Improving your code with Arm Forge

PDC-PRACE Workshop "HPC Tools for the Modern Era"

Conrad Hillairet conrad.hillairet@arm.com

An introduction to Arm

Arm is the world's leading semiconductor intellectual property supplier

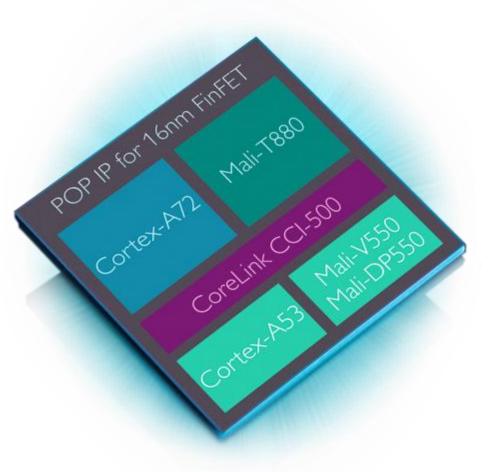
We license to over 350 partners: present in 95% of smart phones, 80% of digital cameras, 35% of all electronic devices, and a total of 60 billion Arm cores have been shipped since 1990

Our CPU business model:

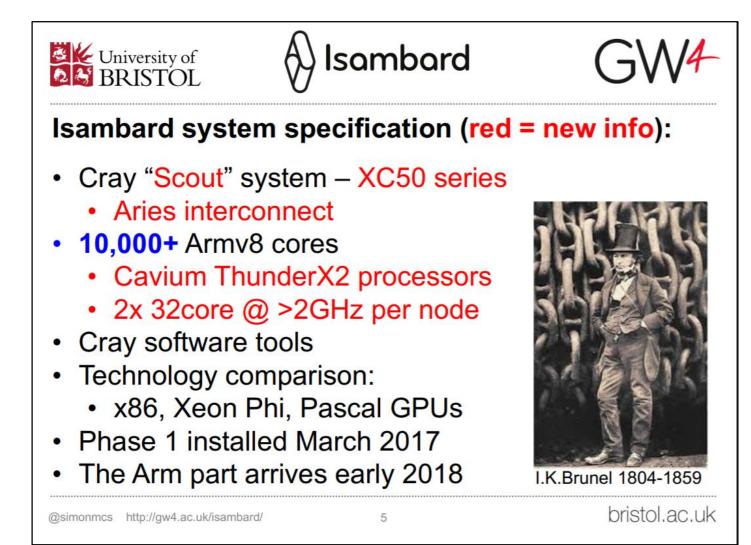
License technology to partners, who use it to create their own system-on-chip (SoC) products

- We may license an instruction set architecture (ISA) such as "Armv8-A"
- or a specific implementation, such as "Cortex-A72"

Partners who license an ISA can create their own implementation, ...and our IP extends beyond the CPU as long as it passes the compliance tests



Early HPC deployments



Catalyst UK

Accelerating Arm adoption in the UK

Sites and Target HPC Applications:

- EPCC: WRF, OpenFOAM, Rolls
 Royce Hydra opt, 2 PhD
 candidates
- Leicester: Data-intensive apps, genomics, MOAB Torque, DiRAC collab
- Bristol: VASP, CASTEP,
 Gromacs, CP2K, Unified Model,
 Hydra, NAMD, Oasis, NEMO,
 OpenIFS, CASINO, LAMMPS

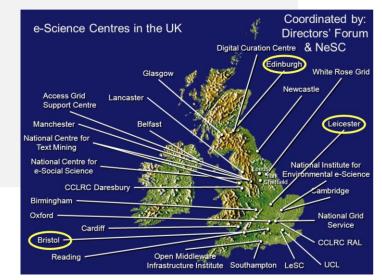


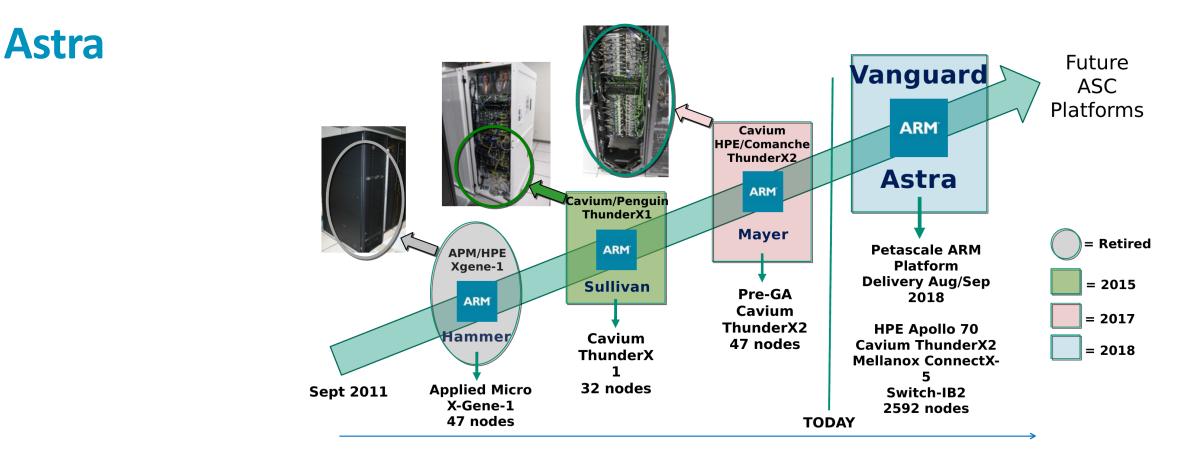
UNIVERSITY OF LEICESTER

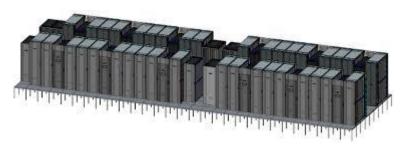


Typical Cluster for each site:

- 64 x Apollo 70 Compute Nodes (2 racks):
 - Dual socket Cavium 32c, 2.2 GHz
 - 256GB memory (16GB DIMMs)
 - Mellanox IB EDR CX5 Clos
 - 4096+ cores







Beskow 2.43 petaflops (<u>source</u>) Astra 2.32 petaflops (<u>source</u>)

Japan



Post-K: Fujitsu HPC CPU to Support ARM v8 ARM Fujitsu

Post-K fully utilizes Fujitsu proven supercomputer microarchitecture

Fujitsu, as a lead partner of ARM HPC extension development, is working to realize ARM Powered® supercomputer w/ high application performance

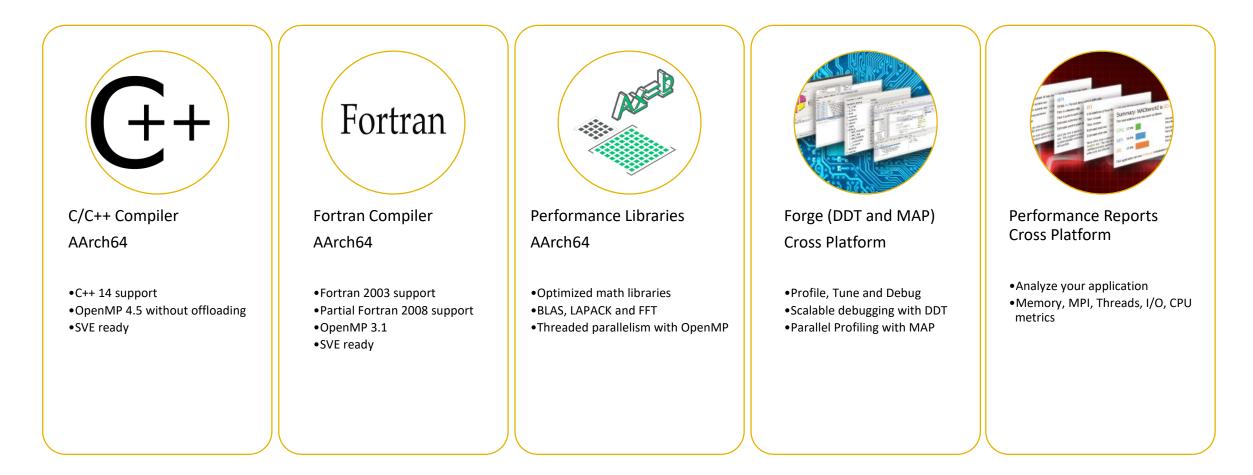
ARM v8 brings out the real strength of Fujitsu's microarchitecture

HPC apps acceleration feature	Post-K	FX100	FX10	K computer	
FMA: Floating Multiply and Add	~	~	~	~	
Math. acceleration primitives*	✓Enhanced	~	~	~	
Inter core barrier	~	~	~	~	
Sector cache	✓Enhanced	~	~	~	
Hardware prefetch assist	✓Enhanced	~	~	~	
Tofu interconnect	✓Integrated	✓ Integrated	~	~	

Mathematical acceleration primitives include trigonometric functions, sine & cosines, and exponential...

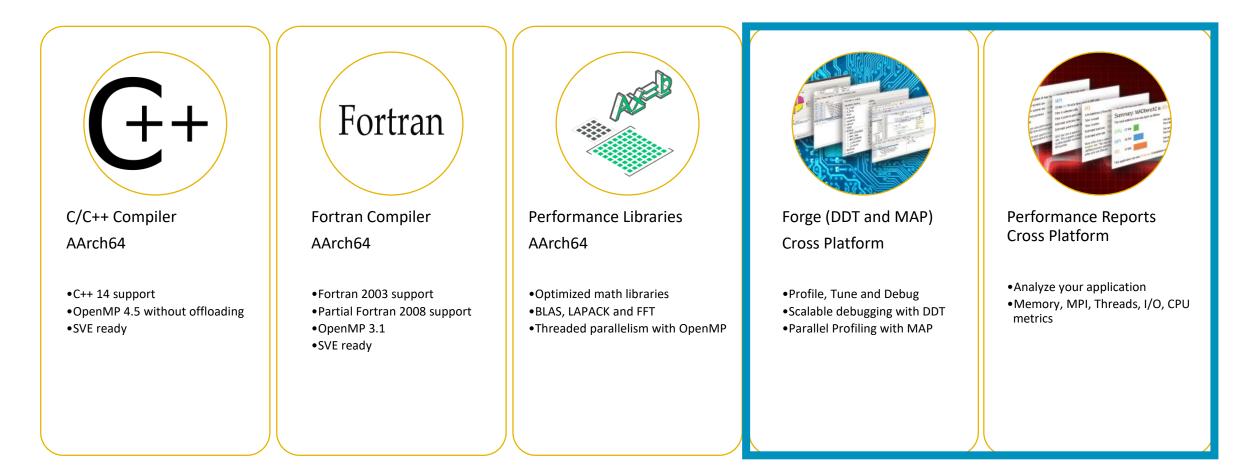
Conrad : Support Engineer - Arm Allinea Studio and Arm Forge

A quick glance at what is in Arm Allinea Studio



Conrad : Support Engineer - Arm Allinea Studio and Arm Forge

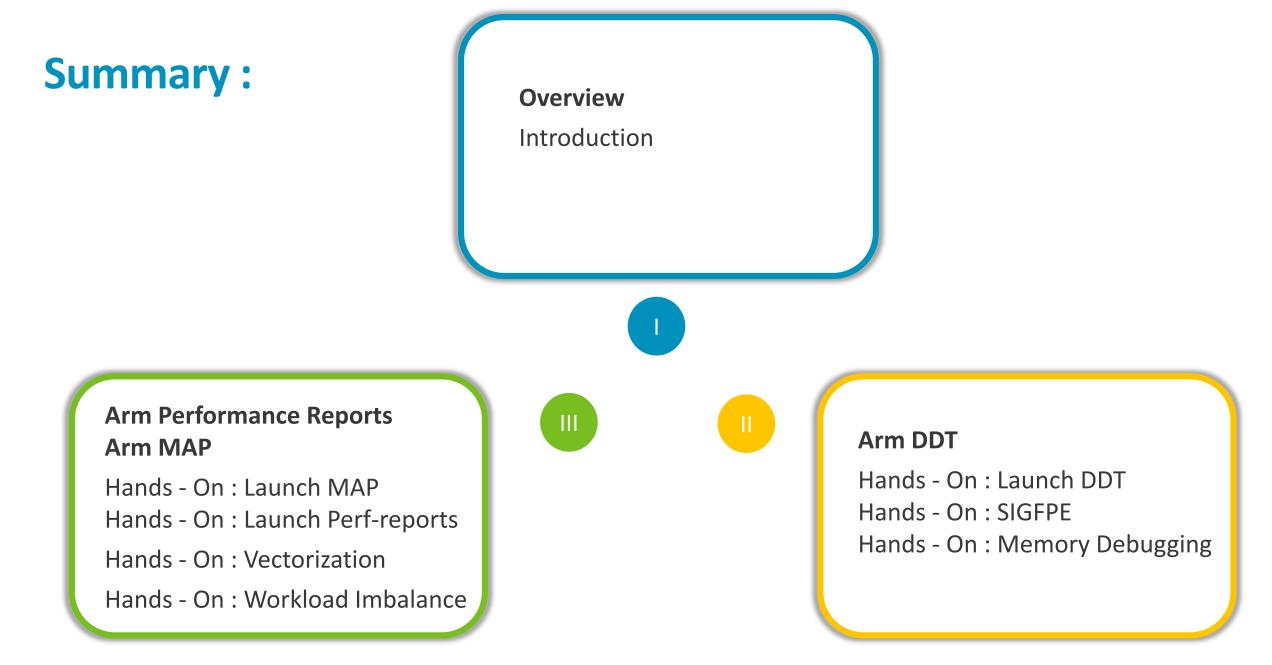
A quick glance at what is in Arm Allinea Studio



Summary



© 2017 Arm Limited



Overview



© 2017 Arm Limited

Extra documentation

PDC Documentation : <u>https://www.pdc.kth.se/software/software/allinea-forge/index.html</u>

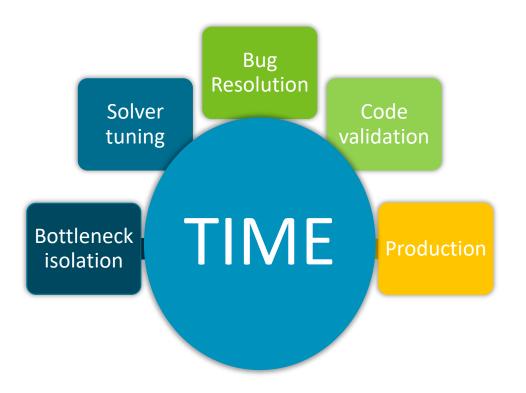
Arm DDT User Guide : <u>https://developer.arm.com/docs/101136/latest/ddt</u>

Arm MAP User Guide : <u>https://developer.arm.com/docs/101136/latest/map</u>

Arm Performance Reports User Guide : <u>https://developer.arm.com/docs/101137/latest/introduction</u>

Arm Forge Webinars : <u>https://developer.arm.com/products/software-development-tools/hpc/training/arm-hpc-tools-webinars</u>

We do tools for a single reason: help people save their time.

