# Advanced Arm Forge for MPI Performance Engineering

John Linford <john.Linford@arm.com> 6 August 2018

Confidential © 2018 Arm Limited

### **Tutorial Schedule**

- [15] Performance Engineering: Methodology and Tools
- [15] Arm Forge Quick Start: DDT, MAP, and Performance Reports
- [30] Exercises
  - Interactive debugging
  - Profiling from the command line
  - Detect memory leaks
  - Debug invalid memory access
- [30] Break
- [20] Exercises and Examples
  - Explore I/O imbalance with MAP and performance reports
  - Real-world success story
  - Custom metrics for Lustre profiling
- [10] Q&A

## **Performance Engineering** Methodology and Tools



Confidential © 2018 Arm Limited

### Welcome to the age of machine-scale computing

It's dangerous to go alone! Take this.

#### 30 years ago: human-scale computing



Cray 2:

- 4 vector processors
- 1.9 gigaflops (9.5 mflops/Watt)

#### **Today: machine-scale computing**



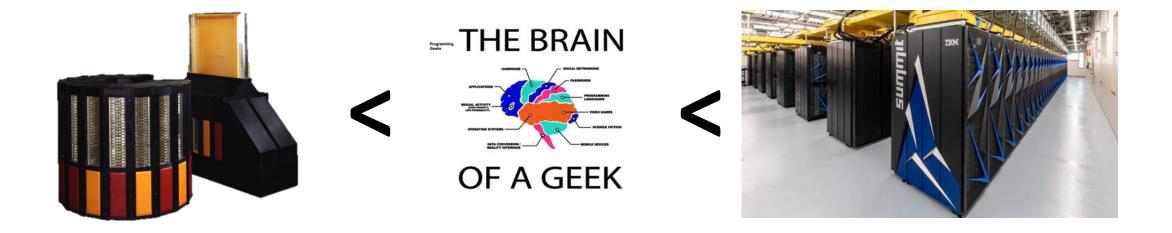
#### Summit:

- 2,282,544 cores
- 2,000,000 gigaflops (154 mflops/Watt)

### Your brain is no longer enough

No way around it, you need tools to achieve maximum performance.

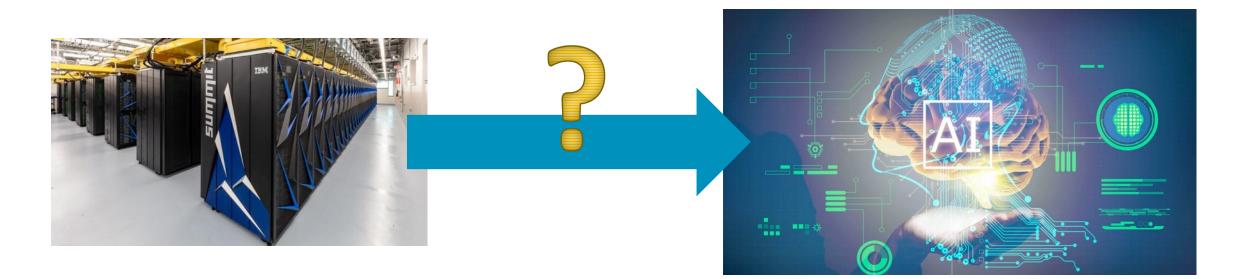
- Supercomputers are now incomprehensibly complex.
- Naïve optimization may harm performance.
- **Performance engineering tools are essential** for realizing performance at scale.



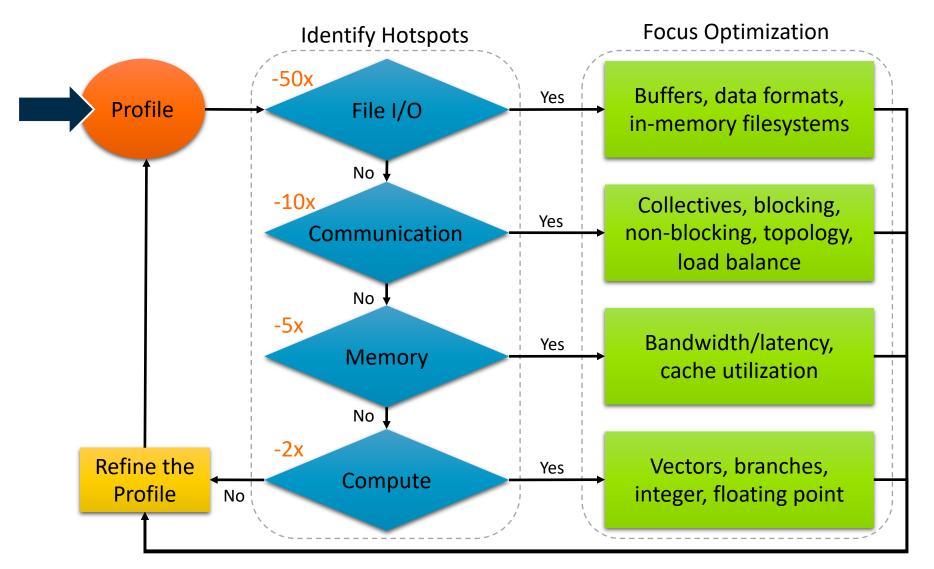
### Your brain is no longer enough

No way around it, you need tools to achieve maximum performance.

- Supercomputers are now incomprehensibly complex.
- Naïve optimization may harm performance.
- **Performance engineering tools are essential** for realizing performance at scale.



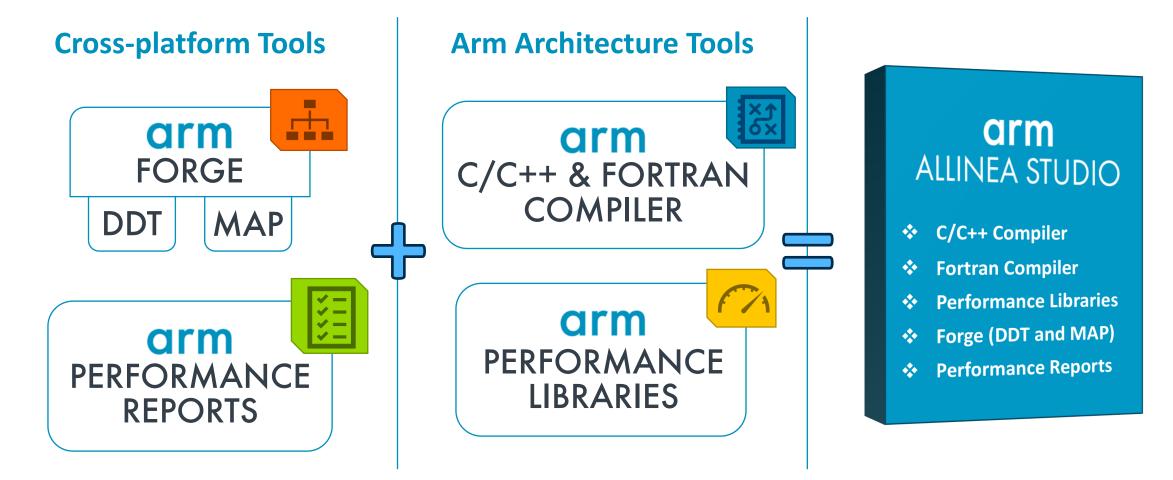
### Identifying and resolving performance issues





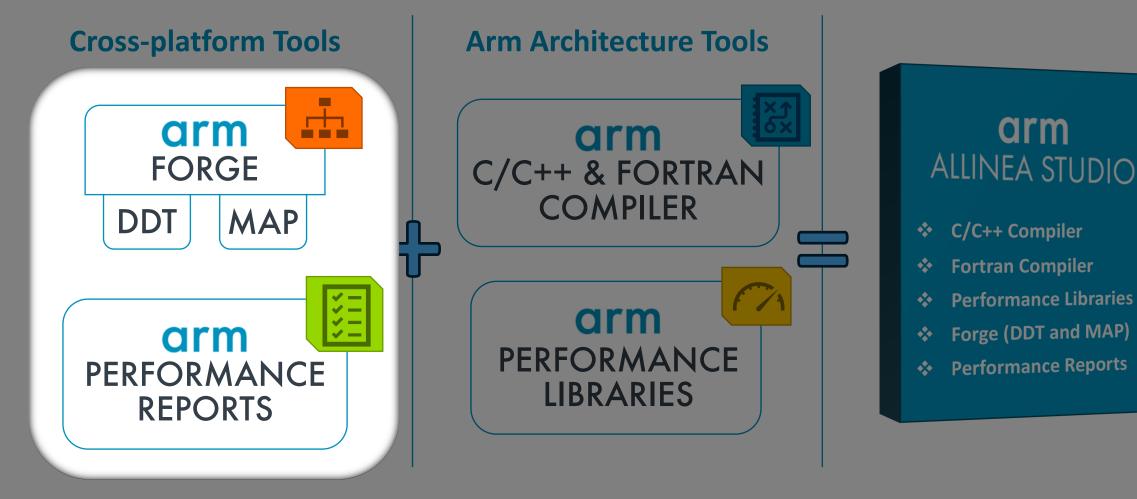
### Arm's solution for any architecture, at any scale

Commercial tools for aarch64, x86\_64, ppc64 and accelerators



### Arm's solution for any architecture, at any scale

Commercial tools for aarch64, x86\_64, ppc64 and accelerators



#### arm

### Arm Forge = DDT + MAP

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 petaflopic applications)

#### Easy to use by everyone

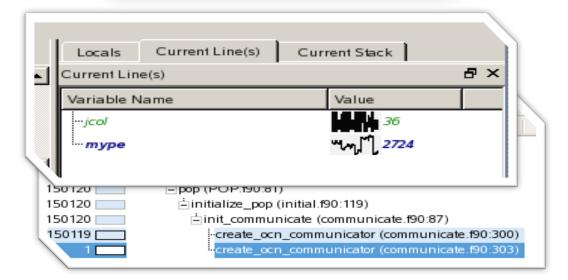
- Unique capabilities to simplify remote interactive sessions
- Innovative approach to present quintessential information to users

### **DDT: Production-scale debugging**

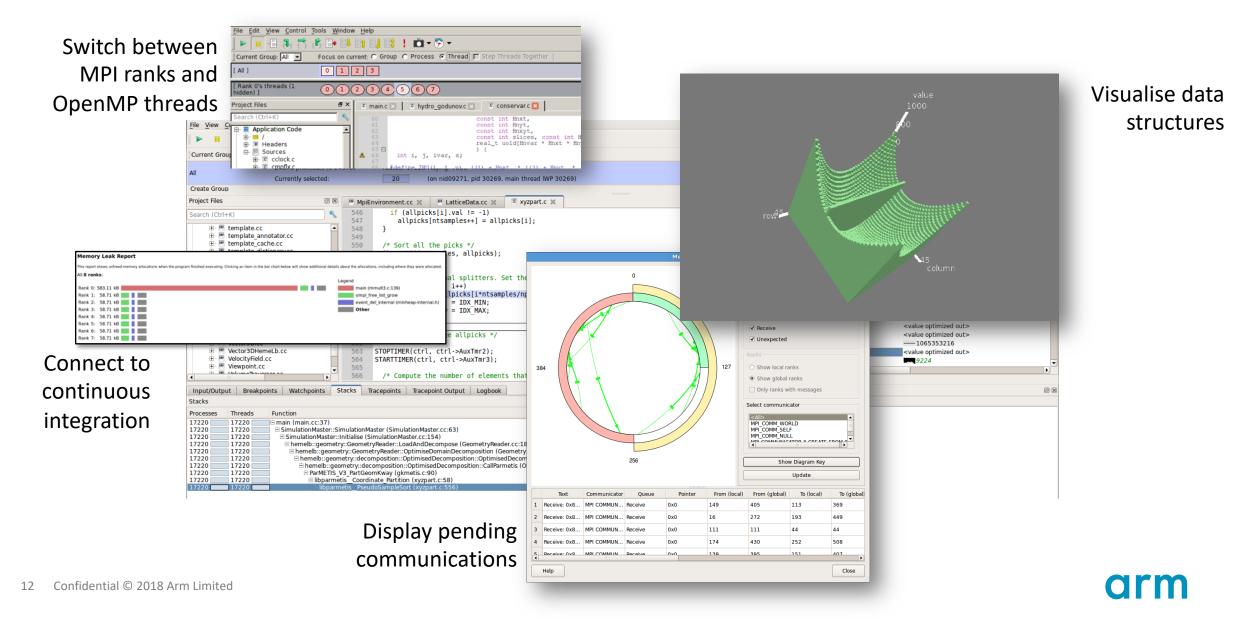
#### Isolate and investigate faults at scale

