. If your	
MAP analy	MAP analysing program					not be able to show you your source files.			
MAP gathe	ering samples					2. Your program may have	e debugging information,		
MAP aener	rated /global/	/u2/r/rhulauir	n/cori/lammps/	'lammps i	io 32.ma	Input/Output Project Files	Main Thread Stacks	Functions	
Neigh I	0.54445	0.59053	0.73866	9.2	2.71	Total core time	▲ MPI Overhead	Function(s) on line	
Comm	3,3678	6.0525	9,6061	70.2	27.77			- LAMMPS_NS::Verlet::run(int)	
	1 776	6 1 2 1	10 5/0	105 /	2, 1,	29.4%	26.4%	✓ LAMMPS_NS::Output::write(long) LAMMPS_NS::Dump::write()	
ourpur	1.//0	0.131	10.049	103.4	20.10				

Break



© 2018 Arm Limited

Profiling with MAP



© 2018 Arm Limited

Arm MAP – The Profiler





orm

33

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

Profile of 2d Laplace Solver with Jacobi Iteration



🗴 SolvePar.c 🗶			Time spent on line 203		Ø×
8.5% <mark>bladin 1.4.</mark>	200 201 202	{	Breakdown of the 41.99	% time spent on this line:	
41.98	202 203 203	double terms = (dx + dx + dy + dy) / (z + dx + dx + z + dy + dy); qrid2[i][i] = term1 * term2; double diff = fabs(qrid2[i][i] - qrid1[i][i]);	Calling other functions	0.0%	
12.18	205 206	<pre>maxDiff = diff > maxDiff ? diff : maxDiff; }</pre>	Executing Python code	0.0%	
	207 208 209	} tempGrid = grid2; grid2 = grid1;	Scalar floating-point 9	ecuted: 1.3%	
	210 211 212	grid1 = tempGrid; tempGrid = NULL;	Vector floating point Scalar integer	0.0%	
10.9% <mark>1</mark>	213 214 215	<pre>/* Get the global MaxDiff */ MPI_Allreduce(&maxDiff, &globalMaxDiff, 1, MPI_DOUBLE, MPI_MAX, MPI_COMM_WORLD);</pre>	Vector integer	0.0%	
	213 216 217	<pre>//mr_bariet(mr_comp_mode); iterations++; //pprintf("MAX_DIFF: %f\n", maxDiff);</pre>	Branch	0.0%	

Input/Output Project Files Main Thread Stacks Functions

Main Thread Stacks					6 🗙
Total core time	A MPI	Function(s) on line	Source		Position 🔺
		😑 🥩 SolvePar [program]			
		🖻 🥖 main	{		SolvePar.c:24
41.9% and the left of the statistic statistic statistics and the statistics of the statistic statistics and the statistics of the statisti					SolvePar.c:203
12.1% <u>Hore - Hore Hildred - Anna - Hore</u>			<pre>maxDiff = diff > maxDiff ? diff : maxDiff;</pre>		SolvePar.c:205
10.9% (10.9%)	10.9%	MPI_Allreduce	MPI_Allreduce(&maxDiff, &globalMaxDiff, 1, MPI_DOUBLE, MPI_MAX, MPI_COMM_WORLD);		SolvePar.c:214
10.1%			double diff = fabs(grid2[i][j] - grid1[i][j]);		SolvePar.c:204
8.9%		WriteGridPatch	WriteGridPatch(resultDatFileName, &patchParam, grid2);		SolvePar.c:226
8.5%	-te		<pre>double term1 = (grid1[i-1][j] + grid1[i+1][j]) / (dx * dx) + (grid1[i][j-1] + grid1[i][j+1]) / (dy * dy);</pre>		SolvePar.c:201
Showing data from 4,000 samples taken over 4 processes (1000 per process)			Arm Forge 19.0 Connected to: (via tunnel) Oathkeeper:4201 -> 0	Dathkeeper 🤹 Main Thread View	

Tracking Largest Change

// Compare newly computed value with old value diff = fabs(grid2[i][j] - grid1[i][j]); // Track largest change between new and old values maxDiff = diff > maxDiff ? Diff : maxDiff;

If (diff > maxDiff)

then maxDiff= diff;

Else

```
maxDiff = maxDiff;
```

Conditional Removal from Innermost Loop



20 % faster, also operation is now vectorized

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.