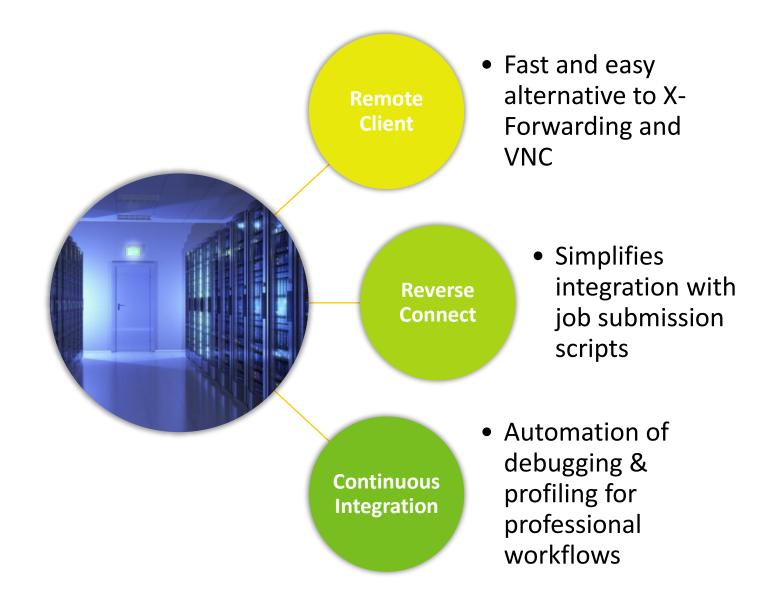


Achieving performance portability

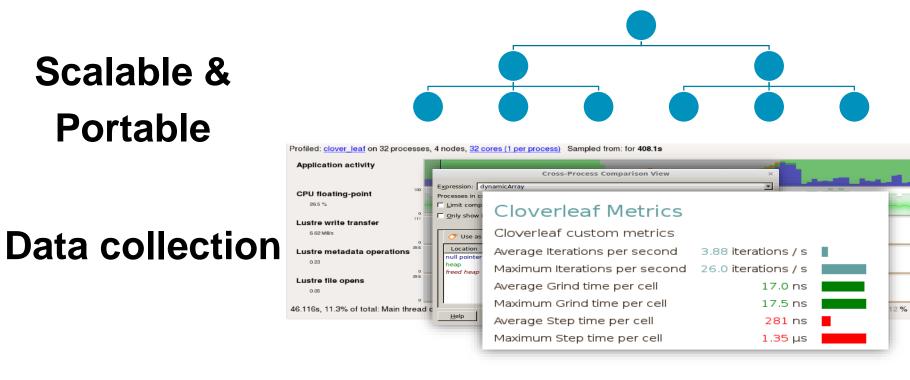
Retrieve useful data Turn "a lot of" data into meaningful information Turn information into better code

Use powerful tools easily

Using powerful tools more easily



Generating useful and meaningful information



Data processing



Arm Forge

An interoperable toolkit for debugging and profiling



Commercially supported by Arm





The de-facto standard for HPC development

- Most widely-used debugging and profiling suite in HPC
- Fully supported by Arm on Intel, AMD, Arm, 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

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)

9 Step guide: optimizing high performance applications

Improving the efficiency of your parallel software holds the key to solving more complex research problems faster. This pragmatic, 9 Step best practice guide will help you identify and focus on application readiness, bottlenecks and optimizations one step at a time.



Arm DDT

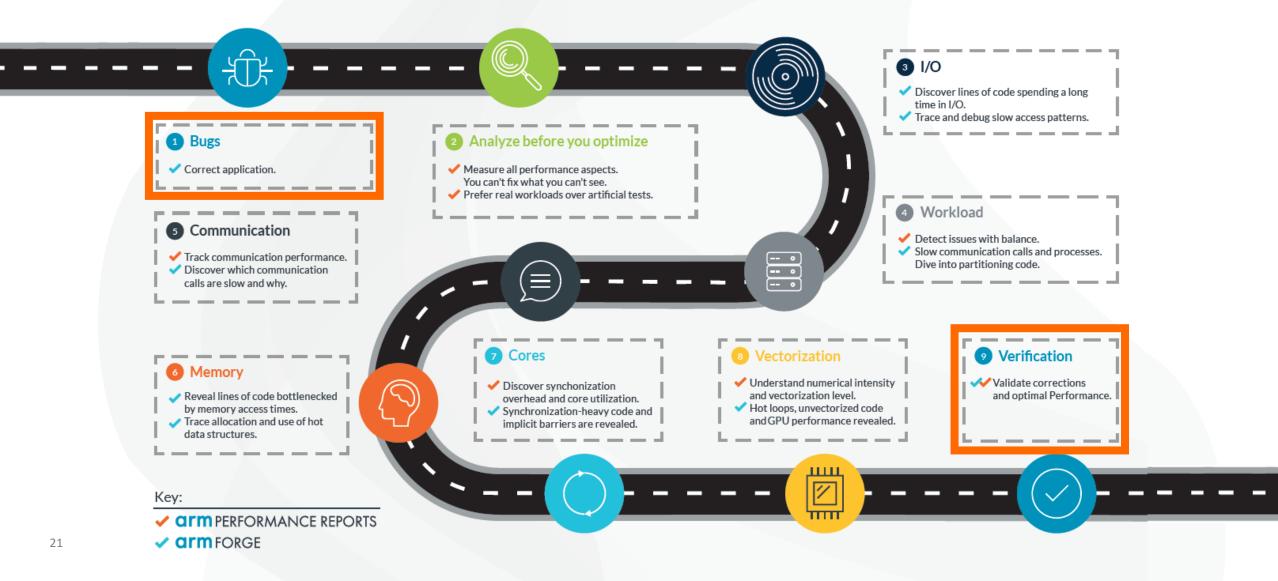


© 2017 Arm Limited

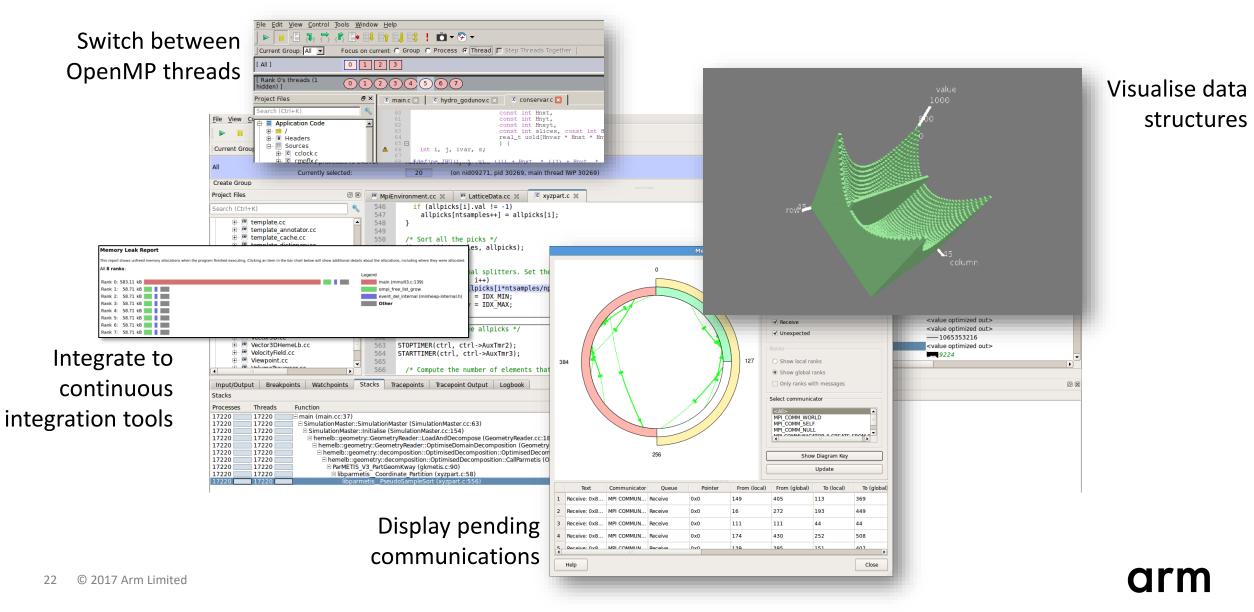
9 Step guide: optimizing high performance applications

arm

Improving the efficiency of your parallel software holds the key to solving more complex research problems faster. This pragmatic, 9 Step best practice guide will help you identify and focus on application readiness, bottlenecks and optimizations one step at a time.



Migrate and debug application



Five great things to try with Arm DDT

&& !strcmp(argv[i], "crash")) {

Program Stopped Processes 0-3:

enabled

s", *(char **)argv[i]);

0:

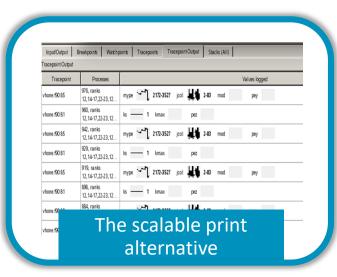
11 s

r, "I

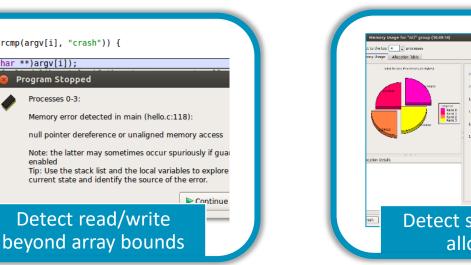
= 1;

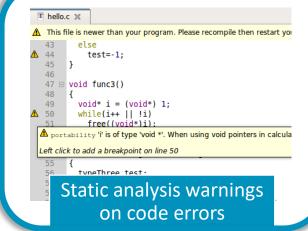
ist.s

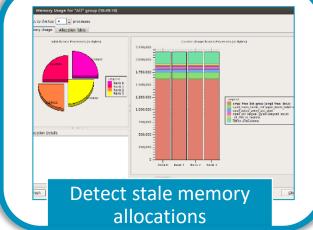
= 0







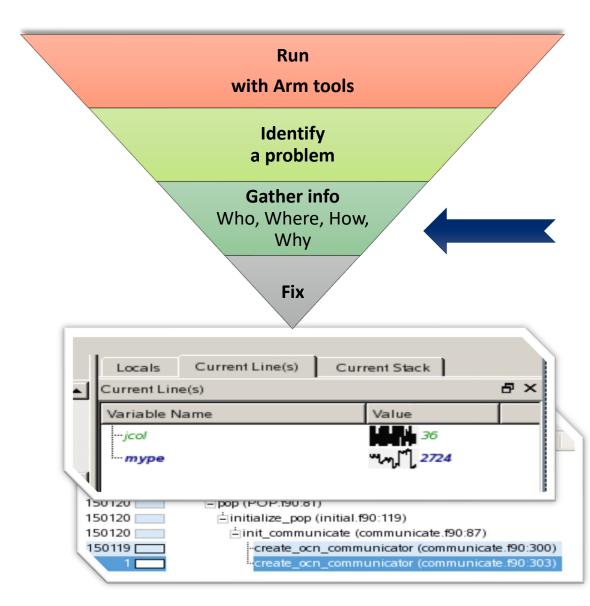




Arm DDT – The Debugger

Who had a rogue behavior ?