- Which MPI rank misbehaved?
  - Merge stacks from processes and threads
  - Sparklines comparing data across processes
- What source locations are related to the problem?
  - Integrated source code editor
  - Dynamic data structure visualization
- How did it happen?
  - Parse diagnostic messages
  - Trace variables through execution
- Why did it happen?
  - Unique "Smart Highlighting"
  - Experiment with variable values

			0			Display mode Process Groups	;	•
38	34	A A A A A A A A A A A A A A A A A A A				Select queues to Select queues to Select queues to Participation Selection Show local ra- Show global Only ranks w	anks	
						Select communic MPI_COMM_WO MPI_COMM_SEL MPI_COMM_NUL MPI_COMM_NUL	RLD F	FROM V
			256			All> MPI_COMM_WO MPI_COMM_SEL MPI_COMM_NUL MPI_COMM_INUL	RLD F L	=
			256			All> MPI_COMM_WO MPI_COMM_SEL MPI_COMM_NUL MPI_COMM_INUL	RLD F L	=
	Text	Communicator		Pointer	From (local)	All> MPI_COMM_WO MPI_COMM_SEL MPI_COMM_NUL MPI_COMM_INUL	RLD F L MICH 3 CREATE W Diagram Key	=
1	Text Receive: 0x8	Communicator MPI COMMUN	Queue	Pointer 0x0		All> MPI_COMM_WO MPI_COMM_SEL MPI_COMM_NUL (	RLD F L Mon 2 COLATE W Diagram Key Update	- FRAM (*
1 2		MPI COMMUN	Queue Receive		From (local)	All> MPI_COMM_WO MPI_COMM_SEL MPI_COMM_NUL All_COMM_NUL All_COMM_NUL Sho	RLD F L w Diagram Key Update To (local)	To (globa
	Receive: 0x8	MPI COMMUN MPI COMMUN	Queue Receive Receive	0x0	From (local) 149	CAILS MP_COMM_WO MP_COMM_SLU MP_COMM_SLU MP_COMMUNIC (1) Sho From (global) 405	RLD F L w Diagram Key Update To (local) 113	To (globa 369
2	Receive: 0x8 Receive: 0x8	MPI COMMUN MPI COMMUN MPI COMMUN	Queue Receive Receive Receive	0x0 0x0	From (local) 149 16	CAIIS MPI_COMM_WO MPI_COMM_SLL MPI_COMM_UNIC ( Sho From (global) 405 272	RLD F LD. 2 COLATE W Diagram Key Update To (local) 113 193	To (globa 369 449



#### **DDT: Feature Highlights**



### **Multi-dimensional Array Viewer**

What does your data look like at runtime?

- View arrays
  - On a single process
  - Or distributed on many ranks
- Use metavariables to browse the array
  - Example: \$i and \$j
  - Metavariables are unrelated to the variables in your program.
  - The bounds to view can be specified
  - Visualise draws a 3D representation of the array
- Data can also be filtered
  - "Only show if": \$value > 0 for example \$value being a specific element of the array

Multi-Dimensi	ional Array Viewer	×
ay Expression: [tables[\$i][\$j]	~	<u>E</u> valuate
tributed Array Dimensions: None 🚔 How do I	view distributed arrays?	Cancel
Staggered Array What does this do?		✓ Align Stack Frames
ange of \$i Range of \$j		Auto-update
rom: 0 🕈 From: 0	•	_
o: 11 ♣ <u>T</u> o: 11		
Display: Rows 🗢 Display: Column	ns 🗘	
Only show if:	See Examples	
ata Ta <u>b</u> le <u>S</u> tatistics		
🗦 Goto 🛛 🖗 Visualize 🛛 📔 Export 🛛 🖸	Full Window	
i		
0 1 2 3 4 5 6		
0         1         2         3         4         5         6         7           1         2         4         6         8         10         12         14	8 9 10 11 12 16 18 20 22 24	
2 3 6 9 12 15 18	10 10 20 22 24	
3 4 8 12 16 20 24		
4 5 10 15 20 25 30		6
5         6         12         18         24         30         36           6         7         14         21         28         35         42		s de la constante de la consta
7 8 16 24 32 40 48		
8 9 18 27 36 45 54		
9 10 20 30 40 50 60		and the second
10 11 22 33 44 55 66	row	
Help		
		45 column
_		

### **MAP: Production-scale application profiling**

Identify bottlenecks and rewrite code for better performance

- Run with the representative workload you started with
- Measure all performance aspects with Arm Forge Professional

#### **Examples:**

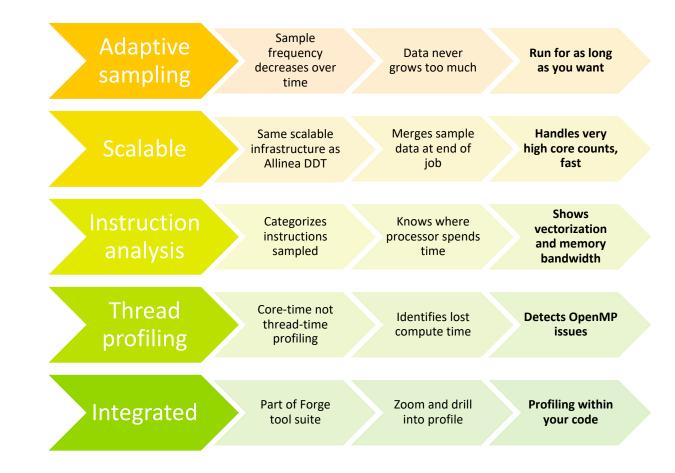
\$> map -profile mpirun -n 48 ./example

Application activity metrophysical activity Herations / s <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup> <sup>200</sup>	e na na herrau a del na na na na na hadi da ana da da ana da da ana da sa
Iterations / s	1
	والمراجعة والمستقبل والمستعد العمالية والمستعد والمستقبل والمستقبل والمستعد
Grind time 175 00 0.00 s	
Step time 0.00 s	alle de la contra d
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.190 ×	Time spent on line 75 III III III III III III III III III
3.28         73         CALL flux_calc()           51.29         75         CALL divection()           3.38         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  A MPI Overhead Function(s) on line	-

Profiled: My_code.exe on 64 p	rocesses Started: Fri Sep 20 14:59:09 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 (%) 0 - 90 (8.2 av 14:59:09-14:59:44 (range 3	g )	Metrics, Reset
7 My_code.f90 🔀		
	87 88 ⊞ module wall excitation [⊓]	
	100   101   102   103   104   105   10	
	102 103 ⊕ module derivativee	
	141 ! MAIN CODE	
	142	
	144 use data mc	
	145 use wall_excitation	
	146 implicit none 147 include 'mpif.h'	
	148 double precision :: max_omx_dt,max_omy_dt,max_omz_dt,t,time_cal	
	149 integer :: option, i, j, k, nn, fwcnt, count_max, counter, ios, next_file_at, W_cnt(1:4)	
	<pre>150 character*30 :: str,file_type,str_t,num_2_str 151</pre>	
0.1%	151 call MPI INIT(ierr)	
	153 call MPI COMM SIZE (MPI_COMM_WORLD, npro, ierr)	
nput/Output Project Files	Parallel Stack View	
allel Stack View		
tal Time	V MPI Function(s) on line Source Posi	ition
	evel vort 3d fp , <un 3d="" fp="" my<="" program="" td="" vel="" vort=""><td>code.f90:143</td></un>	code.f90:143
63.0%	31.4% Bitme integration call time integration My	code.f90:330
16.9% mml		code.f90:297 code.f90:337
1.8%	<pre>@<unknown> <unknown> (no debug info)</unknown></unknown></pre>	
4.1%	1.4% #velvort_3d_fp_ call cell_identifier My_	code.f90:190
4.170 mm	1.379 B 31 UNIELS	

### **How MAP is different**

MAP's flagship feature is lightweight, highly scalable performance profiling



### **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

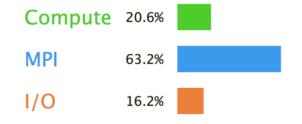
#### Adds value to typical users' workflows

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

### **Arm Performance Reports**

#### A high-level view of application performance with "plain English" insights

<b>Orm</b> PERFORMANCE REPORTS	Command: Resources: Memory: Tasks: Machine: Start time: Total time: Full path:	mpiexec.hydra -host node-1,node-2 -map-by socket -n 16 -ppn 8 ./Bin/low_freq///Src//hydro -i ./Bin/low_freq////Input/input_250x125_corner.nml 2 nodes (8 physical, 8 logical cores per node) 15 GiB per node 16 processes, OMP_NUM_THREADS was 1 node-1 Thu Jul 9 2015 10:32:13 165 seconds (about 3 minutes) Bin//Src	I/O A breakdown of the 16.2 Time in reads Time in writes Effective process read rate Effective process write rate	% I/O time: 0.0% 100.0% 0.00 bytes/s 1.38 MB/s
Summary	: hydro i	s MPI-bound in this configuration	effective transfer rate. This	write operations with a very low may be caused by contention for the ess patterns. Use an I/O profiler to are affected.