• • •	😎 /Users/johlin02/OneDrive - Arm/2018/ISC/demos/florent-demos/Advanced_Demos/11_lustre/Large_hdf5.map - Arm MAP - Arm Forg	e 18.1.2
Profiled: <u>clover_leaf</u> on 32 pro	ocesses, 4 nodes, <u>32 cores (1 per process)</u> Sampled from: Tue Nov 8 2016 16:48:08 (UTC) for 335.5s	Hide Metrics
Application activity	🖌 santa sa kana kana kana kana kana kana kana	ana ala ak ka sa sa shuna ku ka ka shuna
CPU floating-point	¹⁰⁰ There are a start of the second start of	
37.8 %		
Memory usage	164	
151 MB		
10:48:08-10:53:43 (335.502s	s): Main thread compute 0.2 %, OpenMP 73.9 %, MPI 21.3 %, File I/O 1.8 %, OpenMP overhead 0.1 %, Sleeping 2.5 %	Zoom 🔍 🇮 💿
S I hydro.f90 S I visit.	L190	Time spent on line 237 O
	224 : 225 CALL hSscreate_simple_f(2, dims2d, space, hdferr) Breakdown	of the 0.3% time spent on this line:
	220 : 227 ! Create the dataset. We will use all default properties for this Everything in	structions 0.0%
	228 ! example. Calling other	functions 100.0%
	230 dataset='pres'	
	231 CALL DOCTENTE_ICTIE, GATASET, HDT_LEEF_TONLE, Space, dset, HdTerr) 232	
	233 234 1	
	235 ! Write the data to the dataset.	
0.3%	237 CALL h5dwrite f(dset, H5T NATIVE DOUBLE, chunk%tiles(tile)%field%pressure, dims2d, hdferr)	
	238 239 1	
	240 ! Close and release resources.	
	242 CALL h5dclose_f(dset , hdferr)	
	243 CALL hosciose_f(space, hdferr) 244	
	245	
	247 dims2d(l)=chunktiles(tile)t_xmax - chunktiles(tile)t_xmin + 1	
	248 dim220(2)=chunk%tile9(tile)%t_ymax - chunk%tile9(tile)%t_ymin + 1 249	
	250 ! 251 ! Create dataspace. Setting size to be the current size.	
	Input/Output Project Files OpenMP Stacks OpenMP Regions Functions	
	OpenMP Stacks	0
otal core time	Verifield Verifield Function(s) on line Source	Position
otal core time	W MPI Overhead Function(s) on line Source V7 others V7 others V7 others	Position
otal core time 0.7%	V MPI Overhead Function(s) on line Source V7 others V7 others V0.000 (chore_module::clover_allgather CALL clover_allgather (kernel_total,totals) 0.7% V0.000 (chore_module::clover_allgather CALL clover_allgather (kernel_total,totals)	Position hydro.f90:111
0.7%	V MPI Overhead Function(s) on line Source V V Tothers Visit CALL clover_allgather (kernel_total,totals) Visit Visit IF (MOD (step, visit_frequency).EQ.0) CALL visit() V Source Visit CALL b5/write f(dest.WST_NATVE_DOUBLE_churcht)	Position hydro.f90:111 hydro.f90:74
0.7% 0.1% 0.1%	V MPI Overhead Function(s) on line Source V V Visit CALL clover_allgather (kernel_total,totals) Visit Visit IF (MOD (step, visit_frequency).EQ.0) CALL visit() IF (MOD (step, visit_frequency).EQ.0) CALL visit() IF (MDD (step, visit_frequency).EQ.0) CALL visit() IF (MDD (step, visit_frequency).EQ.0) CALL visit() IF (MDD (step, visit_frequency).EQ.0) CALL visit()	Position hydro.f90:111 hydro.f90:74 le)%field visit.f90:237 le)%field visit.f90:221
0.7% 0.1% 0.1% 0.1% 	V MPI Overhead Function(s) on line Source V V Visit CALL clover_allgather (kernel_total,totals) Visit Visit IF (MOD (step, visit_frequency).EQ.0) CALL visit() V >_h5_dble_interface_MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti) >_h5_dble_interface_MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti) >_h5_dble_interface MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti))	Position hydro.f90:111 hydro.f90:74 le)%field visit.f90:237 le)%field visit.f90:321 le)%field visit.f90:265
0.7% 0.1% 0.1%	V MPI Overhead Function(s) on line Source V V others CALL clover_allgather (kernel_total,totals) Visit F(MOD (step, visit_frequency).EQ.0) CALL visit() Visit IF (MOD (step, visit_frequency).EQ.0) CALL visit() Visit Source Visit CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti Ph5_dble_interface_MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti) Ph5_dble_interface_MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti) Ph5_dble_interface_MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti)	Position hydro.f90:111 hydro.f90:74 le)%field visit.f90:237 le)%field visit.f90:321 le)%field visit.f90:265 le)%field
0.7% 0.1% 0.1% <0.1%	v MPI Overhead Function(s) on line Source v7 others v7 others 0.7% bclover_module::clover_allgather CALL clover_allgather (kernel_total,totals) vvisit IF (MOD (step, visit_frequency).EQ.0) CALL visit() b_h5_dble_interface_MOD_h5dw CALL h5dwrite_f (dset, H5T_NATIVE_DOUBLE, chunk%tiles(ti)	Position hydro.f90:111 hydro.f90:74 le)%field visit.f90:237 le)%field visit.f90:321 le)%field visit.f90:265 le)%field visit.f90:180 le)%field visit.f90:293