- 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



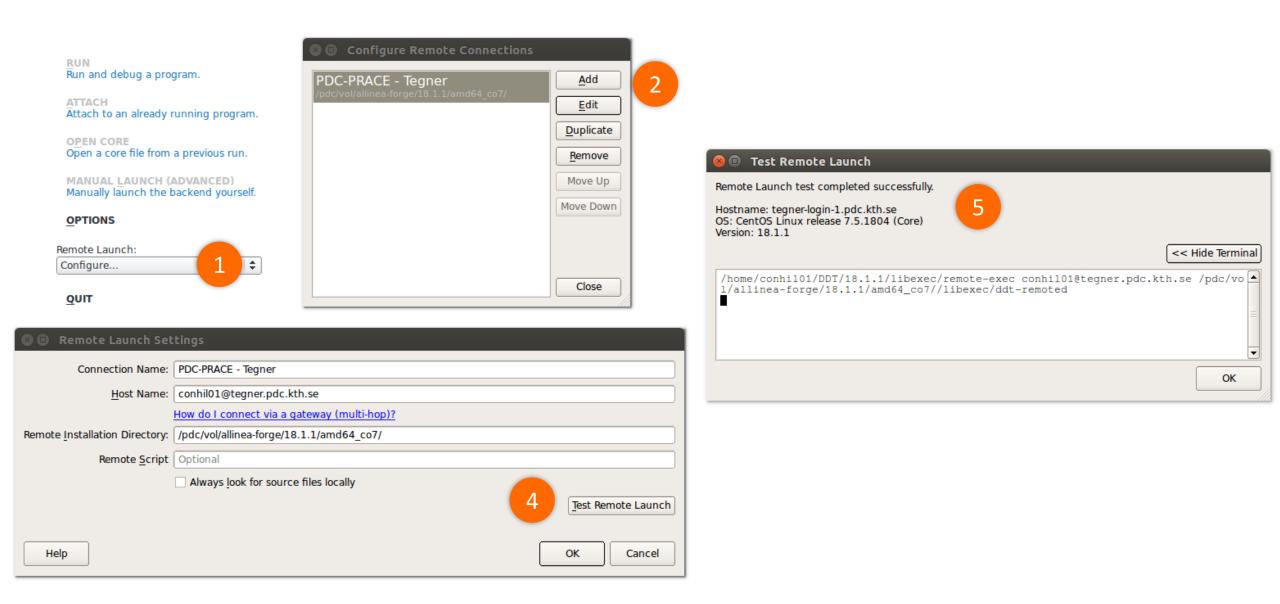
Hands – On : Set up the Tools

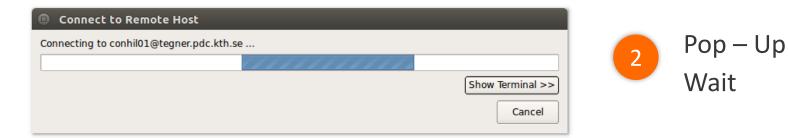


© 2017 Arm Limited

Reverse-Connect – Client / Laptop side

```
kinit -f <userName>@NADA.KTH.SE
klist -f
export PATH=$PATH:<pathToForgeInstall>/bin
export PATH=$PATH:/home/prace/arm/forge/bin
ddt --version
ddt
```





RUN Run and debug a program.

ATTACH Attach to an already running program.

OPEN CORE Open a core file from a previous run.

MANUAL LAUNCH (ADVANCED) Manually launch the backend yourself.

OPTIONS

Remote Launch:

Off Configure...

PDC-PRACE - Tegner

RUN Run and debug a program.

ATTACH Attach to an already running program.

OPEN CORE Open a core file from a previous run.

MANUAL LAUNCH (ADVANCED) Manually launch the backend yourself.

OPTIONS



Remote Launch:

PDC-PRACE - Tegner

\$

arm

Reverse-Connect Client ready

Reverse-Connect – Server / Cluster side

```
ssh conhil01@tegner.pdc.kth.se
module load i-compilers
module load intelmpi
module load allinea-forge
cd /cfs/klemming/nobackup/c/conhil01
cp /afs/pdc.kth.se/home/c/conhil01/Public/arm trial.tar.gz .
tar -xvf arm trial.tar.gz
cp /afs/pdc.kth.se/home/c/conhil01/Public/Licence kth .
unset ALLINEA LICENSE FILE modshare
unset ALLINEA LICENSE FILE
export ALLINEA FORCE LICENCE FILE=$PWD/Licence kth
cd arm trial
cd 0 test reverse connect
make
salloc -nodes=1 -t 00:10:00 -A pdc-test-2018
              mpirun -n 2 ./hello c.exe
ddt<sup>e 201<u>7 Arm</u> coonnect</sup>
```

Reverse-Connect – Client / Laptop side

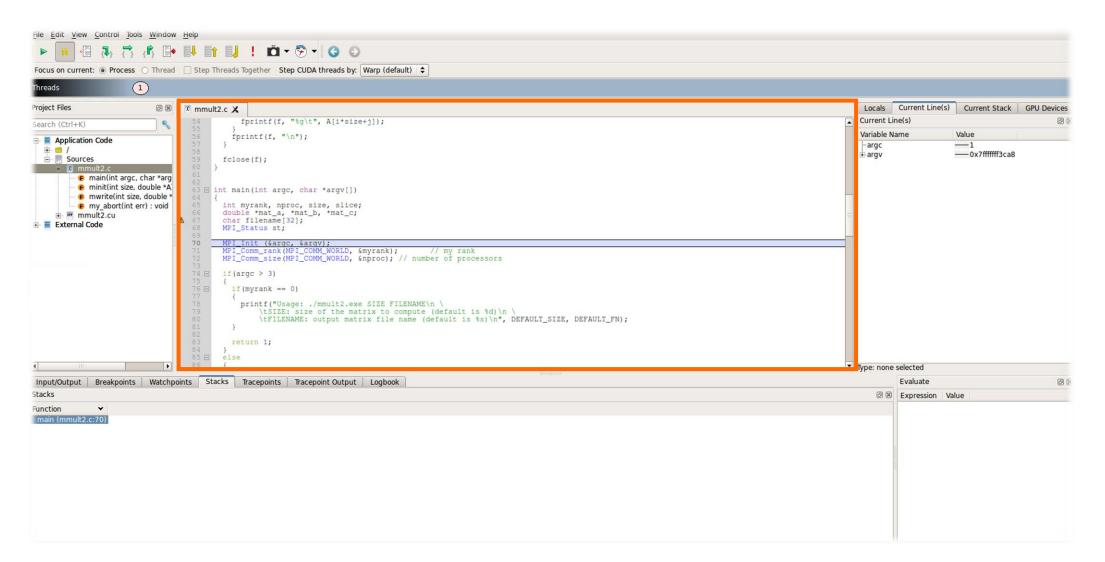


😣 🗊 Run	
Run: mpirun -n 2 ./hello_c.exe	Details
Command: mpirun -n 2 ./hello_c.exe	
OpenMP	Details
	Details
Memory Debugging	Details
Plugins: none	Details
Help Options 2	Disconnect

User Interface

File Edit View Control Tools Window Help				
Focus on current: Process 🔿 Thread 📋 Step Threads Together Step CUDA threads by: Warp (default) 💠				
Threads 1				
Project Files 💿 🗷 📧 mmult2.c 🗶	Locals	Current Line(s)	Current Stack	
Search (Ctrl+K) Search (Ctrl+	Current L			0 X
<pre></pre>	Variable M		Value	
	te argv		0x7fffffff3ca8	
= 🖸 mmult2.c				
□ ⊕ main(int argc, char *arg 62 □ ⊕ minit(int size, double *A 63 □ int main(int argc, char *argv[])				
- 😨 mwrite(int size, double * 64 {				
m mmult2 cu 66 double *mat_a, *mat_b, *mat_c;	=			
MPI_Status st;				
69 70 MPI Init (&argc, &argv);				
71 MPI_Comm_rank(MPI_COMM_WORLD, &myrank); // my rank 72 MPI_Comm_size(MPI_COMM_WORLD, &nproc); // number of processors				
73 74 🖂 if(argc > 3)				
$75 \{ \\ 76 \boxminus \text{ if (myrank == 0)} $				
77 { 78 printf("Usage: ./mmult2.exe SIZE FILENAME\n \				
<pre>79 \tSIZE: size of the matrix to compute (default is %d)\n \ 80 \tFILENAME: output matrix file name (default is %s)\n", DEFAULT_SIZE, DEFAULT_FN);</pre>				
81 } 82				
83 return 1; 84 }				
85 ⊡ else 86 {	Tinoi non	colocted		
Input/Output Breakpoints Watchpoints Stacks Tracepoint Output Logbook		Evaluate		Ø×
Stacks	0 ×	Expression Val	ue	
Function 🗸				
main (mmult2.c:70)				

User Interface – Source code viewer



User Interface – Play/ Pause / Step

Play : Run everything. Use typically at the beginning or after Pause Pause : Stops running current kernel

Step In : Enter a function call and display source code of the functionStep Over : Execute current line of codeStep Out : Comes back one stage above current stack

	Mindow Help	ult) 🗘
Threads 1		
Project Files	Ø⊠ III mmult2.c ★	
Search (Ctrl+K)	<pre>54 fprintf(f, "%g\t", A[i*size+j]); 55 }</pre>	
Application Code	56 fprintf(f, "\n"); 57 } 58 59 fclose(f); 60 }	

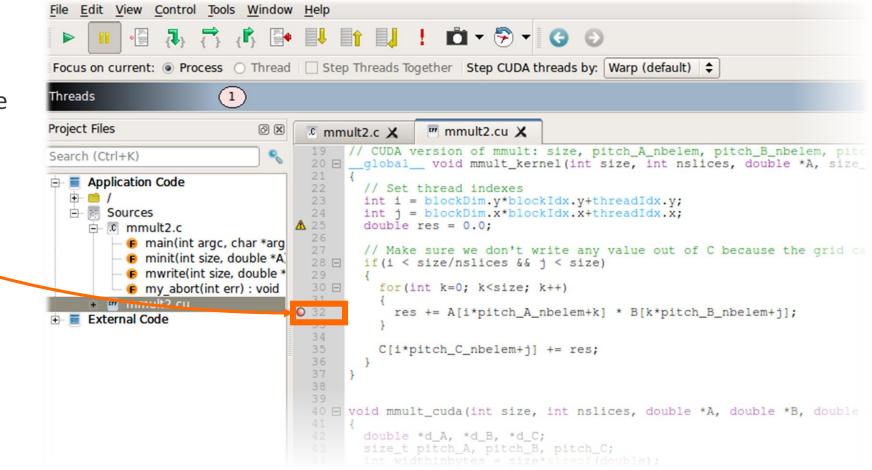
User Interface – Add Breakpoints – Way 1

ne <u>r</u>	uic <u>v</u> i	ew <u>u</u>	ontroi	10015	<u>vv</u> ii	woun	<u>н</u> ер				
	-	•	(1)	₹	{ 1 }	100 (ļ		1	Ö •	5

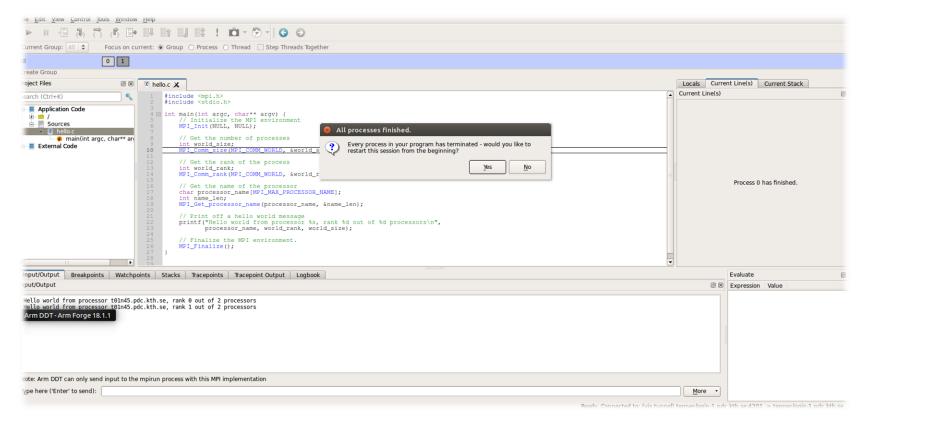
🔊 🗉 🛛 Add Bi	reakpoint
Location:	
Line Line	File: sers/hck06/DDT/Examples/CUDA_debugging/mmult2.c
	Line <u>N</u> umber: 70
O Function	
Applies To:	
Thread: All	÷
Hit Limits:	
<u>S</u> tart on the	n-th pass: 0
Trigger <u>e</u> very	
St <u>o</u> p after n	
<u>C</u> ondition:	
<u>L</u> anguage:	Auto
Help	<u>A</u> dd Cancel

User Interface – Add Breakpoints – Way 2

In the source code viewer, on the left, left click on the line to add a Breakpoint Typical next action : Play



Reverse-Connect – Client / Laptop Side



<u>F</u> ile	<u>Edit View Control</u>	Tools	Window	Hel				
	New Session			•				
	Load Session							
	Save Session							
	Restart Session							
	Change Attached Proc	20220						
	End Session							
Ø	Open Source File		Ctrl+0					
8	Save Source File		Ctrl+S					
2	Save Source File As		Ctrl+Shift+S					
8	Revert to Saved Source	e File						
	Close Source File		Ctrl+W					
	Close All Source Files							
ø	Build		Ctrl+B					
	Configure Build							
	Co <u>m</u> mit							
	Options							
	Import Configuration File							
	Quit							

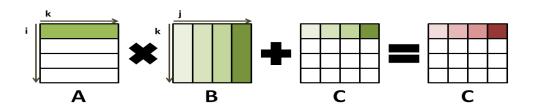
arm

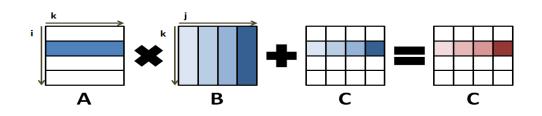
Hands – On : SIGFPE (Arithmetic Exception)

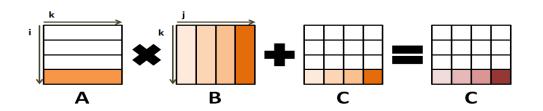


Matrix Multiplication Example

$$C = A \times B + C$$







arm

Environment configuration (reminder)

ssh conhil01@tegner.pdc.kth.se

module load i-compilers
module load intelmpi
module load allinea-forge

cp /afs/pdc.kth.se/home/c/conhil01/Public/arm_trial.tar.gz .
tar -xvf arm_trial.tar.gz

cp /afs/pdc.kth.se/home/c/conhil01/Public/Licence_kth .
unset ALLINEA_LICENSE_FILE_modshare
unset ALLINEA_LICENSE_FILE
export ALLINEA_FORCE_LICENCE_FILE=\$PWD/Licence_kth

Hands – On : SIGFPE

- 1_interactive_debugging
- Compile the program
- Run one of the binaries. What do you see ?
- Let's debug it then !
- Recompile with DEBUG=1, launch DDT and ... debug !
- Can you find where the problem comes from ?
- Modify the code and recompile (in DDT)
- Relaunch the program.