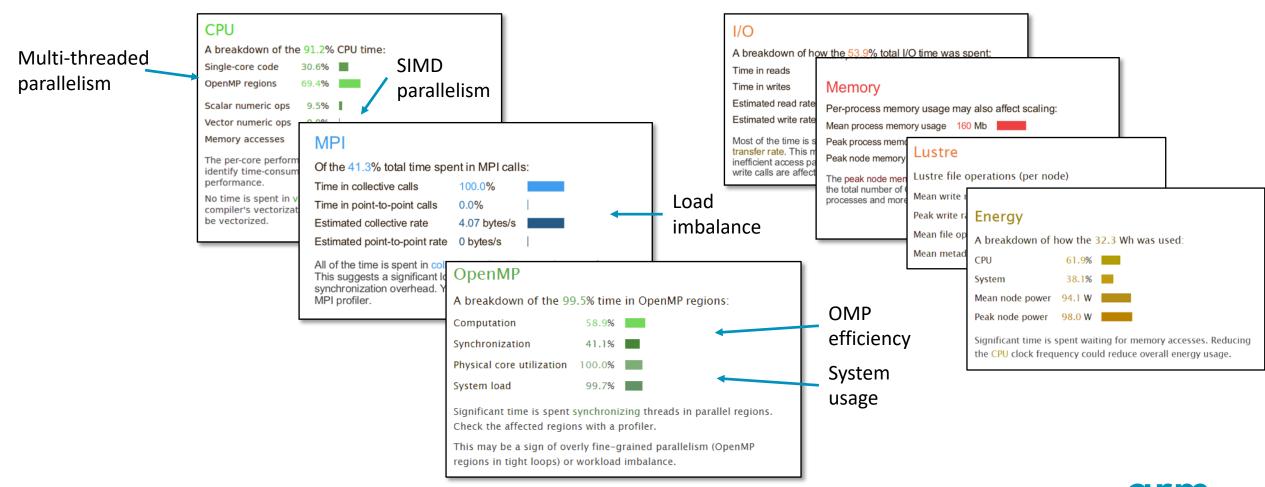


Time spent running application code. High values are usually good. This is **very low**; focus on improving MPI or I/O performance first Time spent in MPI calls. High values are usually bad. This is **high**; check the MPI breakdown for advice on reducing it Time spent in filesystem I/O. High values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

This is **average**; check the I/O breakdown section for optimization advice

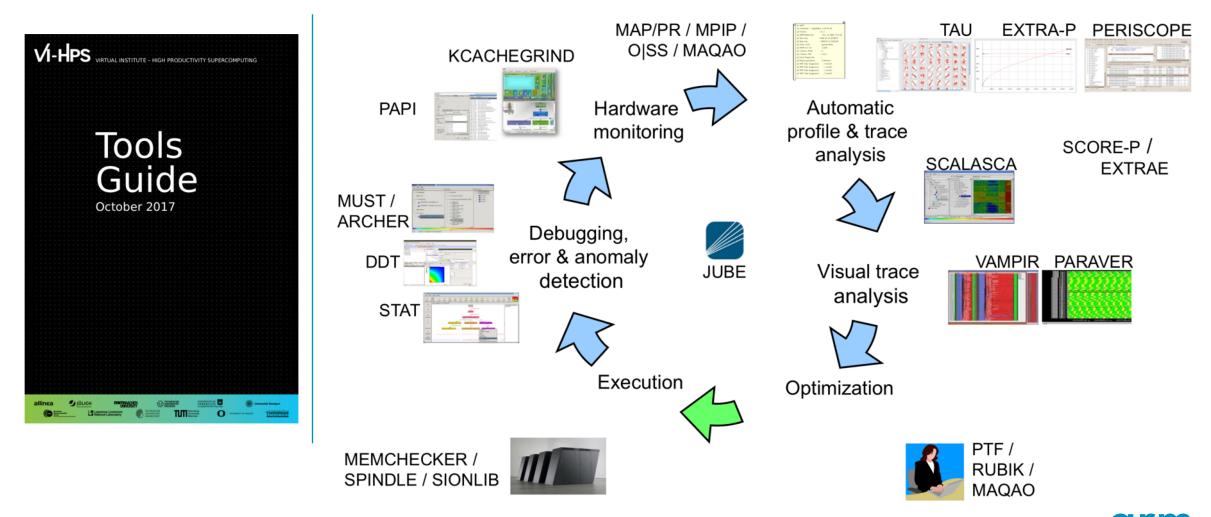
### **Arm Performance Reports Metrics**

Lowers expertise requirements by explaining everything in detail right in the report.



### **VI-HPS and the tools ecosystem**

See the http://www.vi-hps.org/tools/ for an excellent view of the tools ecosystem.



# **Arm Forge Quick Start**

#### **Tool cheat sheets**

Confidential © 2018 Arm Limited



### **Arm DDT cheat sheet**

#### Start DDT interactively, remotely, or from a batch script.

- Load the environment module:
  - \$ module load forge
- Prepare the code:
  - \$ mpicc -OO -g myapp.c -o myapp.exe
  - \$ mpfort -**OO** -g myapp.f -o myapp.exe
- Start DDT in interactive mode:
  - \$ **ddt** mpirun -n 8 ./myapp.exe arg1 arg2 ...
- Or use reverse connect:
  - On the login node:
    - \$ ddt &
  - (or use the remote client)
  - Then, edit the job script to run the following command and submit:
    - **ddt --connect** mpirun -n 8 ./myapp.exe arg1 arg2 ...

### **Run DDT in offline mode**

Run the application under DDT and halt or report when a failure occurs.

- You can run the debugger in non-interactive mode
  - For long-running jobs
  - For automated testing, continuous integration...
- To do so, use the following arguments:
  - \$ ddt --offline --output=report.html mpirun ./jacobi\_omp\_mpi\_gnu.exe
    - --offline enable non-interactive debugging
    - --output specifies the name and output of the non-interactive debugging session
      - Html
      - Txt
    - Add --mem-debug to enable memory debugging and memory leak detection

#### **DDT command line options**

\$ ddt --help

```
Arm Forge 18.2.1 - Arm DDT
```

```
Usage: ddt [OPTION...] [PROGRAM [PROGRAM_ARGS]]
ddt [OPTION...] (mpirun|mpiexec|aprun|...) [MPI_ARGS] PROGRAM [PROGRAM_ARGS]
```

```
--connect
```

```
--attach=[host1:]pid1,[host2:]pid2... [PROGRAM]
--attach-mpi=MPI_PID [--subset=rank1,rank2,rank3,...] [PROGRAM]
--break-at=LOCATION[,START:EVERY:STOP] [if CONDITION]
--trace-at=LOCATION[,START:EVERY:STOP],VAR1,VAR2,...
--cuda
--mem-debug[=(fast|balanced|thorough|off)]
--mpiargs=ARGUMENTS
-n, --np, --processes=NUMPROCS
--nodes=NUMNODES
--procs-per-node=PROCS
--offline
```

-s, --silent

Reverse Connect (launch as a server and wait) attach to PROGRAM being run by list of host:pid attach to processes in an MPI program. set a breakpoint at LOCATION set a tracepoint at LOCATION enable CUDA configure memory debugging (defaults to fast) command line arguments to pass to mpirun specify the number of MPI processes configure the number of nodes for MPI jobs configure the number of processes per node run through program without user interaction don't write unnecessary output to the command line

### **Arm MAP cheat sheet**

Generate profiles and view offline

- Load the environment module
  - \$ module load forge
- Prepare the code
  - \$ mpicc -OO -g myapp.c -o myapp.exe
  - \$ mpfort -OO -g myapp.f -o myapp.exe
- Offline: edit the job script to run Arm MAP in "profile" mode
  - \$ map --profile mpirun ./myapp.exe arg1 arg2
- View profile in MAP:
  - 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 command line options**

\$ map --help Arm Forge 18.2.1 - Arm MAP

Usage: map [OPTION...] [PROGRAM [PROGRAM ARGS]] map [OPTION...] (mpirun|mpiexec|aprun|...) [MPI ARGS] PROGRAM [PROGRAM ARGS] map [OPTION...] [MAP FILE]

--connect

Analysis of the CUDA kernel source code lines --cuda-kernel-analysis --list-metrics Display metrics IDs which can be explicitly enabled or disabled. --disable-metrics=METRICS Explicitly disable metrics specified by their metric IDs. --enable-metrics=METRICS Explicitly enable metrics specified by their metric IDs. --export=FILE.json Exports a specified .map file as JSON --export-functions=FILE Export all the available columns in the functions view to a CSV file (use --profile) --select-ranks=RANKS Select ranks to profile. --mpiargs=ARGUMENTS command line arguments to pass to mpirun -n, --np, --processes=NUMPROCS specify the number of MPI processes --nodes=NUMNODES configure the number of nodes for MPI jobs --procs-per-node=PROCS configure the number of processes per node run through program without user interaction

Reverse Connect (launch as a server and wait for the GUI to connect)

--profile

### **Arm Performance Reports cheat sheet**

#### Generate text and HTML reports from application runs or MAP files

- Load the environment module:
  - \$ module load reports
- Run the application:
  - **perf-report** mpirun -n 8 ./myapp.exe
- ... or, if you already have a MAP file:
  - **perf-report** myapp\_8p\_1n\_YYYY-MM-DD\_HH:MM.txt
- Analyze the results
  - \$ cat myapp\_8p\_1n\_YYYY-MM-DD\_HH:MM.txt
  - \$ firefox myapp\_8p\_1n\_YYYY-MM-DD\_HH:MM.html

#### **Performance Reports command line options**

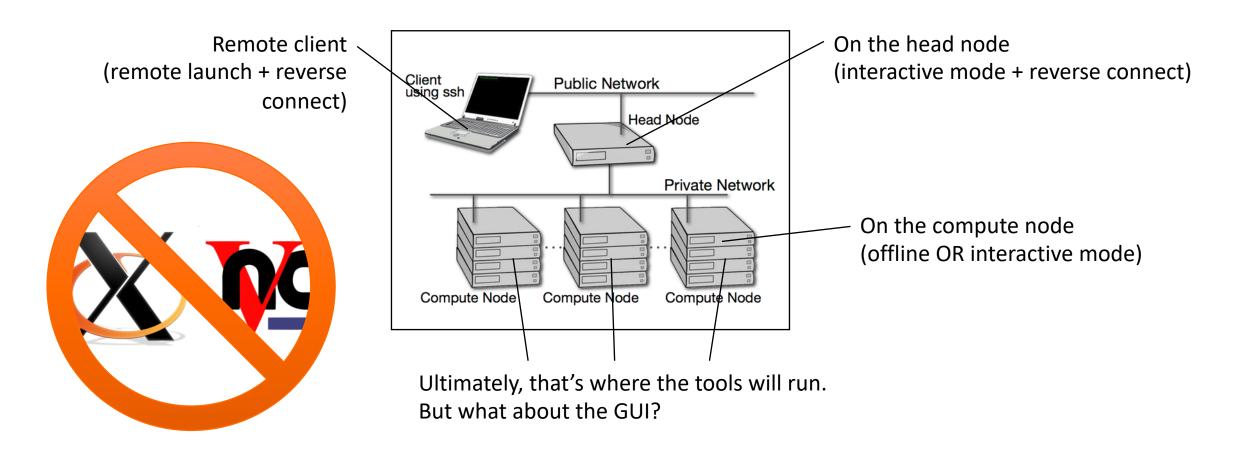
\$ perf-report --help
Arm Performance Reports 18.2.1 - Arm Performance Reports

```
Usage: perf-report [OPTION...] PROGRAM [PROGRAM_ARGS]
perf-report [OPTION...] (mpirun|mpiexec|aprun|...) [MPI_ARGS] PROGRAM [PROGRAM_ARGS]
perf-report [OPTION...] MAP_FILE
```

list-metrics	Display metrics IDs which can be explicitly enabled or disabled.
<pre>disable-metrics=METRICS</pre>	Explicitly disable metrics specified by their metric IDs.
<pre>enable-metrics=METRICS</pre>	Explicitly enable metrics specified by their metric IDs.
mpiargs=ARGUMENTS	command line arguments to pass to mpirun
nodes=NUMNODES	configure the number of nodes for MPI jobs
-o,output=FILE	writes the Performance Report to FILE instead of an auto-generated name.
-n,np,processes=NUMPROCS	specify the number of MPI processes
<pre>procs-per-node=PR0CS</pre>	configure the number of processes per node for MPI jobs
<pre>select-ranks=RANKS</pre>	Select ranks to profile.

### The Forge GUI and where to run it

DDT and MAP provide powerful GUIs that can be run in a variety of configurations.



### Launching the Forge Remote Client

#### The remote client is a stand-alone application that runs on your local system

#### Install the Arm Remote Client (Linux, macOS, Windows)

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

#### Connect to the cluster with the remote client

- Open Forge Remote Client
- Create a new connection: Remote Launch → Configure → Add
  - Hostname: <username>@<hostname>
  - Remote installation directory: </path/to/arm-forge/X.Y/>
- Connect!