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 Handson



© 2018 Arm Limited

Arm MAP: Python profiling

- Launch command
 - \$ python ./laplace1.py slow 100 100
- Profiling command
 - \$ map --profile python ./laplace1.py slow 100 100
 - --profile: non-interactive mode
 - --output: name of output file
- Display profiling results
 - \$ map laplace1.map

Laplace1.py

```
[...]
err = 0.0
for i in range(1, nx-1):
    for j in range(1, ny-1):
        tmp = u[i,j]
        u[i,j] = ((u[i-1, j] + u[i+1, j])*dy2 +
            (u[i, j-1] + u[i, j+1])*dx2)*dnr_inv
        diff = u[i,j] - tmp
        err += diff*diff
return numpy.sqrt(err)
[...]
```

Naïve Python loop (laplace1.py slow 100 1000)



Optimizing computation on NumPy arrays

Naïve Python loop

```
err = 0.0
for i in range(1, nx-1):
    for j in range(1, ny-1):
        tmp = u[i,j]
        u[i,j] = ((u[i-1, j] + u[i+1, j])*dy2 +
        (u[i, j-1] + u[i, j+1])*dx2)*dnr_inv
        diff = u[i,j] - tmp
        err += diff*diff
return numpy.sqrt(err)
```

NumPy loop

```
u[1:-1, 1:-1] =
	((u[0:-2, 1:-1] + u[2:, 1:-1])*dy2 +
	(u[1:-1,0:-2] + u[1:-1, 2:])*dx2)*dnr_inv
```

return g.computeError()

NumPy array notation (laplace1.py numeric 1000 1000)

This is 10 times more iterations than was computed in the previous profile



Arm MAP cheat sheet

Load the environment module (manually specify version)

• \$ module load allinea-forge

Follow the instructions displayed to prepare the code

- \$ cc -O3 -g myapp.c -o myapp.exe
- Edit the job script to run Arm MAP in "profile" mode
- \$ map --profile -n 8 ./myapp.exe arg1 arg2

Open the results

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

Six Great Things to Try with Allinea MAP







30

! late to the party

Improve memory access







Cori Specific Settings



© 2018 Arm Limited

Configure the remote client

Install the Arm Remote Client

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>@cori.nersc.gov
 - Remote installation directory:

/usr/common/software/allinea-forge/20.1-Suse-15.0-x86_64

Examples for hands-on session

Examples are available at /global/cfs/cdirs/training/2020/arm-tools/

ddt/ddt_demo

ddt/memory_debugging

perf-report/

map/

python/

Questions?



© 2018 Arm Limited

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