Hands – On : Memory Debugging



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

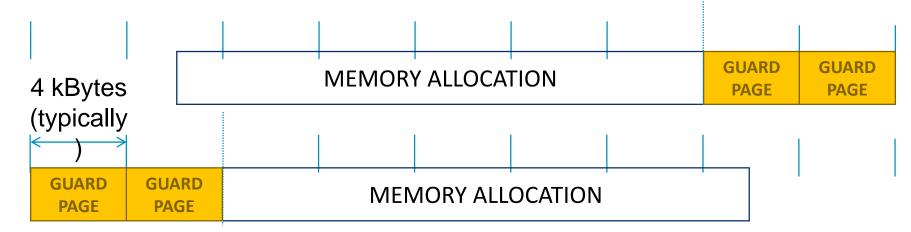
•Check to see if space that was blanked when a pointer was allocated/freed has been overwritten.

check-blank

check-funcs

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

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

Compilation flags for memory debugging

Compiler : -00 -g

```
Linking:-L<path_to_DDT_install>/lib/64 -Wl,--allow-multiple-
definition,--undefined=malloc,--undefined=_ZdaPv -
ldmallocthcxx
```

Memory debugging

RUN Run and debug a program.

ATTACH Attach to an already running program.

OPEN CORE Open a core file from a previous run.

MANUAL LAUNCH (ADVANCED) Manually launch the backend yourself.

OPTIONS

Remote Launch:

(via tunnel) daint105:4201 -> daint1 🗢

QUIT

🛞 🗉 Run			
Run: srunreservation=hackathon -C gpu -n 1 ./mmult2.exe	Details		
Command: srunreservation=hackathon -C gpu -n 1 ./mmult2.exe			
OpenMP	Details		
CUDA: Track allocations: enabled, Detect invalid accesses: enabled	Details		
✓ Track GPU allocations (also enables CPU memory debugging)			
✓ Detect invalid accesses (memcheck)			
✓ Memory Debugging: Fast / Balanced, 1 guard page after, Backtrac	Details		
Plugins: none	Details		
Help Options Run	Disconnect		

Memory Debugging Options

<u>Preload</u> the memory	ry debugging library	Language: C/Fo	ortran, threads 🛛 🖨
	works for programs li ally linked, you must r		
Heap Debugging			
Fast	Balanced	Thoro	ugh Custom
			· · ·
Enabled Checks:	basic,check-fence,free-	protect	More Information
Heap Overflow/Unde	rflow Detection		
✓ Add guard pages	s to detect out of boun	ds heap access	
Guard pages: 1	Add guard page	es: After 🗘)
Ad <u>v</u> anced			
✓ Check heap cons	sistency every 100	+ heap opera	ations
✓ Store stack <u>b</u> ack	traces for memory allo	cations	
Only enable for t	hese processes:		
0	100% Sele	ect All x2	x0.5 1%
Help		ОК	Cancel

arm

Environment configuration (reminder)

ssh conhil01@tegner.pdc.kth.se

module load i-compilers
module load intelmpi
module load allinea-forge

cp /afs/pdc.kth.se/home/c/conhil01/Public/arm_trial.tar.gz .
tar -xvf arm_trial.tar.gz

cp /afs/pdc.kth.se/home/c/conhil01/Public/Licence_kth .
unset ALLINEA_LICENSE_FILE_modshare
unset ALLINEA_LICENSE_FILE
export ALLINEA_FORCE_LICENCE_FILE=\$PWD/Licence_kth

Hands – On : Memory debugging

- 3_offline_debugging
- Compile the program
- Run one of the binaries. What do you see ?
- No problem ? Are you sure ? Let's launch DDT, just in case!
- Recompile with DEBUG=1
- Launch the application with DDT
- Check memory debugging and guard-pages
- Run the program ... Any problem ?
- Can you resolve it ?
- Modify the code and recompile (in DDT)
- Relaunch the program.

Hands – On : Memory debugging

- Are you sure we are done with hidden issues ?
- Use DDT offline report with "--offline --mem-debug" flags
- Have a look to the report, anything suspicious ?
- Do you see how to fix this ?

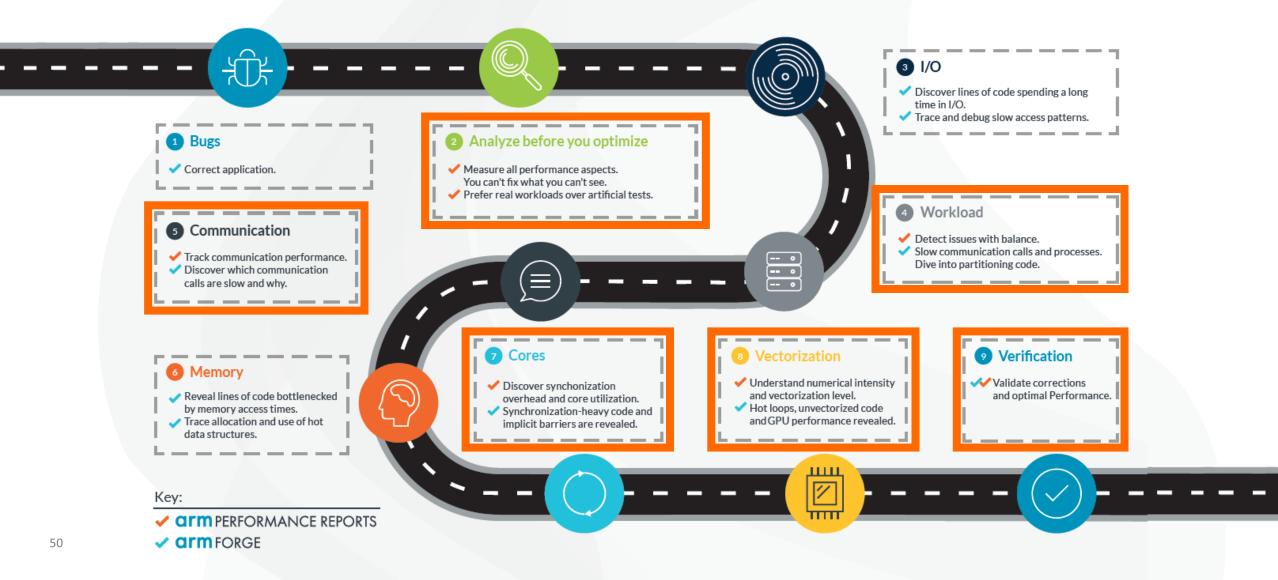
Arm Performance Reports



9 Step guide: optimizing high performance applications

arm

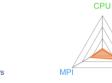
Improving the efficiency of your parallel software holds the key to solving more complex research problems faster. This pragmatic, 9 Step best practice guide will help you identify and focus on application readiness, bottlenecks and optimizations one step at a time.



"Learn" with Arm Performance Reports

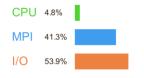


MADbench2 16 processes, 1 node sandybridge2 Mon Nov 4 12:27:50 2013 109 seconds (2 minutes) /tmp/MADbench2 12-core server / HDD / 16 readers + writers



Summary: MADbench2 is I/O-bound in this configuration

The total wallclock time was spent as follows:



Time spent running application code. High values are usually good. This is **low**; it may be worth improving I/O performance first. Time spent in MPI calls. High values are usually bad. This is **average**; check the MPI breakdown for advice on reducing it.

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

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

CPU

w the 4.	8% total CPU time was spent:
4.9%	1
0.1%	1
95.0%	
0.0	
	4.9% 0.1% 95.0%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance. No time was spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

I/O





Most of the time is spent in write operations, which have a very low 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.

MPI

Of the 41.3% total time spent in MPI calls:			
Time in collective calls	100.0%		
Time in point-to-point calls	0.0%	1	
Estimated collective rate	4.07 bytes/s		
Estimated point-to-point rate	0 bytes/s	1	

All of the time is spent in collective calls with a very low transfer rate. This suggests a significant load imbalance is causing synchronization overhead. You can investigate this further with an MPI profiler.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	160 Mb	
Peak process memory usage	173 Mb	
Peak node memory usage	17.2%	•

The **peak node memory usage** is low. You may be able to reduce the total number of CPU hours used by running with fewer MPI processes and more data on each process.