### Arm Forge 18.1.2 and MVAPICH2

- To use DDT's memory debugging features, set the environment variable MV2\_ON\_DEMAND\_THRESHOLD to the maximum job size you expect. This setting should not be a system wide default; it should be set as needed.
- To use mpirun\_rsh with DDT, from File → Options go to the System page, check Override default mpirun path and enter mpirun\_rsh. You should also add – hostfile <hosts>, where <hosts> is the name of your hosts file, within the mpirun\_rsh arguments field in the Run window.
- To enable message Queue Support MVAPICH 2 must be compiled with the flags --enable-debug --enable-sharedlib. These are not set by default.
- MVAPICH2 MPI programs cannot be started using Express Launch syntax.
  - Do use: "ddt ./a.out" and configure MPI launch parameters in the GUI.
  - Don't use: "ddt mpirun <mpi\_args> ./a.out"

# Interactive Debugging

#### **Crash and hang**

arm

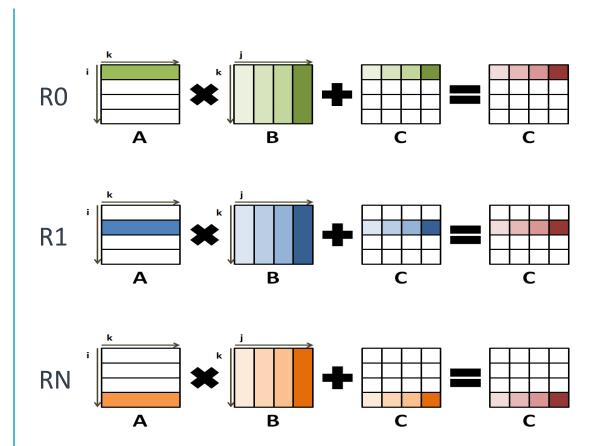
Confidential © 2018 Arm Limited

### $\mathbf{C} = \mathbf{A} \times \mathbf{B} + \mathbf{C}$

Simply multiply and add two matrices

#### Algorithm

- 1. Rank 0 (R0) initialises matrices A, B & C
- 2. R0 slices the matrices A & C and sends them to Rank 1...N (R1+)
- 3. R0 and R1+ perform the multiplication
- 4. R1+ send their results back to R0
- 5. R0 writes the result matrix C to file



### Fix a simple crash in a MPI code

Simple matrix multiply and add? No problem! Except that it crashes...

#### **Exercise Outline**

- Objectives
  - Discover Arm DDT's interface
  - Interactively debug a crash in a MPI application

#### Commands

- \$ make
- \$ mpirun -np 4 ./mmult1\_c.exe
- # Observe crash
- \$ ddt ./mmult1\_c.exe
- # Observe cause of crash

#### **Initial Result: Crash!**

🕒 🗊 johlin02@johlin02-VM: ~/MUG18/01\_walkthrough/1\_crash johlin02@johlin02-VM:~/MUG18/01\_walkthrough/1\_crash\$ make mpicc -q -ffast-math -OO -DDEBUG -std=c99 mmult1.c -o mmult1 c.exe -lm mpif90 -g -ffast-math -O0 -DDEBUG -cpp mmult1.f90 -o mmult1\_f90.exe -lm johlin02@johlin02-VM:~/MUG18/01\_walkthrough/1\_crash\$ mpirun -np 4 ./mmult1 c.exe 0: Size of the matrices: 64x64 0: Initializing matrices... 0: Sending matrices... 0: Processina... [johlin02-VM:mpi rank 0][error sighandler] Caught error: Segmentation fault (signal 11) 3: Receiving matrices... 2: Receiving matrices... 1: Receiving matrices... 2: Processing... [johlin02-VM:mpi rank 2][error sighandler] Caught error: Segmentation fault (signal 11) 1: Processing... [johlin02-VM:mpi rank 1][error sighandler] Caught error: Segmentation fault (signal 11) BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES PID 9160 RUNNING AT johlin02-VM EXIT CODE: 139 CLEANING UP REMAINING PROCESSES YOU CAN IGNORE THE BELOW CLEANUP MESSAGES

YOUR APPLICATION TERMINATED WITH THE EXIT STRING: Segmentation fault (signal 11) This typically refers to a problem with your application. Please see the FAQ page for debugging suggestions johlin02@johlin02-VM:~/MUG18/01\_walkthrough/1\_crash\$ \$

### **Answer: Fix incorrect limits on k-loop**

Incorrect limits lead to invalid memory access

#### Before

164	<pre>do i=0,size/nslices-1</pre>
165	do j=0,size-1
166	res=0.0
167	<pre>do k=size,size*size</pre>
168	res=A(i*size+k)*B(k*size+j)+res
169	end do
170	C(i*size+j)=res+C(i*size+j)
171	end do
172	end do

#### After

164	<pre>do i=0,size/nslices-1</pre>
165	do j=0,size-1
166	res=0.0
167	do k=0,size-1
168	res=A(i*size+k)*B(k*size+j)+res
169	end do
170	C(i*size+j)=res+C(i*size+j)
171	end do
172	end do



### **Answer: Fix incorrect limits on i-loop**

Incorrect limits on i-loop lead to unmatched MPI\_Send

#### Before

73 do i=1,nproc-2

- 74 call MPI\_Send(mat\_a(slice\*i), slice, & MPI\_DOUBLE, i, 100+i, & MPI\_COMM\_WORLD, ierr)
- 75 call MPI\_Send(mat\_b, size\*size, & MPI\_DOUBLE, i, 200+i, & MPI\_COMM\_WORLD, ierr)

77 end do

#### After

	(i) clico
<pre>74 call MPI_Send(mat_a(slice&gt; MPI_DOUBLE, MPI_COMM_WOB</pre>	i, <mark>100+</mark> i, &
<pre>75 call MPI_Send(mat_b, size&gt; MPI_DOUBLE, MPI_COMM_WOB</pre>	i, <mark>200+</mark> i, &
<pre>76 call MPI_Send(mat_c(slice&gt; MPI_DOUBLE, MPI_COMM_WOF</pre>	i, <mark>300+</mark> i, &

77 end do

## Improve performance

### **Efficient memory access**

Confidential © 2018 Arm Limited



## Fix inefficient memory access pattern

It works! But wow it's slow.

### **Exercise Outline**

- Objectives
  - Discover Arm MAP's interface
  - Gather initial profiles of a MVAPICH2 application

#### Commands

\$ make

- \$ map --profile -n 4 \
   ./mmult2\_f90.exe
- \$ map mmult2\_f90\_4p\*.map
- # Observe profile

### **Initial Result: SLOW**

😕 🗇 💷 johlin02@johlin02-VM: ~/MUG18/01\_walkthrough/2\_memory\_accesses

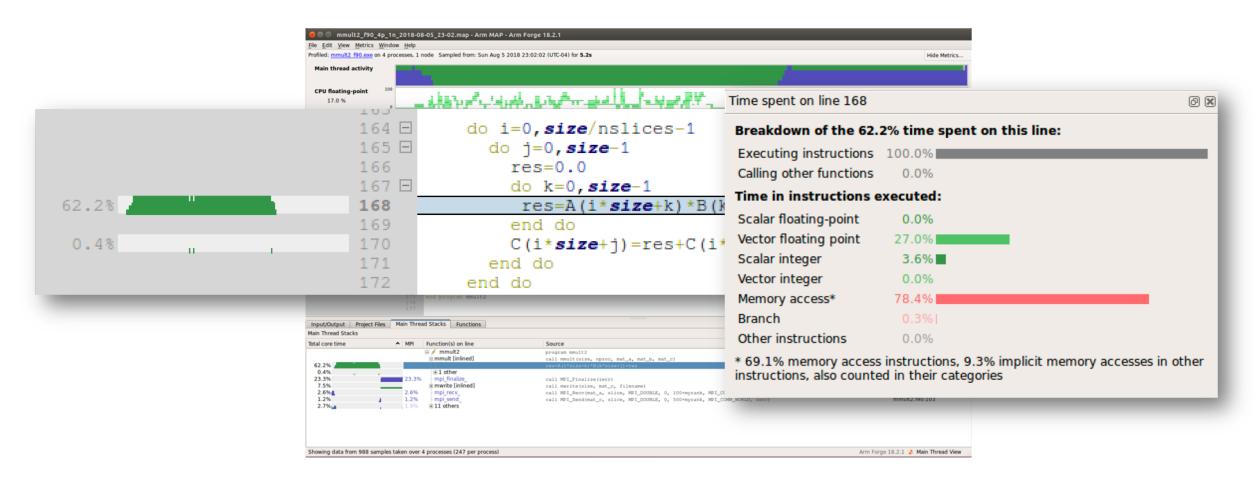
johlin02@johlin02-VM:~/MUG18/01\_walkthrough/2\_memory\_accesses\$ map --profile -n 4 ./mmult2\_f90.exe
Arm Forge 18.2.1 - Arm MAP

```
Profiling
                     : /home/johlin02/MUG18/01_walkthrough/2_memory_accesses/mmult2_f90.exe
Allinea sampler
                     : not preloading
MPI implementation : Auto-Detect (MVAPICH 2)
 number of processes : 4
  number of nodes
                  : 1
 Allinea MPI wrapper : not preloading
          1 : Receiving matrices...
          0 : Size of the matrices:
                                            1024 x
                                                          1024
          2 : Receiving matrices...
          3 : Receiving matrices...
          0 : Initializing matrices...
          0 : Sending matrices...
          1 : Processing...
           2 : Processing...
              Processing...
              Processing...
          2 : Sending result matrix...
          1 : Sending result matrix...
          3 : Sending result matrix...
          0 : Receiving result matrix...
          0 : Writing results...
          0 : Done.
MAP analysing program...
MAP gathering samples...
```

MAP gathering samples... MAP generated /home/johlin02/MUG18/01\_walkthrough/2\_memory\_accesses/mmult2\_f90\_4p\_1n\_2018-08-05\_23-0 2.map johlin02@johlin02-VM:~/MUG18/01\_walkthrough/2\_memory\_accesses\$

## **Initial profile**

#### Find the hotspot: look for the line with the highest core time.



## **Memory access patterns**

- Data locality
  - Temporal locality: use of data within a short time of its last use
  - Spatial locality: use memory references close to memory already referenced

```
Temporal locality example
for (i=0 ; i < N; i++) {
   for (loop=0; loop < 10; loop++) {</pre>
```

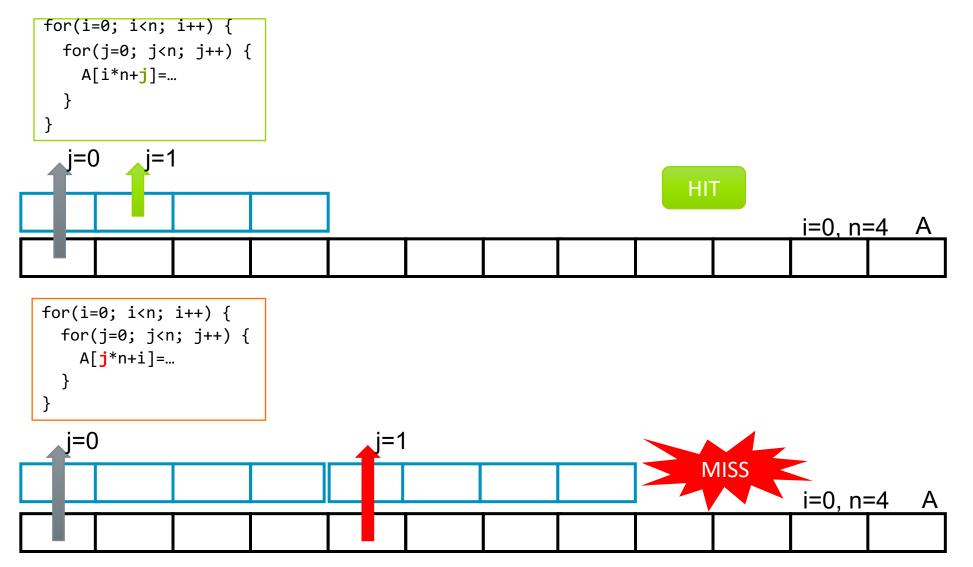
... = ... x[i] ...

```
Spatial locality example
```

```
for (i=0 ; i < N*s; i+=s) {
    ... = ... x[i] ...
}</pre>
```