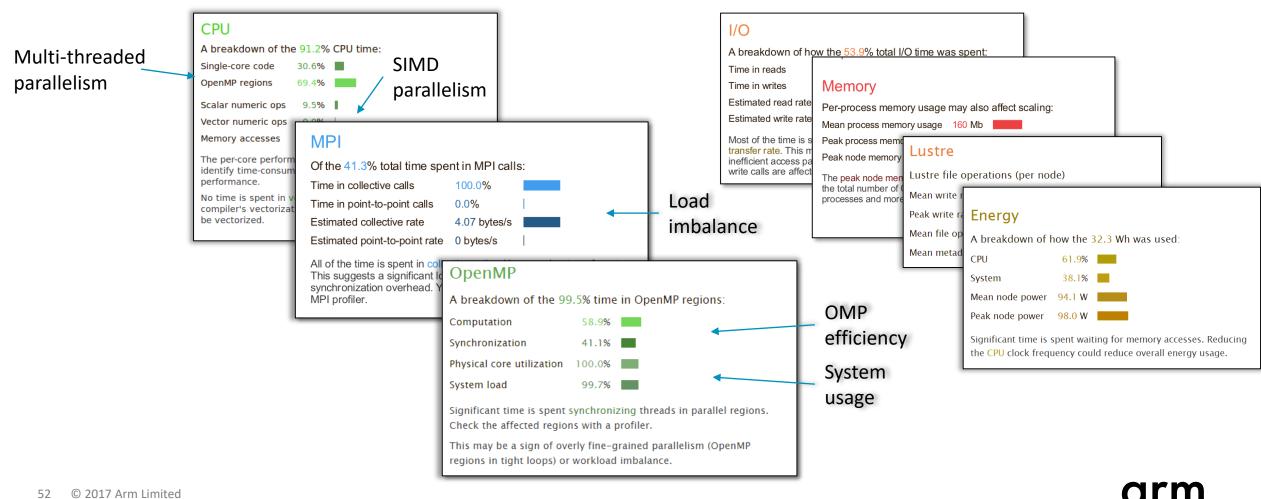
Very simple start-up

Fully scalable, very low overhead

Rich set of metrics

Powerful data analysis

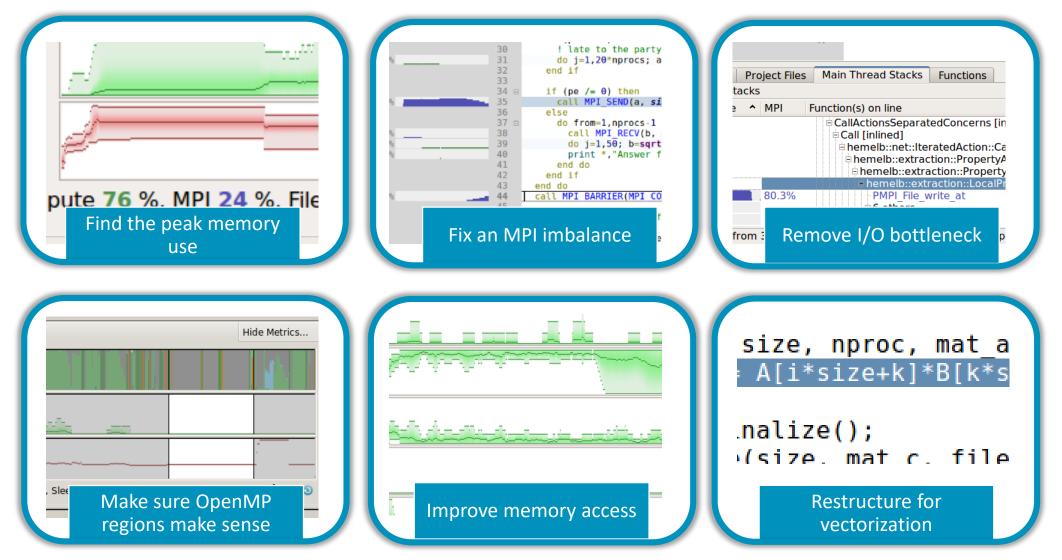
Metrics Overview



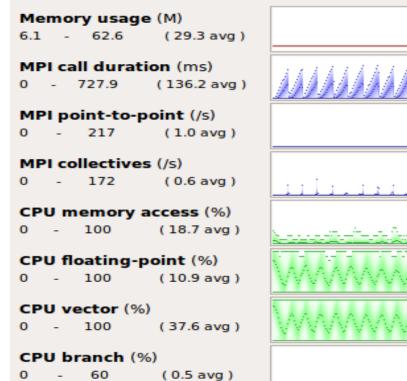
Arm MAP

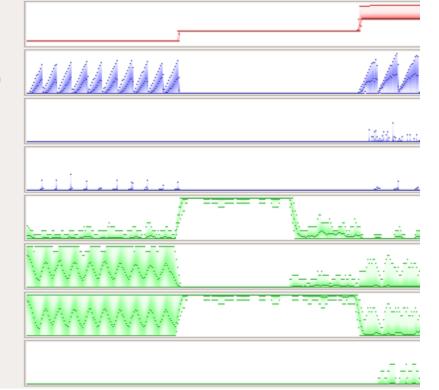


Six Great Things to Try with Arm MAP



Glean Deep Insight from our Source-Level Profiler





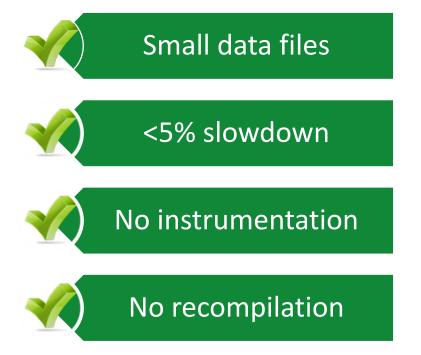
Track memory usage across the entire application over time

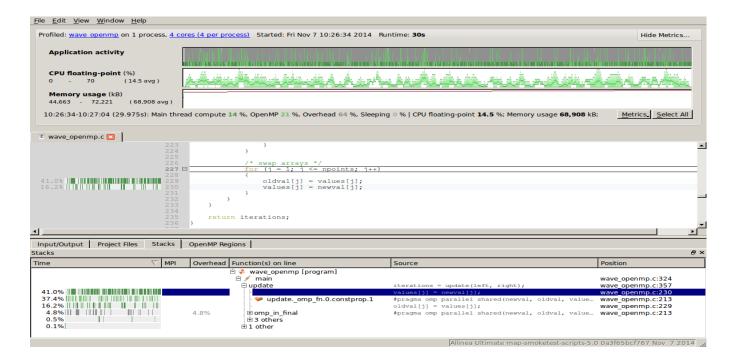
Spot MPI and OpenMP imbalance and overhead

Optimize CPU memory and vectorization in loops

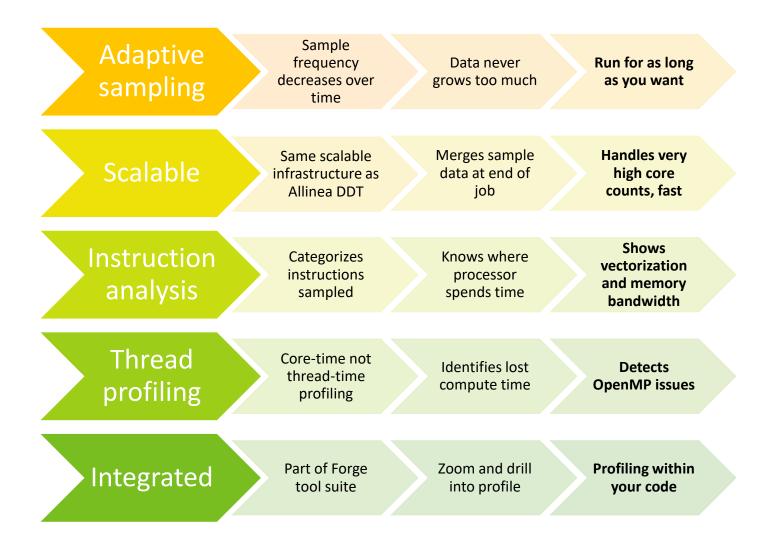
Detect and diagnose I/O bottlenecks at real scale

Allinea MAP – The Profiler





How Arm MAP is different



Preparing Code for Use with MAP

To see the source code, the application should be compiled with the debug flag typically **-g**

It is recommended to *always* keep optimization flags on when profiling

Hands – On : Launch MAP



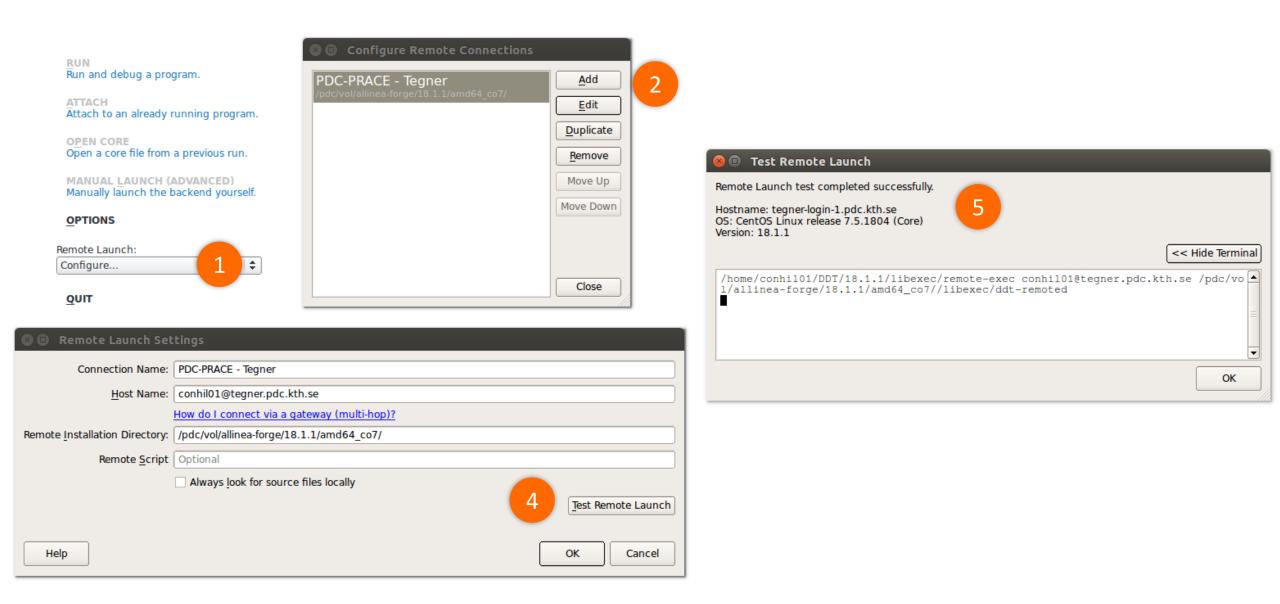
Reverse-Connect – Client / Laptop side

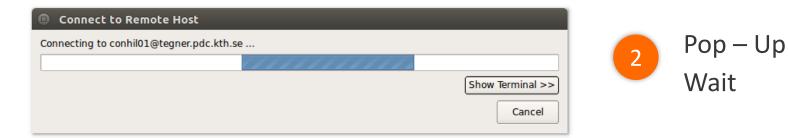
kinit -f <userName>@NADA.KTH.SE

klist -f

export PATH=\$PATH:<pathToForgeInstall>/bin

map





RUN Run and debug a program.

ATTACH Attach to an already running program.

OPEN CORE Open a core file from a previous run.

MANUAL LAUNCH (ADVANCED) Manually launch the backend yourself.

OPTIONS

Remote Launch:

Off Configure...

PDC-PRACE - Tegner

RUN Run and debug a program.

ATTACH Attach to an already running program.

OPEN CORE Open a core file from a previous run.

MANUAL LAUNCH (ADVANCED) Manually launch the backend yourself.

OPTIONS



Remote Launch:

PDC-PRACE - Tegner

\$

arm

Reverse-Connect Client ready

Reverse-Connect – Server / Cluster side

```
ssh conhil01@tegner.pdc.kth.se
module load i-compilers
module load intelmpi
module load allinea-forge
cp /afs/pdc.kth.se/home/c/conhil01/Public/arm trial.tar.gz .
tar -xvf arm trial.tar.gz
cp /afs/pdc.kth.se/home/c/conhil01/Public/Licence kth .
unset ALLINEA LICENSE FILE modshare
unset ALLINEA LICENSE FILE
export ALLINEA FORCE LICENCE FILE=$PWD/Licence kth
cd arm trial
cd 0 test reverse connect
make
salloc -nodes=1 -t 00:10:00 -A pdc-test-2018
map --connect mpirun -n 2 ./hello c.exe
  © 2017 Arm Limited
```

Reverse-Connect – Client / Laptop side



B Run	
Application: ./hello_c.exe	Details
Application: ./hello_c.exe	◄ 🗧
Arguments:	~
stdin file:	• 🗧
Working Directory:	- 3
Duration: Sampling entire program	Details
Metrics	Details
CUDA Kernel analysis	Details
Run: mpirun -n 4 ./hello_c.exe	Details
Implementation: no MPI Change	
Profile selected processes: 0 100%	Select All
OpenMP	Details
Submit to Queue Configure Par	rameters
Environment Variables: none	Details
Help Options 2 Run	Disconnect

arm

Hands – On : Launch Perf-Reports



Launch Performance Reports

```
ssh conhil01@tegner.pdc.kth.se
module load i-compilers
module load intelmpi
module load allinea-reports
cp /afs/pdc.kth.se/home/c/conhil01/Public/arm trial.tar.gz .
tar -xvf arm trial.tar.gz
cp /afs/pdc.kth.se/home/c/conhil01/Public/Licence kth .
unset ALLINEA LICENSE FILE modshare
unset ALLINEA LICENSE FILE
export ALLINEA FORCE LICENCE FILE=$PWD/Licence kth
cd arm trial
cd 0 test reverse connect
make
salloc -nodes=1 -t 00:10:00 -A pdc-test-2018
perf-report mpirun -n 2 ./hello c.exe
```

Visualize Performance Reports outputs

- Two files outputted : .txt and .html
- .txt can be visualized on the cluster with file editor
- Use scp to copy the .html file back to your laptop
- Open it with a Web Browser

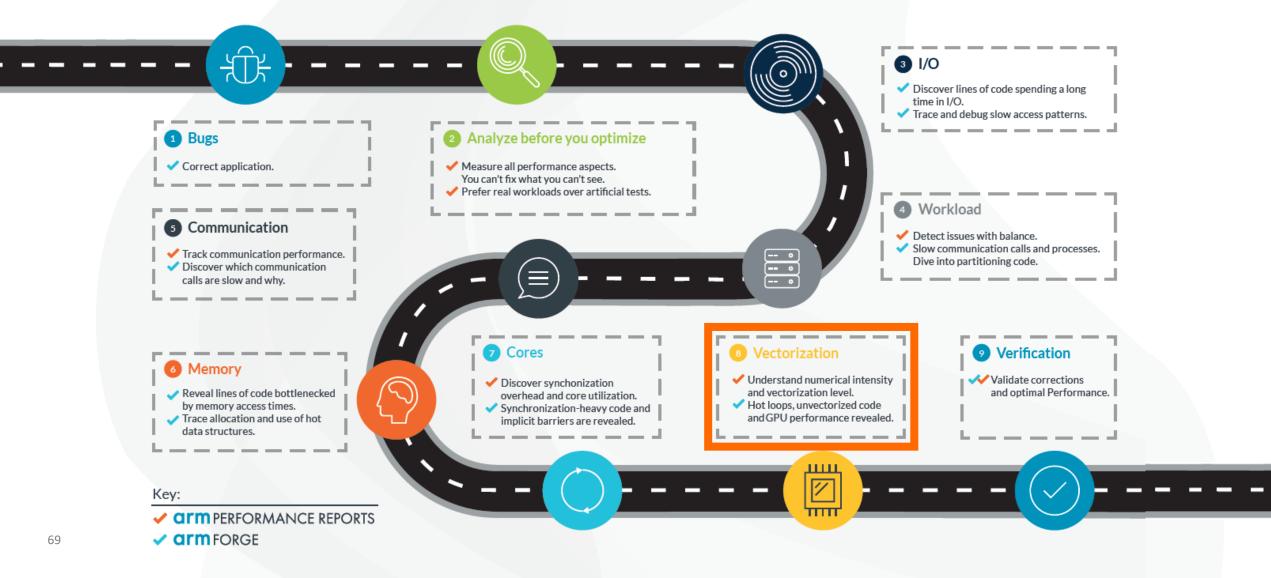
Hands – On : Vectorization



9 Step guide: optimizing high performance applications

arm

Improving the efficiency of your parallel software holds the key to solving more complex research problems faster. This pragmatic, 9 Step best practice guide will help you identify and focus on application readiness, bottlenecks and optimizations one step at a time.