## **Memory Accesses and Cache Misses**



## **Answer: Transpose matrix and interchange loops**

Transposing the matrix improves locality  $\rightarrow$  performance

#### Before

164	<pre>do i=0,size/nslices-1</pre>
165	do j=0,size-1
166	res=0.0
167	do k=0,size-1
168	res=A(i*size+k)*B(k*size+j)+res
169	end do
170	C(i*size+j)=res+C(i*size+j)
171	end do
172	end do

## After

165	<pre>do i=0,size/nslices-1</pre>
166	do j=0,size-1
167	res=0.0
168	do k=0,size-1
169	res=A(i*size+k)*transB(j*size+k)+res
170	end do
171	C(i*size+j)=res+C(i*size+j)
172	end do
173	end do



# **Final profile**

About 3x faster

## Before



## After



arm

# Leak Detection ... and DDT in Offline Mode

arm

Confidential © 2018 Arm Limited

# **Possible memory leak**

Transpose is working great, but sometimes I run out of memory?

#### **Exercise Outline**

- Objectives
  - Use DDT in offline mode
  - Explore DDT's report logbook

#### Commands

- \$ make
- \$ ddt --offline \
   --output=report.html \
   -n 4 \
   ./mmult3\_f90.exe
- \$ xdg-open report.html
- # Observe report

## DDT in offline mode (--offline)

johlin02@johlin02-VM: ~/MUG18/01_walkthrough/3_memory_leak johlin02@johlin02-VM:~/MUG18/01_walkthrough/3_memory_leak\$ ddtofflineoutput=report.htmlmem- debug -n 4 ./mmult3_f90.exe Arm Forge 18.2.1 - Arm DDT				
MPI implementation * number of processes * number of nodes Memory debugging enabled * setting	: /home/johlin02/MUG18/01_walkthrough/3_memory_leak/mmult3_f90.exe : Auto-Detect (MVAPICH 2) : 4 : 1 : Yes : Fast : Fast : Off			
0 : Size of the 0 : Initializi 1 : Receiving f 2 : Receiving f 0 : Sending ma 1 : Processing 2 : Processing 3 : Processing 3 : Processing 1 : Sending re: 0 : Receiving 3 : Sending re: 0 : Writing re: 0 : Writing re: 0 : Done.	ng matrices matrices matrices trices trices    sult matrix sult matrix sult matrix sult matrix			

johlin02@johlin02-VM:~/MUG18/01\_walkthrough/3\_memory\_leak\$ xdg-open report.html



## **DDT Debugging Report**

## Use DDT's reporting feature to debug long-running applications

I report logbook × John	Se 💿 🕐 report logbook 🗙 🚬 John
← → C 🛈 file:///home/johlin02/MUG18/01_walkthrough/3_memory_leak/report.html#leaks 🔄 🖕 🚥 🗉 🔅 🔾 📕 💿 🗄	← → C 🛈 file:///home/johlin02/MUG18/01_walkthrough/3_memory_leak/report.html#leaks 🔄 🔮 🔤 💿 🔅 🔾 📓 💿 :
🖿 Arm 🌾 Hal Higdon Train 📴 Mail - John. Linfo 📴 Calendar - John 🛛 😤 Mission Controlo 😤 Issue Navigatoro 📄 🗈 Other bookmarks	🖿 Arm 🦸 Hal Higdon Train 📴 Mail - John. Linfo 🔯 Calendar - John. 😨 Mission Control - 😨 Issue Navigator - 📄 Other bookmarks
A	
report logbook	Messages Tracepoints Memory Leak Report Output
Debugging /home/johlin02/MUG18/01_walkthrough/3_memory_leak/mmult3_f90.exe	Tracepoints
Messages Tracepoints Memory Leak Report Output	No tracepoints set or hit.
Messages	Messages Tracepoints Memory Leak Report Output
[+] Expand All [-] Collapse All	Memory Leak Report
# Type Time Processes Message	This report shows unfreed memory allocations when the program finished executing. Clicking an item in the bar chart below will show additional details about the allocations, including where they were allocated.
1       1	All 4 ranks: Legend
2 1 0:02.621 0-3 Startup complete.	Rank 0: 151.18 MB mmult3 (mmult3.f90:62)
3 0:02.623 n/a Select process group All	Rank 1: 173.39 kB Other
4 Additional Information	Rank 2: 173.39 kB Rank 3: 173.39 kB
▼ Stacks	
Processes         Function         Source         Variables           mmult3 (mmult3.f90:17)         call MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)         Rank 0, thread 1	Allocation data can also be exported to CSV format.
► Current Stack	
5 (1) 0:04.809 n/a Debugging : /home/johlin02/MUG18/01_walkthrough/3_memory_leak/mmult3_f90.exe MPI implementation : Auto-Detect (MVAPICH 2)	Messages Tracepoints Memory Leak Report Output
* number of processes : 4 * number of nodes : 1 Memory debugging enabled : Yes	Output
* setting : Fast * check bounds : Off	0 : Size of the matrices : 3072 x 3072 0 : Initializing matrices 1 : Receiving matrices
6 🕨 0:04.809 0-3 Play	2 : Receiving matrices 3 : Receiving matrices θ : Sending matrices 1 : Processing 2 : Processing
7         O         0:42.396         0-3         Program stopped at exit.	0 : Processing 3 : Processing 1 : Sending result matrix 2 : Sending result matrix 0 : Receiving result matrix 3 : Sending result matrix
8 Additional Information	θ : Writing results θ : Done.



# View the memory leak report to see unfreed allocations

Allocations that are not freed when the program exits *could* be leaks

## **Click allocation to see function source**



Source			
<pre>allocate(mat_b(0:size*size-1))</pre>			
59.	if(myrank==0) then		
60.	allocate(mat a(0:size*size-1))		
61.	allocate(mat_b(0:size*size-1))		
62.	allocate(mat c(0:size*size-1))		
63.			
	<pre>print *,myrank,": Initializing matrices"</pre>		

#### **Review source code to verify leak**

← → C () file:///home/johlin02/MUG18/01_walkthro	ugh/3_memory_leak/report.html#leaks 📩 🛧 🐢 🔤 🤃 🔅 📿 📶 🌔	0
🖿 Arm 🦸 Hal Higdon Train 🔯 Mail - John Linfo 🧕	🛛 Calendar - John 🛛 🏆 Mission Control 📲 Issue Navigator -	kmar
Memory Leak Report		
This report shows unfreed memory allocations when the program f where they were allocated.	inished executing. Clicking an item in the bar chart below will show additional details about the allocations, inclu	ding
All <b>4 ranks</b> :		
	Legend	
Rank 0: 151.18 MB	mmult3 (mmult3.f90:62)	
Rank 1: 173.39 kB	Other	
Rank 2: 173.39 kB		
Rank 3: 173.39 kB		
	2)] on Irank (I)-	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6	2)] on [rank 0]:	
Allocation data can also be <u>exported to CSV format</u> . Largest <b>allocation call path</b> at [mmult3 (mmult3,f90;6 1 unfreed allocation (75.50 MB in total) <b>Function</b>	Source	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 1 unfreed allocation (75.50 MB in total)		
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 1 unfreed allocation (75.50 MB in total) Function	<pre>Source v allocate(mat_b(0:size*size-1)) 59. if(myrank==0) then</pre>	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 1 unfreed allocation (75.50 MB in total) Function	<pre>Source v allocate(mat_b(0:size*size-1)) 59. if(myrank==0) then 60. allocate(mat_a(0:size*size-1)) 61. allocate(mat b(0:size*size-1))</pre>	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 1 unfreed allocation (75.50 MB in total) Function	<pre>Source v allocate(mat_b(0:size*size-1)) 59. if(myrank=0) then 60. allocate(mat_a(0:size*size-1)) 61. allocate(mat_b(0:size*size-1)) 62. allocate(mat_c(0:size*size-1)) 62. allocate(mat_c(0:size*size-1)) 63. allocate(mat_c(0:size*size-1)) 64. allocate(mat_c(0:size*size-1)) 65. allocate(mat_c(0:size*size+size-1)) 65. allocate(mat_c(0:size*size-1)) 65. allocate(mat_c(0:size*size-1)) 65. allocate(mat_c(0:size*size-1)) 65. allocate(mat_c(0:size*size-1)) 65. allocate(mat_c(0:size*size+size+1)) 65. allocate(mat_c(0:size*size+size+size+1)) 65. allocate(mat_c(0:size*size+size+1)) 65. allocate(mat_c(0:size*size+size+size+size+size+size+size+size+</pre>	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 1 unfreed allocation (75.50 MB in total) Function	<pre>Source v allocate(mat_b(0:size*size-1)) 59. if(myrank==0) then 60. allocate(mat_a(0:size*size-1)) 61. allocate(mat b(0:size*size-1))</pre>	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 L unfreed allocation (75.50 MB in total) <b>Function</b> #0 mmult3 (mmult3.f90:62)	<pre>Source v allocate(mat_b(0:size*size-1)) 59. if(myrank==0) then 60. allocate(mat_a(0:size*size-1)) 61. allocate(mat_b(0:size*size-1)) 62. allocate(mat_c(0:size*size-1)) 63.</pre>	
Largest <b>allocation call path</b> at [mmult3 (mmult3.f90:6 1 unfreed allocation (75.50 MB in total) Function	<pre>Source v allocate(mat_b(0:size*size-1)) 59. if(myrank==0) then 60. allocate(mat_a(0:size*size-1)) 61. allocate(mat_b(0:size*size-1)) 62. allocate(mat_c(0:size*size-1)) 63. 64. print *,myrank,": Initializing matrices"</pre>	

# **Memory Debugging** Allocation tracking and guard pages

Confidential © 2018 Arm Limited



## Three levels of heap debugging overhead



#### 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 allocated/freed has been overwritten.

check-blank

#### check-funcs

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

See user-guide: Chapter 12.3.2

## arm

# **Tri-diagonal solve: segmentation fault**

Crashing with invalid memory reference. Sounds like a job for a memory debugger!