Computational Intensity

"My program is doing a lot of computation ... How do I make it go faster"

DO k=y_min-2,y_max+2

```
DO j=x_min-2,x_max+2
```

```
pre_vol(j,k)=volume(j,k)+(vol_flux_x(j+1,k)-vol_flux_x(j,k)+vol_flux_y(j,k+1)-vol_flux_y(j,k))
post_vol(j,k)=pre_vol(j,k)-(vol_flux_x(j+1,k)-vol_flux_x(j,k))
ENDDO
```

ENDDO

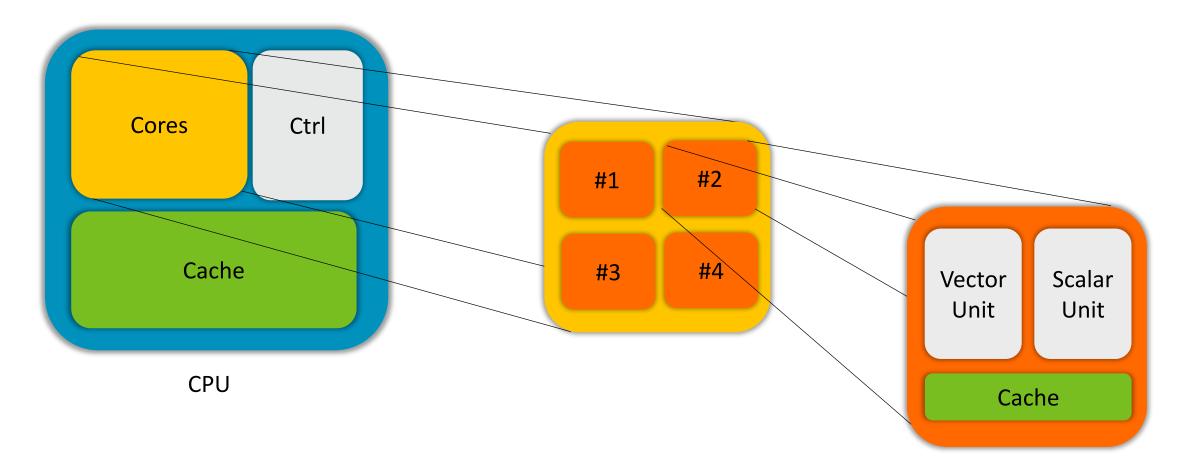
•••

...

```
Example with modified version of CloverLeaf
```

- non-threaded version without OpenMP
- MPI, no IO

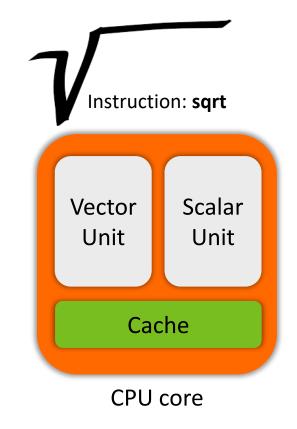
Vector Units



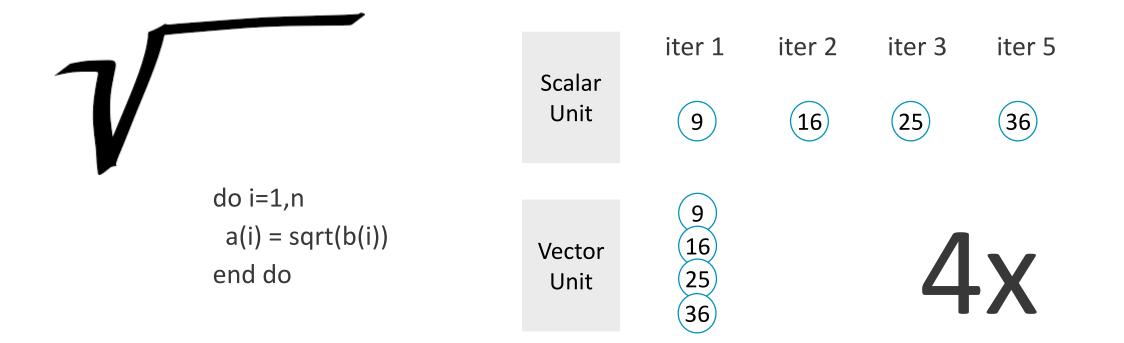
arm

Vectorization / SIMD





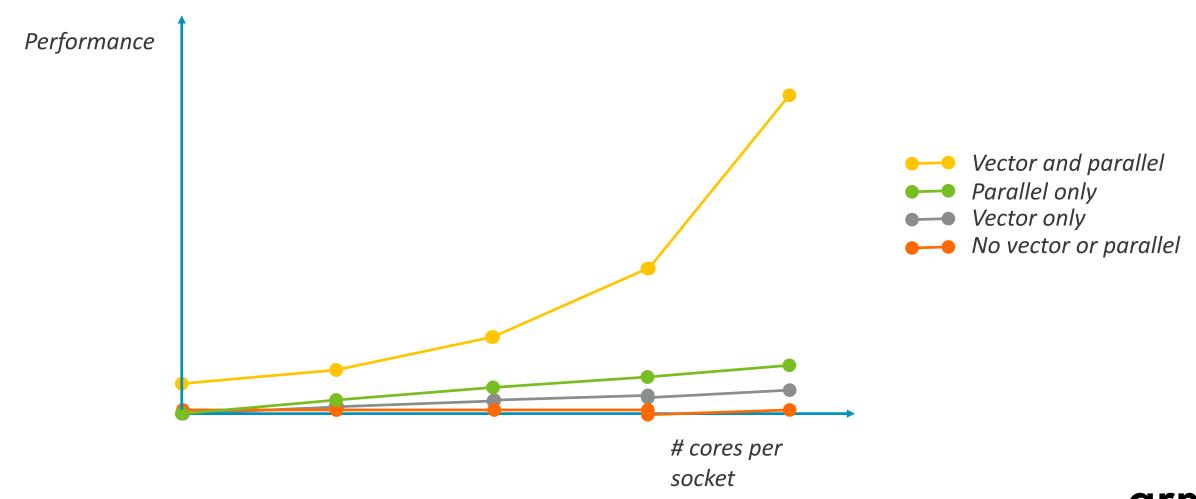
Vectorization / SIMD



Intel[®] AVX2: 256-bit vector unit → 8 SP / 4 DP Intel[®] AVX-512: 512-bit vector unit → 16 SP / 8 DP Arm[®] NEON: 128-bit vector unit → 4 SP / 2 DP



Why? Performance lies in the software

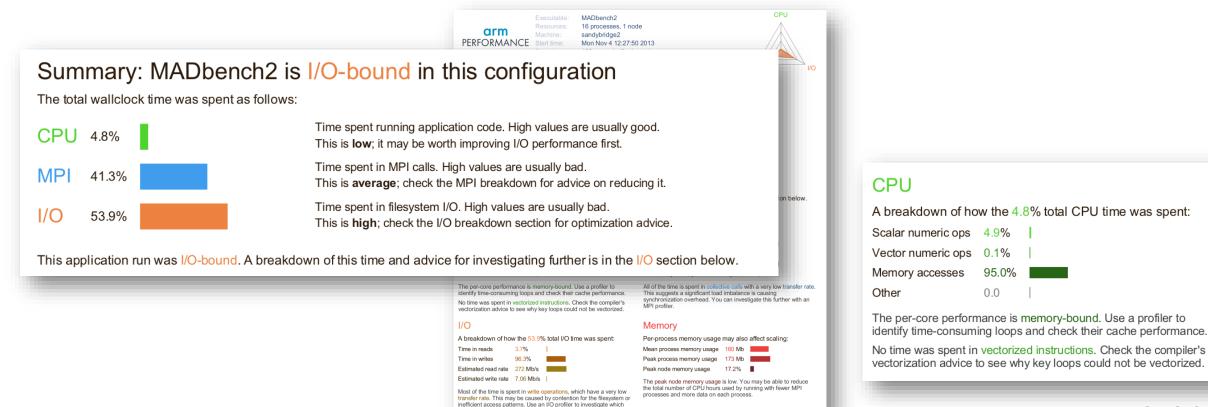




Identifying the amount of vectorized code

- Arm Performance Reports is an application reporting tool for HPC
 - Easy to use: no re-compiling required
 - Gives a comprehensible and readable summary of the application behavior

write calls are affected.



Analyze the results

Running Performance Reports with CloverLeaf using 8 MPI tasks indicates that:

- Time spent in scalar ops is 14.7%
- Time spent in vector ops 18.9%

Summary: clover_leaf is Compute-bound in this configuration

Compute 93.4%	Time spent running application code. High values are usually good. This is very high ; check the CPU performance section for advice
MPI 6.6%	Time spent in MPI calls. High values are usually bad. This is very low ; this code may benefit from a higher process count
I/O 0.0%	Time spent in filesystem I/O. High values are usually bad. This is negligible ; there's no need to investigate I/O performance

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

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

CPU 🔶			
A breakdown of th	ne 93.49	% CPU time:	
Scalar numeric ops	14.7%	1	
Vector numeric ops	18.9%		
Memory accesses	66.3%		

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

Little time is spent in vectorized instructions. Check the compiler's

vectorization advice to see why key loops could not be vectorized.

Π.	л	
1.	/ 1	

A breakdown of the 6.6% MPI time:				
Time in collective calls	20.9%			
Time in point-to-point calls	79.1%			
Effective process collective rate	1.55 kB/s			
Effective process point-to-point rate	33.1 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.

When? Time to use a profiler

Arm MAP is a lightweight multi-node profiling tool

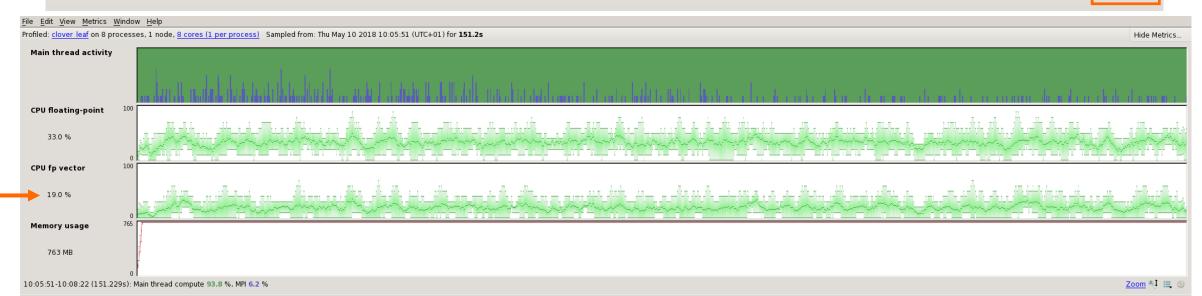
- Compiling with debugging flag required
- Shows processes and threads activity over time
- Source code is annotated
- Information aggregated by stacks and function

Compute, IO and MPI

		The Marc 10 2019 00.41 22 (107 101) for 143 6-	
	a. 8 cores (1 per process) Sampled from:	Thu May 10 2018 09:41:23 (UTC+01) for 143.6s	Hide
Main thread activity			
- 1 T			
all should			
CPU floating-point 100	a ba bibli a carrie the dirit in the fe	n de sterne na selente a la constante de la constante de la constante de la constante de la desta de la constante de la constan	name and a company of the state
CPU floating-point 100		aller statistics and the contract of the second state of the statistic statistics and the second by generative	
÷		A REAL PROPERTY AND A REAL	e - mail a desarra anda de la caralla addedada
30.5 %			and
2.72			
CPU fp vector			
cro ip vector		1	1 1
11 A		A CALL AND A REAL AND A	
27.2 %	in the fight of the second states of the	The state of the s	and the second second state of the second
Memory usage 766 7			
include a suge			
764 MB			
704 198			
19-41-23-09-43-46 (1.43-577s) Main thread	d compute 93.2 %, MPI 6.8 %, File I/O 0.0 %		Zoom
/ hydro.f90 🖸 🧧 / clover_leaff90 🗔 📔			Time spent on line 62
	timerstart - timer()		Breakdown of the 56.6% time spent on this
			Executing instructions 0.0%
	1 00		Calling other functions 100.0%
	step_time - timer()		
	step = step + 1		
12 Westernet Arthrough and 11			
12.29 udministration in the considerabilities of	CALL timestep()		
12.29 almatical decombined days 12.29 almatical decombined days 12.29 almatical control of the second			
	CALL timestep()		
10.25 annial anti-analogical 4.05 anti-analogical anti-analogical 50	CALL timestep() CALL PdV(.TROE.) CALL accelerate()		
10.23 attentiskingstreationistic angeling 53 42.03 accelerationistic angeling 55 62.03 accelerationistic angeling 55 62.03 accelerationistic angeling 55	CALL timestep() CALL PdV(.TRUE.) CALL accelerate() CALL PdV(.PALSE.)		
 23 constitutions to a state of the second sec	CALL timestep() CALL PdV(.TRDE.) CALL accelerate() CALL pdV(.FALSE.) CALL flux_calc()		1
10.23 attentiskloventosistasonolin 53 42.03 secolacionistationistasonolin 55 52.03 secolacionistationistasonolinista 55	CALL timestep() CALL PdV(.TRUE.) CALL accelerate() CALL PdV(.PALSE.)		
 Construction of the state of th	CALL timestep() CALL PdV(.TRDE.) CALL accelerate() CALL pdV(.FALSE.) CALL flux_calc()		
 23 constitutions to a state of the second sec	CALL timestep() CALL PdV(.TRUE.) CALL PdV(.TRUE.) CALL PdV(.FALSE.) CALL flw_calc() CALL stweetion() CALL reset_field()		
 10.2 V eternitektiveniteetentoonoonalis 4.4 V eternitektiveniteeten 4.4 V eternitektiveniteiten 4.4 V eternitektiven 4.4 V eternitektiv	CALL timestep() CALL PaV(.TRDE.) CALL accelerate() CALL accelerate() CALL flux_calc() CALL flux_calc() I CALL advection()	×	
$\label{eq:approximate} \begin{array}{l} 0 & \text{attraction} and out of the optimization of the set o$	CALL timestep() CALL pdV(TADE.) CALL scolerate() CALL pdV(TALSE.) CALL fluc_calc() CALL fluc_calc() CALL reset_field() advect_x = .NOT. advect	×	
attentiskin stational dia astalia 4.0 maximum dia materia dia dia dia dia dia dia dia dia dia d	CALL timestep() CALL pdV(TADE.) CALL scolerate() CALL pdV(TALSE.) CALL fluc_calc() CALL fluc_calc() CALL reset_field() advect_x = .NOT. advect	×	
4.9 Sector Secto	CALL timestep () CALL POP(TRDE,) CALL POP(TRDE,) CALL POP(TALDE,) CALL fluc,calc() CALL fluc,calc() CALL freet_field() advect_x = .MOT. advect Stacks Functions		
4.9 Sector Secto	CALL timestep() CALL pdv(TREE.) CALL scoreirate() CALL scoreirate() CALL flux_calc() CALL flux_calc() CALL starter(Ini) CALL starter(Ini) CALL score_Ini() advect_x = ,NOT. advect Starter(Ini) Scall score (Ini) Scall score (Ini)	Source	
4.9 Sector Secto	CALL timestep() CALL POP(,TROE.) CALL POP(,TROE.) CALL POP(,FALSE.) CALL POP(,FALSE.) CALL fun_call() CALL fun_call() CALL function CALL function CALL function Finction() on the B 2 chorer tad[recorm]	Source	
 Setteration and a set with - Setteration and a set with - Setteration and a set with setteration - Setteration and a set with setteration - Setteration and setteration and setteration - Seteration - Setteration	CALL timestep() CAL POV(TRDE.) CALL POV(TRDE.) CALL POV(TRDE.) CALL POV(TRLE.) CALL fluc,calc() CALL fluc,calc() CALL fluc,calc() CALL fluc,calc() CALL freet_fleid() advect_k = .807. advect 450xks Functions Functions E J Chart Jud (popul) B Undo	Source PRODAM clows_leaf CALL hypro	clover_leaff90:39 clover_leaff90:72
15 of a standard with non-blane skin 15 of the standard with non-blane skin 15 of the standard with standard with the 15 of the standard with standard with the 15 of the standard with standard with the 15 of the standard with the 15 of the standard with the 15 of the 15 o	CALL timestep() CALL POV(TROE) CALL POV(TROE) CALL POV(TROE) CALL POV(TALGE.) CALL fou_cal() CALL fou_cal() CALL fou_cal() CALL rest_field() advect_x = _MOT. advec Stacks four lad fou	Source Source Source State Sta	clover_leaf90.39 clover_leaf90.72 hydro190.62
10.25 started lands hand size size 10.25 started lands that site size 10.25 started lands that site size 10.25 started lands that site size 10.25 started lands to size size 10.25 started lands 10.25 st	CALL timestep () CALL prof(TMDE.) CALL prof(TMDE.) CALL prof(TMLE.) CALL prof(TMLE.) CALL file(call() CALL file(call() CALL rest_field() advect_x = .807. advec dStats Proctoms Proctoms Proctoms	Searce FRODBA clove_last RODBA clove_last ROD	clover_leaff90.39 clover_leaff90.72 hydro 100.62 hydro 100.52
 A startistic for sith on the set with a site of the set of the s	CALL timestep () CALL POP(TROE) CALL POP(TROE) CALL POP(TROE) CALL POP(TALSE.) CALL POP(TAL	Source Source Source State Sta	clover_leaff90.39 clover_leaff90.29 hydra f00.52 hydra f00.54 hydra f00.54
1.5 started lands and shares kin 1.5 started lands that the land share 1.5 started lands that the land shares 1.5 started lands 1.5 started 1.5 star	CALL timestep () CALL prof(TREE) CALL prof(TREE) CALL prof(TRLEE) CALL prof(TRLEE) CALL prof(TRLEE) CALL flux_calc() CALL rest_field() advect_w = .800% .advect d Stacks Factors: advect_w = .800% .advect d Stacks b / close; ball b / close; b	Source FRODAD slove_last FRODAD slove_last KAL slove_plot	clover_leaf50.39 down leaf50.72 hydroff0.72 hydroff0.52 hydroff0.53 hydroff0.54 hydroff0.54
1977 A startistic for state and durates also 4.19 months and the state state of the state also 4.19 months and the state state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 4.19 months and the state of the state of the state of the state 5.20 months and the state of the state of the state of the state 5.20 months and the state of the state of the state of the state 5.20 months and the state of the state of the state of the state 5.20 months and the state of the state of the state of the state 5.20 months and the state of the state of the state of the state 5.20 months and the state of	CALL timestep () CALL POP(_TROP.) CALL POP(_TROP.) CALL POP(_TROP.) CALL POP(_TRIP.) CALL fluc_said() CALL fluc_said() CALL fluc_said() CALL rest_field() advect_x = _MOP. advect Stacks Functions Functions Functions CALL POP(_TRUP.) CALL	Source Source Stock closes_isst CAL byte CAL byte CAL storetup() CAL bit(rot(r)) CAL bit(r)) CAL bit(rot(r)) CAL bit(rot(r)) CAL bit(r)) CAL b	clover_leaffr0.39 clover_leaffr0.27 hydra506.72 hydra506.32 hydra506.34 hydra506.34 hydra506.34 hydra506.35 hydra606.35 hydra606.34
1.5 started lands and shares kin 1.5 started lands that the land share 1.5 started lands that the land shares 1.5 started lands 1.5 started 1.5 star	CALL timestep () CALL Part(TREE) CALL PART(TRE	Source FRODAD slove_last FRODAD slove_last KAL slove_plot	Clover_Jeaff50.35 clover_Jeaff50.25 Nyder505.22 Nyder505.22 Nyder505.23 Nyder505.25 Nyder505.25

How much of the code is vectorized?

Profiled: clover leaf on 8 processes, 1 node, 8 cores (1 per process) Sampled from: Thu May 10 2018 10:05:51 (UTC+01) for 151.2s



Main Thread Stacks	Main Thread Stacks			Main Thread Stacks			
Total core time		Function(s) on line	Source	Total core time ∇ M		Function(s) on line	Source
Iotal core une	V MIET		Source		E	🗄 🥩 clover_leaf [program]	
		🖻 🥩 clover_leaf [program]				🖻 💉 clover_leaf	PROGRAM clover_leaf
		🖻 🥖 clover leaf	PROGRAM clover_leaf			🖻 hydro	CALL hydro
53.3%	2.0%	advection module::advection	CALL advection()			🖻 advection_module::advection	CALL advection()
16.5% Juli - Million - Mil		timestep module::timestep		11.100 all to to all has all a			. CALL advec_mom_driver(tile,xvel,direction,sweep_number)
			CALL timestep()				. CALL advec_mom_kernel(chunk%tiles(tile)%t_xmin, & . CALL advec_mom_driver(tile,xvel,direction,sweep_number)
	nhanniki hatu kinalilin 0.5%	🖻 pdv_module::pdv	CALL PdV(.TRUE.)	9.8% kalina damata kutaka			CALL advec_mom_ariver(tile,xvel,direction,sweep_number) CALL advec_cell_driver(tile,sweep_number,direction)
6.4% <u>dintalihilati</u>	Litter and the second	i pdv_module∷pdv	CALL PdV(.FALSE.)	9.1% totottisme de biogranationationalitate			CALL advec_cell_driver(tile, sweep_number, direction) CALL advec_cell_driver(tile, sweep_number, direction)
4.9% _(dotated)	htaduutaduk, abhutu dadattat	i accelerate module∷accelerate	CALL accelerate()	7.1% all the star of a tribute i.i.i.i.i. bad with the balance inter-			. CALL advec_mom_driver(tile, yvel, direction, sweep_number)
4.2% and and the second second		i reset field module∷reset field	CALL reset_field()	6.6% automotives and distantial addated birth			. CALL advec_mom_driver(tile,yvel,direction,sweep_number)
3.4% Hatte Labort delte la		∃flux calc module::flux calc	CALL flux calc()	1.7% month also total a state 1	.7%	update_halo_module::update_halo	CALL update_halo(fields,2)
			CALL IIUX_Call()	0.9%	0.7%		
1.2%		並 4 others		12.2% Internal data instantia data 3	3.2%	itimestep_module∷timestep	CALL timestep()
				10.2% ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		i pdv_module∷pdv	CALL PdV(.TRUE.)
				6.5% taladalahininadalahinintanadalahinintan	0.7%	i pdv_module∷pdv	CALL PdV(.FALSE.)
				4.8% others.thdars.bib.dor.bib.attil.attb.anorbal.attb.att		accelerate_module::accelerate	CALL accelerate()
				4.2% 🗤 name o availabilitation distributed and the second		reset_field_module∷reset_field	CALL reset_field()
				4.2%	-0.1.0/	flux_calc_module∷flux_calc	CALL flux_calc()
				0.6%	<0.1%	i ∃ others ⊡ 1 other	

Showing data from 8,000 samples taken over 8 processes (1000 per process)

Where is the code vectorized?

0 🖂 📗	📧 advec_mom_driver.f90 📧 🛛 💽 advec_mom_kernel.f90 🔯 🖊	Time spent on line 159 🗗 🗡
150	dif=donor	Breakdown of the 0.1% time spent on this line:
151	ELSE	
152	upwind=j-1	Executing instructions 100.0%
153	donor=j	Calling other functions 0.0%
154	downwind=j+1	
155	dif=upwind	Time in instructions executed:
156	ENDIF	Scalar floating-point 63.6%
157	<pre>sigma=ABS(node_flux(j,k))/(node_mass_pre(donor,k))</pre>	
158	width=celldx(j)	51
159	vdiffuw=vel1(donor,k)-vel1(upwind,k)	Scalar integer 18.2%
160	vdiffdw=vel1(downwind,k)-vel1(donor,k)	Vector integer 0.0%
161	limiter=0.0	Memory access* 81.8%
162 🗉	IF (vdiffuw*vdiffdw.GT.0.0) THEN	
163	auw=ABS(vdiffuw)	Branch 0.0%
164	adw=ABS(vdiffdw)	Other instructions 0.0%
165 166	wind=1.0_8	* 10 DW memory and instructions. CD CW involution and a state
167	IF(vdiffdw. LE .0.0) wind=-1.0_8 limiter=wind* MIN (width*((2.0 8-sigma)*adw/widt	* 18.2% memory access instructions, 63.6% implicit memory accesses in other
168	ENDIF	instructions, also counted in their categories
169	advec_vel_s=vel1(donor,k)+(1.0-sigma)*limiter	
170	<pre>mom_flux(j,k) = advec_vel_s*node_flux(j,k)</pre>	
171	ENDDO	
172	ENDDO	
173	SOMP END DO	
174	I SOMP DO	

Follow Performance Reports advice

Main Thread Stacks	-	1	1
Total core time 🗸 🗸	MPI	Function(s) on line	Source
		🖻 🥵 clover_leaf [program]	
		🖻 🥖 clover_leaf	PROGRAM clover_leaf
		🛱 hydro	CALL hydro
		🖨 advection_module :: advection	CALL advection()
			CALL advec_mom_driver(tile,xvel,direction,sweep_numb
			CALL advec_mom_kernel(chunk%tiles(tile)%t_xmin, &
10.2% manhalandin halanda		advec_mom_driver_module::advec	
9.8% ladamana dalah katalah kat		advec_cell_driver_module::advec	CALL advec_cell_driver(tile,sweep_number,direction)
9.1% mmmillend fahlunnaltunationalilini		advec_cell_driver_module::advec	
7.1 % addatational addatation and interface of the state		advec_mom_driver_module::advec	
6.6% automotives and dialog the device of th		🖻 advec_mom_driver_module::advec	
1.7% annouse and solar restrictions are a second	1.7%	🗈 update_halo_module ::update_halo	CALL update_halo(fields,2)
0.9%	0.7%		
12.2% (1996) 1996) 1996 (1996)		i timestep_module∷timestep	CALL timestep()
10.2% <u>adddaddadadaanaandaanaanaanaa</u>		i pdv_module∷pdv	CALL PdV(.TRUE.)
6.5% pain databilities dan berte berte and barbart.	ել 0.7%	pdv_module::pdv	CALL PdV(.FALSE.)
4.8% attact their chine in a data in a state of the second s		accelerate_module::accelerate	CALL accelerate()
4.2% 🖬 սոստ և ստովելի հանդիմ։ Ասկելինելին	dit .	reset_field_module::reset_field	CALL reset_field()
4.2% չվանվակերութունին անգային Մնուժովիսիս		flux_calc_module::flux_calc	CALL flux_calc()
0.6%	<0.1%		
0.6%	0.1%	🗄 1 other	

Showing data from 8,000 samples taken over 8 processes (1000 per process)

Follow Performance Reports advice

advec_mom_kernel.f90

- •••
- 144 DO k=y_min,y_max+1
- 145 DO j=x_min-1,x_max+1 ←
- 146 IF(node_flux(j,k).LT.0.0)THEN
- 147 upwind=j+2
- 148 donor=j+1
- 149 downwind=j
- 150 dif=donor
- 151 ELSE
- 152 upwind=j-1
- 153 donor=j
- 154 downwind=j+1
- 155 dif=upwind
- 156 ENDIF
- 157 sigma=ABS(node_flux(j,k))/(node_mass_pre(donor,k))
- 158 width=celldx(j)
- 159 vdiffuw=vel1(donor,k)-vel1(upwind,k) ←
- 160 vdiffdw=vel1(downwind,k)-vel1(donor,k)

-fopt-info-vec-missed

advec_mom_kernel.f90:145: note: not vectorized: control flow in loop advec_mom_kernel.f90:145: note: bad inner-loop form. advec_mom_kernel.f90:145: note: not vectorized: Bad inner loop. advec_mom_kernel.f90:145: note: bad loop form. Analyzing loop at advec_mom_kernel.f90:145

advec_mom_kernel.f90:145: note: not vectorized: control flow in loop advec_mom_kernel.f90:145: note: bad loop form.