#### **Exercise Outline**

- Objectives
  - Use DDT's memory debugging features
  - Use guard pages to find out-of-bounds access
- Commands
  - \$ make
  - \$ ddt -n 4 ./trisol.exe
  - # Enable fast memory debugging
  - # Do <u>not</u> enable guard pages

#### **Invalid memory access**

interfact in the second state of the seco

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error:

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error:

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error:

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error: ^C[mpiexec@johlin02-VM] Sending Ctrl-C to processes as requested [mpiexec@johlin02-VM] Press Ctrl-C again to force abort johlin02@johlin02-VM:~/MUG18/03\_mem\_debugging\$

## DDT's heap memory debugging framework

#### Dynamically linked binaries

- LD\_PRELOAD is usually used automatically
- Not on static binaries, not on all Crays or old SLURMs

#### Statically linked binaries

- If not, manual linking is required
- LFLAGS = -dynamic -L/path/to/forge/lib/64/ -zmuldefs -Wl,--undefined=malloc -ldmalloc

Run				
Run: mpirun -n 8 ./mmult2_c.exe	Details			
Command: mpirun -n 8 ./mmult2_c.exe				
□ OpenMP	Details			
CUDA: Track allocations: enabled, Detect invalid accesses: disabled				
<ul> <li>Track GPU allocations (also enables CPU memory debugging)</li> <li>Detect invalid accesses (memcheck)</li> </ul>				
Memory Debugging: Fast, 1 guard page after, Backtraces, Preload	Details			
Plugins: none	Details			
Help Options	<u>R</u> un Quit			

#### When manual linking is used, untick "Preload" box

<u>         P</u> reload the mem	ory debugging library <u>L</u> an	guage: C++, thread	ts 🗧			
	nly works for programs lin atically linked, you must re					
Fast	Balanced	Thorough	Custom			
<u> </u>	basic	More I	nformation			
Heap Overflow/Underflow Detection						
$\blacksquare$ Add guard pages to detect out of bounds heap access						
Guard pages: 1 Add guard pages: After +						
Ad <u>v</u> anced						
<u>Check heap cons</u>	sistency every 100 📮 he	eap operations				
Store stack <u>b</u> acktraces for memory allocations						
$\Box$ <u>O</u> nly enable for these processes:						
0	100% Se	elect All x2 x0	.5 1%			
Help		O	Cancel			

# It works in DDT????

The code appears to run fine when launched from the debugger! Why?

### **DDT launch configuration**

🛛 🗊 Run				
Application: /home/johlin02/MUG18/03_mem_de				
Application: /home/johlin02/MUG18/03_mem_d	ebugging/trisol.exe 👻 📹			
Arguments:				
stdin file:	v 🖻			
Working Directory:	😕 💿 Memory Debugging Options			
MPI: 4 processes, MVAPICH 2	Preload the memory debugging library Langu	age: C++, threads		
Number of Processes: 4	Note: Preloading only works for programs linked a program is statically linked, you must relink it aga manually.			
mpirun arguments	Heap Debugging Fast Balanced	Thorough Custom		
OpenMP				
CUDA	Enabled Checks: basic	More Information		
Memory Debugging: Fast, No guard pages,	Heap Overflow/Underflow Detection			
Submit to Queue	Add guard pages to detect out of bounds he	ap access		
Environment Variables: none	Guard pages: 1 Add guard pages: A	fter 🗘		
Plugins: none	Advanced			
	Check heap consistency every 100	heap operations		
	Store stack backtraces for memory allocation			
	Only enable for these processes:			
Help Options	0-3 100% Select	All x2 x0.5 1%		
	Help	OK Cancel		

#### Uh oh, program output looks great

#### It should have crashed! What changed?

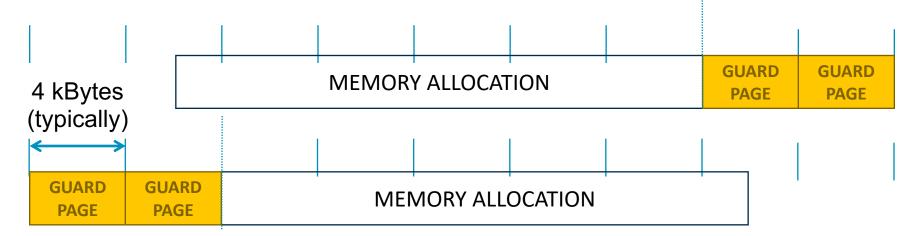
Input/Output	Breakpoints	Watchpoints	Stacks	Tracepoints	Tracepoint Output	Logbook
Input/Output						
<pre>*** Solution correct  x  / (sqrt(n)*epsilon*( A * x  +  b ) = 6.6245D-07</pre>						
😣 All processes finished.						

Yes

No

Every process in your program has terminated - would you like to restart this session from the beginning?

## **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

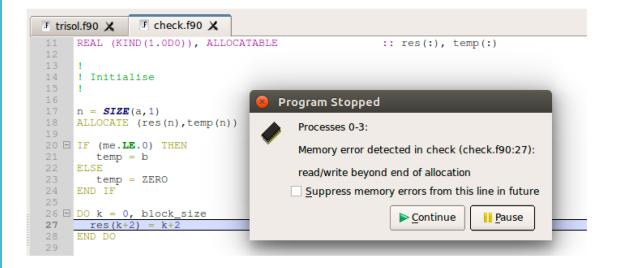
## **OK, this time enable guard pages**

The code appears to run fine when launched from the debugger! Why?

## Add one guard page after every allocation

😂 🗉 Memory Debugging Options					
✓ <u>P</u> reload the mem	nory debugging library <u>L</u> angua	ge: C++, threads	÷		
	nly works for programs linked ag y linked, you must relink it again				
Fast	Balanced	Thorough	Custom		
Enabled Checks:	basic	More I	nformation		
✓ Add guard pages to detect out of bounds heap access          Guard pages:       1       ♣         Add guard pages:       Add guard pages:       After					
✓ Store stack base	onsistency every 100 – ho acktraces for memory allocations r these processes: 100% Select A		1%		
Help OK Cancel					

## Gotcha! Write OOB at res(k+2)



# Debugging Imbalance MPI I/O

Confidential © 2018 Arm Limited



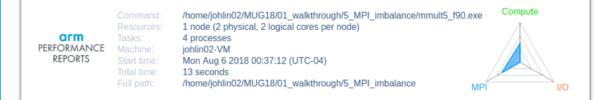
# Can we improve I/O performance?

R0 responsible for all file I/O after R1+ return results. Surely we can do better?

#### **Exercise Outline**

- Objectives
  - Use MAP's I/O profiling features
  - Use performance reports to quantify speedup
- Commands
  - \$ make
  - \$ map --profile -n 4 \
     ./mmult5\_f90.exe
  - \$ perf-report mmult5\_f90\_4p\*.map
  - \$ xdg-open mmult5\_f90\_4p\*.html

#### **Performance report shows MPI bound**



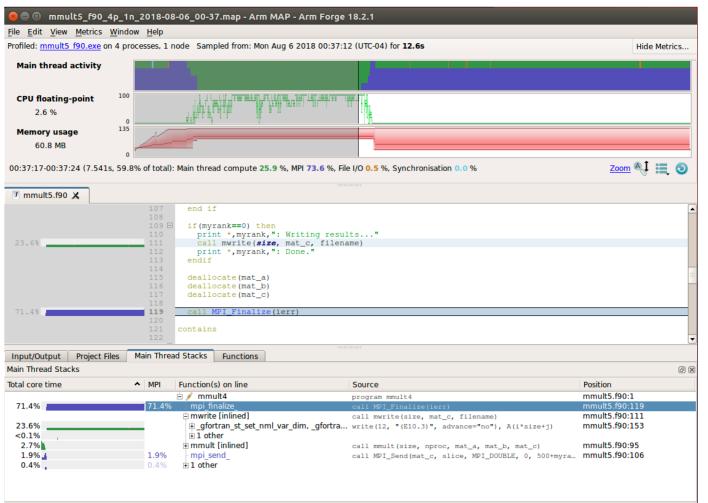
#### Summary: mmult5\_f90.exe is MPI-bound in this configuration

Comput	e 46.7%	Time spent running application code. High values are usually good. This is <b>low</b> ; consider improving MPI or I/O performance first
MPI	53.0%	Time spent in MPI calls. High values are usually bad. This is <b>high</b> ; check the MPI breakdown for advice on reducing it
I/O	0.3%	Time spent in filesystem I/O. High values are usually bad. This is <b>very low</b> ; however single-process I/O may cause MPI wait times
This applica	tion run was MPI-bound. A breat	kdown of this time and advice for investigating further is in the MPI section below.

55 Confidential © 2018 Arm Limited

## Initial profile shows MPI\_Finalize dominates

#### Time spent in MPI\_Finalize is due to load imbalance in file I/O



Showing data from 1,404 samples taken over 4 processes (351 per process)

Arm Forge 18.2.1 🥵 Main Thread View

# **Answer: improve scalability of I/O routines**

Use MPI-IO to let all MPI ranks write their results to file simultaneously.

#### Before

97 if(myrank==0) then

```
100 do i=1,nproc-1
```

102 end do

103 else

107 end if

```
109 if(myrank==0) then
```

```
111 call mwrite(size, mat_c, filename)
```

113 endif

## After

102	<pre>call MPI_FILE_OPEN(MPI_COMM_WORLD, &amp;     filename, &amp;     MPI_MODE_CREATE+MPI_MODE_WRONLY, &amp;     MPI_INF0_NULL, fh, ierr)</pre>
103	<pre>call MPI_FILE_SET_VIEW(fh, &amp;</pre>
104	<pre>call MPI_FILE_WRITE_AT(fh, disp, mat_c, &amp;     slice, MPI_DOUBLE, st, ierr)</pre>
105	<pre>call MPI_BARRIER(MPI_COMM_WORLD, ierr)</pre>
106	<pre>call MPI_FILE_CLOSE(fh, ierr)</pre>

## New approach: use MPI-IO for file output

Each MPI rank writes its results to it's own part of the output file

#### **Before: runtime 13 seconds**

	COIII
	Reso
arm	Task
PERFORMANCE	Mach
REPORTS	Start
	Total

Ind: /home/johlin02/MUG18/01\_walkthrough/5\_MPI\_imbalance/mmult5\_f90.exe 1 node (2 physical, 2 logical cores per node) 4 processes iohlin02-VM Mon Aug 6 2018 00:37:12 (UTC-04) 13 seconds h: /home/johlin02/MUG18/01 walkthrough/5 MPI imbalance



#### Summary: mmult5\_f90.exe is MPI-bound in this configuration

Compute	46.7%	Time spent running application code. High values are usually good. This is <b>low</b> ; consider improving MPI or I/O performance first
MPI	53.0%	Time spent in MPI calls. High values are usually bad. This is <b>high</b> ; check the MPI breakdown for advice on reducing it
I/O	0.3%	Time spent in filesystem I/O. High values are usually bad. This is <b>very low</b> ; however single-process I/O may cause MPI wait times

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

### After: runtime 5 seconds (2.6x speedup)

erm PERFORMANCE REPORTS PERFORMANCE REPORTS Command: Resources Tasks: Machine: Start time: Total time: Full path: /home/johlin02/MUG18/01\_walkthrough/5\_MPI\_imbalance/solution/mmult6\_f90.Exempute 1 node (2 physical, 2 logical cores per node) 4 processes johlin02-VM Mon Aug 6 2018 00:34:17 (UTC-04) 5 seconds /home/johlin02/MUG18/01\_walkthrough/5\_MPI\_imbalance/ solution MPI

#### Summary: mmult6\_f90.exe is Compute-bound in this configuration

Compute	74.5%	
MPI	20.1%	
I/O	5.3%	

Time spent running application code. High values are usually good. This is **high**; check the CPU performance section for advice

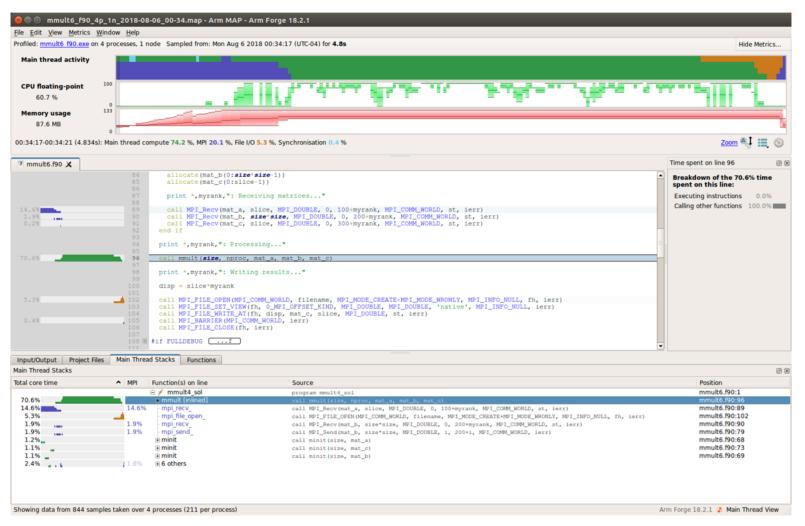
Time spent in MPI calls. High values are usually bad. This is **low**; this code may benefit from a higher process count

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

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.

## Final profile shows balanced I/O and compute dominates

#### New approach is about 3x faster



arm

# Success at Scale

**Curtin Quantum Collisions** 

arm

Confidential © 2018 Arm Limited

# CCC and the ORNL GPU Hackathon @ Pawsey

### Quantum collisions in atomic and molecular physics

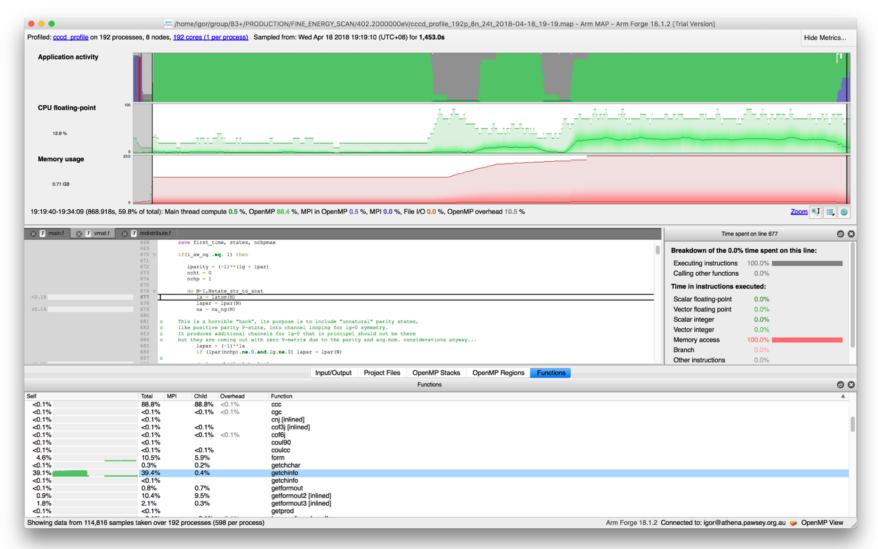
- CCC: Quantum mechanics
  - Fusion energy
  - Laser science
  - Lighting industry
  - Medical imaging / therapy
  - Astrophysics
- Igor Bray, Head of Physics and Astronomy, and the Theoretical Physics Group, in the Faculty of Science and Engineering, at Curtin University







## **Initial profile at production scale**



## Load balancer is imbalanced?

Customized load balancing algorithm wasn't delivering expected results

0	8	0	-10	199	329	492	1.21	13530	0	89	-1	<b>91</b> %
LG,noc	de,ip	bar,	inc,vt,	,i1,i2,	tperi,	nch,nap	s,mt,p	rev LG,eff				
1	8	0	-7	591	573	872	1.97	45150	0	350	0	80%
LG,noc	de,ip	bar,	inc,vt,	,i1,i2,	tperi,	nch,nap	s,mt,p	rev LG,eff				
2	8	0	-16	894	762	1153	2.28	77028	0	607	1	86%
LG,noc	de,ip	bar,	inc,vt,	,i1,i2,	tperi,	nch,nap	s,mt,p	rev LG,eff				
3	8	0	-24	916	886	1331	2.05	99681	0	766	2	<b>91</b> %
LG,noc	de,ip	bar,	inc,vt,	,i1,i2,	tperi,	nch,nap	s,mt,p	rev LG,eff				

## "That makes no sense!"

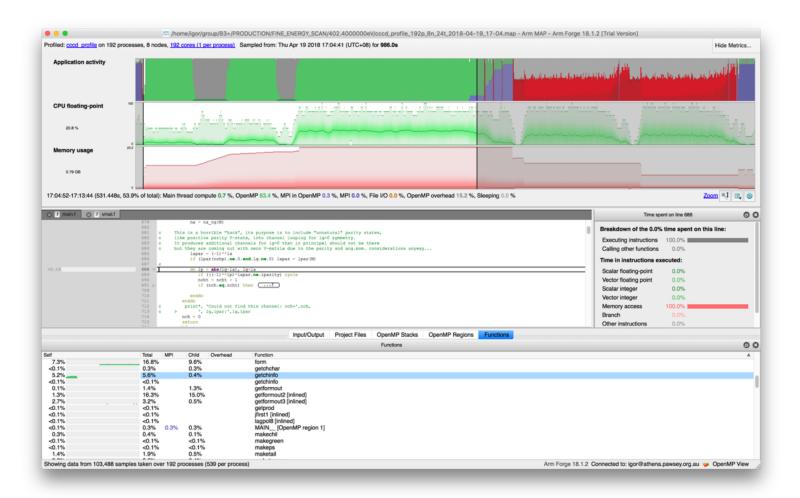
#### Computing one grid point takes as much time as computing the entire grid

Self	Total	MPI	Child	Overhead	Function
<0.1%	88.8%		88.8%	<0.1%	CCC
<0.1%	<0.1%		<0.1%	<0.1%	cgc
<0.1%	<0.1%				cnj [inlined]
<0.1%	<0.1%		<0.1%		cof3j [inlined]
<0.1%	<0.1%		<0.1%	<0.1%	cof6j
<0.1%	<0.1%				coul90
<0.1%	<0.1%		<0.1%		coulcc
4.6%	10.5%		5.9%		form
<0.1%	0.3%		0.2%		getchchar
39.1%	. 39.4%		0.4%		getchinfo
<0.1%	<0.1%				getchinfo
<0.1%	0.8%		0.7%		getformout
0.9%	10.4%		9.5%		getformout2 [inlined]
1.8%	2.1%		0.3%		getformout3 [inlined]
<0.1%	<0.1%				getprod

## Surprise! Didn't expect that.

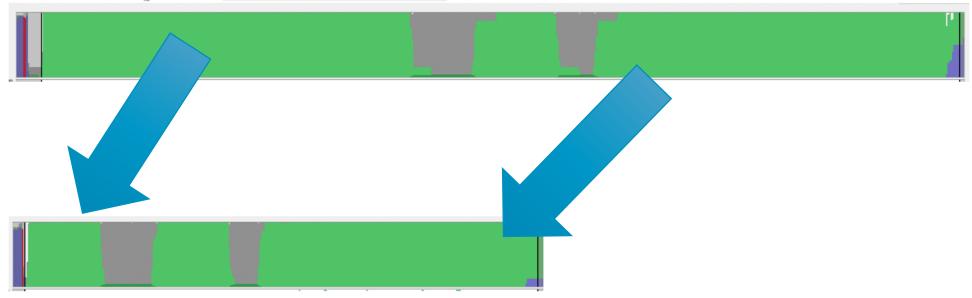
## Final profile, again at production scale

### Found an unbounded array copy a(:) that should have been a(1:N)



## **Before and after**

4.6%	10.5%	5.9%	form
<0.1%	0.3%	0.2%	getchchar
39.1%	. 39.4%	0.4%	getchinfo
<0.1%	<0.1%		getchinfo
<0.1%	0.8%	0.7%	getformout



Self	Total MPI	Child Overhead	Function
7.3%	16.8%	9.6%	form
<0.1%	0.3%	0.3%	getchchar
5.2%	5.6%	0.4%	getchinfo
<0.1%	<0.1%		getchinfo
0.1%	1.4%	1.3%	getformout

## **Balanced the load balancer**

Load can be balanced mow that work blocks are of expected sizes

## Before:

0	8	0 -10	199	329	492	1.21	13530	0	89	-1	<b>91</b> % LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff
1	8	0 -7	591	573	872	1.97	45150	0	350	0	80% LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff
2	8	0 -16	894	762	1153	2.28	77028	0	607	1	<pre>86% LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff</pre>
3	8	0 -24	916	886	1331	2.05	99681	0	766	2	<b>91</b> % LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff

#### After:

0	8	0 -10	174	329	492	1.06	13530	0	85	-1	<b>93</b> % LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff
1	8	0 -11	415	577	872	1.40	43956	0	340	0	<b>97</b> % LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff
2	8	0 -11	616	757	1153	1.55	79003	0	592	1	<b>97</b> % LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff
3	8	0 -12	667	874	1331	1.46 1	105111	0	734	2	<b>96</b> % LG,node,ipar,inc,vt,i1,i2,tperi,nch,naps,mt,prev LG,eff

# **Custom metrics for Lustre** Combine I/O performance data from system and application

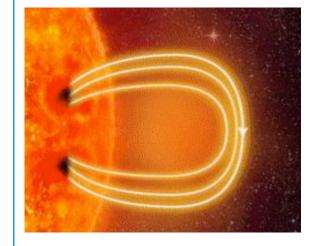
arm

Confidential © 2018 Arm Limited

# **Advanced I/O investigation of Lustre on Archer**

Simultaneously view system-level and application-level performance.

- Show data from Lustre client logs along with application data
- iPIC3D: kinetic simulation of plasma
  - Fully 3D implicit particle-in-cell (PIC)
  - C++ and MPI
  - Intermediate simulation results saved in VTK binary files, single file per quantity
  - Checkpointing done through HDF5 to individual files per process
  - Field values saved using collective MPI-IO to single file



## **Available performance data**

Use MAP's ability to measure filesystem performance at the system and application levels

#### System level performance data

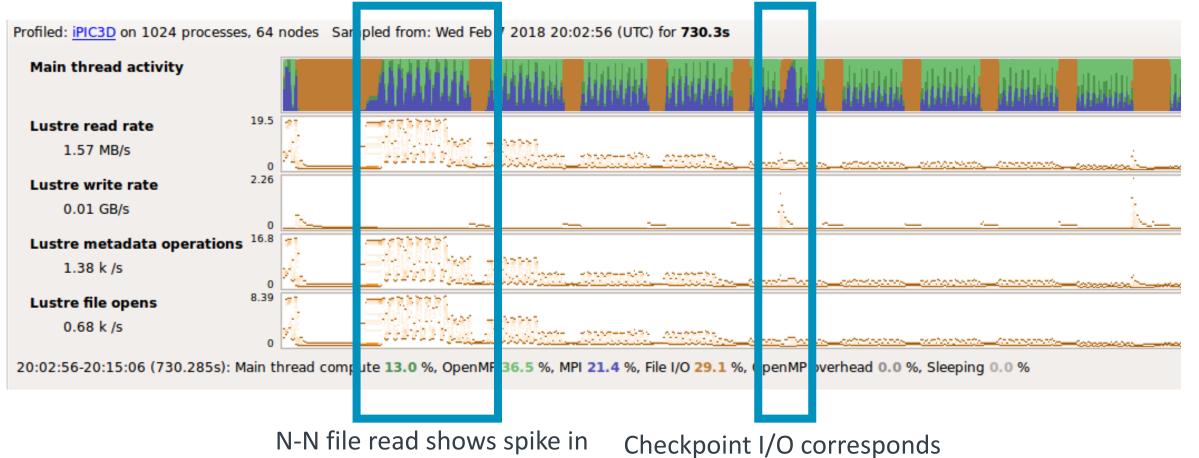
- Lustre logs: each read, write, or metadata operation recorded from each Lustre client.
- Aggregate I/O data for precise bandwidth figures for read/write at any moment in time.
- Max/min/mean bandwidth.
- Scheduler logs: application run start and end time and assigned nodes.

#### **Application level performance data**

- Approximate I/O bandwidth in a timeline.
- Approximate classification of I/O instructions (methods).
- In block-synchronous approach, it is possible to identify different I/O phases.

# MAP aligns the system timeline with the application timeline

#### Lustre data is read from the lustre client's log files, while application data is read directly.



file open/read operations.

to spike in Lustre write rate

# We can focus on each I/O operation individually

#### Select a portion of the application timeline to view the source code performing I/O.

<u>File Edit View Metrics Win</u> d	low <u>H</u> elp												
Profiled: iPIC3D on 1024 proces	ses, 64 nodes Sampled from	: Wed Feb	7 2018 20:0	2:56 (UTC) for <b>730.3s</b>									
Main thread activity			NUMPR	NAME AND A	ARI DIR ANDRES	ADDINIO DINIONI		loviller att	lul lulli				ulun lunilul
Lustre read rate 1.78 MB/s	19.5	Myz :				alatha bata damarana			· · · · · · · · · · · · · · · · · · ·				
Lustre write rate 0.00 GB/s	2.26										i.		
Lustre metadata operation 1.61 k /s		H <u>m</u>		anani <mark>a manta</mark>									
Lustre file opens 0.80 k /s	8.39		1910) <u></u>										
20:05:26-20:05:41 (15.336s, 2	.1% of total): Main thread con	npute 9.2	8 %, OpenMP	21.4 %, MPI <b>18.7</b> %, File	e I/O <mark>50.7</mark> %, OpenMP over	head 0.0 %, Sleeping	0.0 %						
Input/Output	Project Files	Ma	in Thre	ad Stacks	Functions								Time
Main Thread Sta	:ks												
Total core time		*	MPI	Overhead	Function(s)	on line							
					🗄 🥩 iPIC3D	[program]							
					🗄 🥖 main								
					🖻 iPic3D::	c_Solver::W	riteOutp	ut(int)					
43.4%					🖻 Writel	MomentsVTk	K(Grid3D	CU*, EN	/fields3	D*, Colle	ective*	, VCtopo	ol
7.4%					🕀 Write	ieldsVTK(Gr	rid3DCU*	*, EMfiel	ds3D*,	Collectiv	ve*, VC	topolog	C3D.cpp C3D.cpp
0.4%			0.2%		🕀 1 othe	er							n/iPIC3D1ib n/iPIC3D1ib
20.6%			3.1%		🗄 iPic3D::	c_Solver::Pa	rticlesMo	over()					C3D.cpp
15.5%			9.1%		🗄 iPic3D::	c_Solver::Ca	alculateF	ield(int)					C3D.cpp C3D.cpp
12.3%			6.0%		🗄 iPic3D::	c_Solver::Ca	alculateM	Ioments	5()				
0.3%			0.3%		🗄 2 other	-							

# MAP's timeline shows I/O overlapping with communication

#### We see elevated Lustre write rate when writing checkpoint restart files in HDF5.

<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>M</u> etrics <u>W</u> indo	ow <u>H</u> elp				
Profiled: <u>iPIC3D</u> on 1024 processe	es, 64 nodes Sampled from	m: Wed Feb 7 2018 20:0	02:56 (UTC) for <b>730.</b>	3s	
Main thread activity		ALIANTIK ATATATAK	HAMMAN ANNI	ht M <mark>alanda kundu dulah dulah dulah dulah badua balua balua balua dalah ba</mark>	ulla limiter limiter
Lustre read rate 1.04 MB/s	19.5	ini <u>ni mu</u>			
Lustre write rate 0.05 GB/s	2.26				~ ~ ~
Lustre metadata operation 0.96 k /s	A	M <u>m mun</u>	12000001200 00000020		
Lustre file opens 0.38 k /s	8.39				
Input/Output	Project Files	Main Threa	d Stacks	Functions	
Main Thread Stack	ks				Time
Total core time		▲ MPI	Overhead	Function(s) on line	
				🖻 🛸 iPIC3D [program] 🖻 🥖 main	
37.8% 💶		30.7%		iPic3D::c_Solver::CalculateField(int)	
28.3%	_	4.7%	<0.1%	iPic3D::c_Solver::ParticlesMover()     iPic3D::c_Solver::WriteOutput(int)	
17.5%				iPic3D::c_Solver::WriteRestart(int)	ipi:3d/iPIC3D.cpp
<0.1% 16.0%		4.7%	<0.1%	I other I other I iPic3D::c_Solver::CalculateMoments(),cray_memcpy_SNB	ipic3d/iPIC3D.cpp ipic3d/iPIC3D.cpp ipic3d/iPIC3D.cpp ipic3d/iPIC3D.cpp ipic3d/main/iPIC3D1i
0.4%		0.2%		2 others	ipic3d/iPIC3D.cpp

# It's possible to overlap different I/O approaches

#### HDF5 and VTK I/O operations occur at the same time on different ranks.

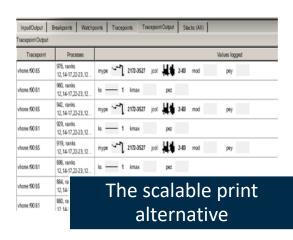
<u>File Edit View Metrics Wind</u>	dow <u>H</u> elp														
Profiled: iPIC3D on 1024 process	ses, 64 nodes Sampled fro	m: Wed Feb 7	7 2018 20:02	2:56 (UTC) for <b>730.</b>	3s										
Main thread activity		um l	MANK	INNIN ANNI		NUMB NOM	lik alatala		uilliu uil	lini. Imilli				lhihi	linital la
Lustre read rate	19.5	Million es	10001												
0.52 MB/s		<u>المن الر</u> الا ال	incon <mark>ette d</mark>	<u> 1997 (Name) (Name)</u>	<u>an martinan a</u>	<u></u>	<u></u>	ama in the second s		210 <del>00</del>		-			
Lustre write rate 0.04 GB/s	2.20				2							1			
Lustre metadata operation	0			<u> </u>	<u>~ ^ ~</u>			<u></u>	<u> </u>	<u> </u>	<u>~~ ~</u>			/_	
0.55 k /s	→ → → → → → → → → → → → → → → → → → →	611 <u>11 1</u>		antreit: antreit	a and an a	ا د بر امر رد									
Lustre file opens	8.39 71														
Input/Output	Project Files			d Stacks	Function	s								40000	
Main Thread Stac	ks														Time s
Total core time		<b>^</b>	MPI	Overhead	Function	(s) on lin	e								
					🗄 🥩 iPIC	3D [prog	ram]							-	
						ain									
					🕂 iPic3	D::c Sol	ver::Write	Output(	int)						
26.3%					主 Wr	iteMome	ntsVTK(G	rid3DCU	*, EMfie	elds3D*,	Collecti	ive*, VO	Ctopol.		
25.8%					🕂 Wr	iteFields\	VTK(Grid3	DCU*, E	Mfields	3D*, Col	lective*	, VCtop	ology.		
13.1%	h.,					_	olver::Writ	teRestar	t(int)					_	
0.4% _		<	<0.1%			others									IC3D.cpp IC3D.cpp
16.5%		2	2.1%	<0.1%		_	ver::Partic								in/iPIC3D1ib. in/iPIC3D1ib.
10.4% 📑 📕		2 <b>2</b>	2.1%	<0.1%		_	ver::Calcu								in/iPIC3Dlib.
						_	ver::Calcu		l(int)						IC3D.cpp
							::calculate							:3d/iPl	IC3D.cpp IC3D.cpp
6.9% 🛌		3	3.0%		÷ (	GMRES(v	oid (EMfie	lds3D::*	)(doub)	e*, dout	ole*), do	ouble*,	int, d.		in/iPIC3D1ib. elds/EMfields

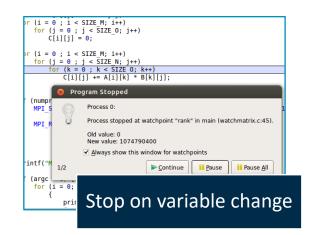
# Wrap Up

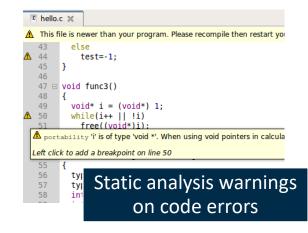


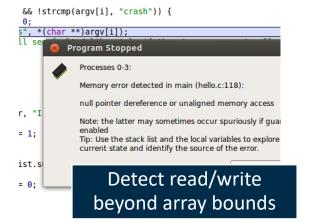
Confidential © 2018 Arm Limited

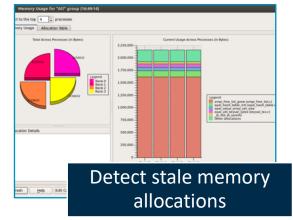
## Five great things to try with Arm DDT







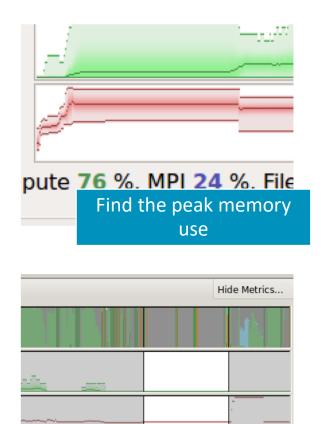






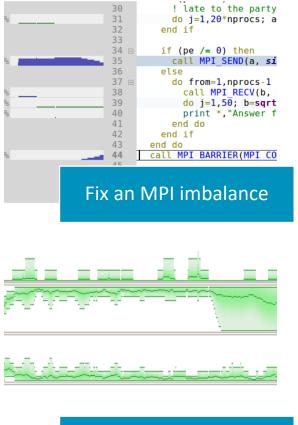


## Six Great Things to Try with Arm MAP



Make sure OpenMP

regions make sense



Project Files	Main Thread Stacks Functions						
tacks							
e 🔨 MPI 🛛 F	unction(s) on line						
	CallActionsSeparatedConcerns [in Call [inlined] hemelb::net::IteratedAction::Ca hemelb::extraction::PropertyA hemelb::extraction::Property						
	hemelb::extraction::LocalPr						
80.3%	PMPI_File_write_at						
from 32,76 R	emove I/O bottleneck						



size, nproc, mat\_a A[i\*size+k]\*B[k --/ .nal

Restructure for vectorization

arm

Sleeping

# Wrap Up

Visit arm.com/hpc to learn more about Arm Forge and download a free trial.

**CITM** ALLINEA STUDIO

- C/C++ Compiler
- Fortran Compiler
- Performance Libraries
- Forge (DDT and MAP)
- Performance Reports

- Tools are a must-have when programming HPC systems
- Use a structured, profile-driven optimization methodology
- Arm DDT can help improve code correctness
- Arm MAP can help improve code performance
- Arm Forge = DDT + MAP is a great choice at scale

Download at arm.com/hpc

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

# arm