...

How well is the compiler vectorizing?

advec_mom_kernel.f90

- •••
- 144 DO k=y_min,y_max+1
- 145 DO j=x_min-1,x_max+1 ←
- 146 IF(node_flux(j,k).LT.0.0)THEN
- 147 upwind=j+2
- 148 donor=j+1
- 149 downwind=j
- 150 dif=donor
- 151 ELSE
- 152 upwind=j-1
- 153 donor=j
- 154 downwind=j+1
- 155 dif=upwind
- 156 ENDIF
- 157 sigma=ABS(node_flux(j,k))/(node_mass_pre(donor,k))
- 158 width=celldx(j)
- 159 vdiffuw=vel1(donor,k)-vel1(upwind,k)
- 160 vdiffdw=vel1(downwind,k)-vel1(donor,k)

-qopt-report=2

LOOP BEGIN at advec_mom_kernel.f90(145,9) <Peeled loop for vectorization> remark #25456: Number of Array Refs Scalar Replaced In Loop: 2 LOOP END

LOOP BEGIN at advec_mom_kernel.f90(145,9) remark #15300: LOOP WAS VECTORIZED LOOP END

LOOP BEGIN at advec_mom_kernel.f90(145,9) <Remainder loop for vectorization> LOOP END

...

Analyze the results

Running Performance Reports with CloverLeaf using 8 MPI tasks indicates that:

- Time spent in scalar ops is 4.8%
- Time spent in vector ops 28.2%

Summary: clover_leaf is Compute-bound in this configuration

Compute	92.9%	Time spent running application code. High values are usually good. This is very high ; check the CPU performance section for advice
MPI	7.1%	Time spent in MPI calls. High values are usually bad. This is very low ; this code may benefit from a higher process count
I/O	0.0%	Time spent in filesystem I/O. High values are usually bad. This is negligible ; there's no need to investigate I/O performance

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

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

CPU

A breakdown of th	ne 92.99	% CPU time:
Scalar numeric ops	4.8%	I
Vector numeric ops	28.2%	
Memory accesses	67.0%	

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 7.1% MPI time:						
Time in collective calls	24.4%					
Time in point-to-point calls	75.6%					
Effective process collective rate	1.35 kB/s	1				
Effective process point-to-point rate	33.9 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.

Where is the code vectorized?

				Time spent on line 15	50	5
🚺 🗉 advec mom driver.f90 🖂 🛛 🗉 adv	ec mom kernel.f90 🖂 丨			Time spent on line 13	29	
2 upwind=j-1						
3 donor=j				Breakdown of th	ne 0.4% time spent on this	: line:
4 downwind=j+1				Dieakaowii oi ti	ie 0.4% time spent on time	, mie.
5 dif=upwind						
6 ENDIF 7 sigma= ABS (node flux	:(j,k))/(node_mass_pre(donor,k))			Executing instruction	ons 100.0%	
8 width=celldx(j)	(j,k))/(node_mass_pre(donor,k))			5		
9 vdiffuw=vel1(donor,				Calling other function	ons 0.0%	
vdiffdw=vel1(downwi	nd,k)-vel1(donor,k)			canning outer ranear	0.070	
1 limiter=0.0 2 ⊡ IF(vdiffuw*vdiffdw.	CT 0 0) THEN					
auw= ABS (vdiffuw)	GI:0.0) THEN			Time in instructi	ions executed:	
4 adw=ABS(vdiffdw)						
5 wind=1.0_8				Scalar floating poin	nt 0.0%	
IF (vdiffdw. LE.0.0 limiter=wind*MIN) wind=-1.0_8 (width*((2.0_8-sigma)*adw/width	(1.0.8) cigma) * auto(co	11dr/dif)/608 ave adv	Scalar floating-poin	L 0.0%	
ENDIF	(widin*((2.0_6-signa)*adw/widin	+(1.0_0+sigma) ~auw/ce	IIdx(dII))/8.0_8,auw,adw)			
	nor,k)+(1.0-sigma)*limiter			Vector floating poin	nt 28.6% 👥 🖊 🔶	
<pre>mom_flux(j,k)=advec</pre>	_vel_s*node_flux(j,k)			5.		
ENDDO ENDDO				Scalar integer	0.0%	
\$0MP END DO				beanar meeger	0.070	
I \$OMP DO				Vector integer	0.0%	
<pre>DO k=y_min,y_max+1</pre>				vector integer	0.070	
DO j=x_min,x_max+1 vel1 (j,k)=(vel1 (j	,k)*node_mass_pre(j,k)+mom_flux	(j-1,k) -mom flux (j,k)	/node mass post (i k)	Mamanyaccac	60.6%	
ENDDO	(x) hode_mass_pre(j,x) hhom_rrax	(j 1, k) mom_11dk(j, k);	// node_mass_pose (j, k)	Memory access	68.6%	
ENDDO						
I SOMP END DO				Branch		
ELSEIF (direction. EQ .2) TH IF (which_vel. EQ .1) THEN	EN					
				Other instructions	2.9%	
			Main Thread Stacks	outer moductions	2.070	
!\$OMP DO □ DO k=y_min-2,y_max+2			r			
<pre>!\$OMP DO DO k=y_min-2,y_max+2 DO j=x_min,x_max+1</pre>			Total core time $ abla MPI$			
I \$0MP DO DO k=y_min-2,y_max+2 DO j=x_min,x_max+1 hread Stacks			Total core time ∇ MPI			
i \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ∇ MPI	Function(s) on line	Source				
I \$0MP DO DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ∇ MPI	🛛 🦻 clover_leaf [program]		10.9% and kinternation in the statistical and the statistical statistical statistics of the stat	e advec mom driver module sadvec CAL	L advec_mom_driver(tile,xvel,direction,sweep_number)	
I \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ∇ MPI	3 Ø clover_leaf [program] ⋵ ∮ clover_leaf	PROGRAM clover_leaf	10.9% and demonstrated when proton to all		<pre>L advec_mom_driver(tile,xvel,direction,sweep_number) L advec_mom_driver(tile,xvel,direction,sweep_number) L advec cell driver(tile,sweep number,direction)</pre>	
DO k=y_min-2,y_max+2 DO j=x_min,x_max+1 Thread Stacks core time ∇ MPI	a ✿ clover_leaf [program] 亩 ✔ clover_leaf 亩 hydro	PROGRAM clover_leaf CALL hydro	10.9% stadt dastrasti kan för ähningat i andra st 9.7% stadssa das andra stadt stadt stadt stadt stadt stadt 8.9% som kan stadt för stadt stadt stadt stadt stadt stadt 8.5% som kan stadt stadt stadt stadt stadt stadt stadt stadt	advec_cell_driver_module::advec_ce call advec_cell_driver_module::advec_ce call	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction)	
I \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ▼ MPI E	a ♀ clover_leaf [program] ia ✔ clover_leaf ia hydro ia advection_module::advection	PROGRAM clover_leaf CALL hydro CALL advection()	10.9% staktiskuusta kardisteiheinenteentimisessä 9.7% suotaasadaadestilleitteetti kuista kuista 8.9% asadaasada kuista kuista kuista kuista 8.5% aantaasad ataatuttiitteitteitteitteitteitteitteitteitt	advec_cell_driver_module::advec_ce call advec_cell_driver_module::advec_ce call advec_mom_driver_module::advec call	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction) L advec_mom_driver(tile,yvel,direction,sweep_number)	
I \$0MP DO DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ♥ MPI 6% [maximum block [maxi	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::timestep 	PROGRAM clover_leaf CALL hydro CALL advection() CALL timestep()	10.9% stable destruction has a log binder through the barrier of the state of 9.7% to be access of a state of the state of	B advec_cell_driver_module::advec_ce CALI B advec_cell_driver_module::advec_ce CALI B advec_mom_driver_module::advec CALI B advec_mom_driver_module::advec CALI B advec_mom_driver_module::advec CALI	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction)	
i \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ✓ MPI 6.6% MPI 2.2% Jubit to main the back to m	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::timestep pdv_module::pdv 	PROGRAM clover_leaf CALL hydro CALL advection() CALL timestep() CALL PdV(.TRUE.)	10.9% staltidenes in the strip in the strip in the strip of the state of the strip	B advec_cell_driver_module::advec_cecAll B advec_cell_driver_module::advec_cecAll B advec_mom_driver_module::adveccAll B advec_mom_driver_module::adveccAll B advec_mom_driver_module::adveccAll Update_halo_module::update_halo CAll D update_halo_module::update_halo CAll D update_nalo_module::update_halo CAll D update_halo_module::update_halo CAll D update_halo D update_halo	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction) L advec_mom_driver(tile,yvel,direction,sweep_number) L advec_mom_driver(tile,yvel,direction,sweep_number) L update_halo(fields,2)	
1 \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ▼ MPI .6% 100 mmmulation of the second state of the	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::timestep pdv_module::pdv pdv_module::pdv 	PROGRAM clover_leaf CALL hydro CALL advection() CALL timestep() CALL PdV(.TRUE.) CALL PdV(.FALSE.)	10.9% and demonstrates for the interaction is a state 9.7% attacks to a state of the interaction is a state of the interaction 8.9% and a state of the interaction is an attack of the 8.5% and a state of the interaction is a state of the interaction 6.8% and a state of the interaction is a state of the interaction 6.2% and a state of the interaction is a state of the interaction 1.5% attacks at the interaction is a state of the interaction 1.5% attacks at the interaction is a state of the interaction 1.5% attacks at the interaction is a state of the interaction is a state 0.6% attacks at the interaction interaction is a state of the interaction 0.6% attacks at the interaction is a state of the interaction in the interaction is a state 0.6% attacks attacks at the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interaction is a state of the interaction in the interac	B advec_cell_driver_module::advec_ce cALL B advec_cell_driver_module::advec_ce cALL B advec_mom_driver_module::advec cALL B advec_mom_driver_module::advec cALL B update_halo_module::update_halo B update_halo_module::update_halo B timestep_module::timestep CALL	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction) L advec_mom_driver(tile,yvel,direction,sweep_number) L advec_mom_driver(tile,yvel,direction,sweep_number) L update_halo(fields,2) L timestep()	
1\$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ✓ MPI 6.6% Construction of the second state of the second s	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::timestep pdv_module::pdv pdv_module::pdv accelerate_module::accelerate 	PROGRAM clover_leaf CALL hydro CALL advection() CALL fumestep() CALL PdV(.TRUE.) CALL PdV(.FALSE.) CALL accelerate()	10.9% to be to be a set of the se	B advec_cell_driver_module::advec_ce CALL B advec_cell_driver_module::advec_ce CALL B advec_mom_driver_module::advec CALL B advec_mom_driver_module::advec CALL B update_halo_module::update_halo B z others B timestep_module:timestep CALL B pdv_module::pdv CALL	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction) L advec_mom_driver(tile,yvel,direction,sweep_number) L advec_mom_driver(tile,yvel,direction,sweep_number) L update_halo(fields,2)	
i \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time MPI .6% [Intercontrol of the optimal for t	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::bimestep pdv_module::pdv pdv_module::pdv accelerate_module::accelerate reset_field_module::reset_field 	PROGRAM clover_leaf CALL hydro CALL advection() CALL timestep() CALL PdV(.TRUE.) CALL PdV(.FALSE.) CALL accelerate() CALL reset_field()	10.9% and demonstrates for the instruments of a state 9.7% accessed and an annual state of the instruments of the state 8.9% and and demonstrates and the state of the state 8.8% and and demonstrates and the state of the state 6.8% and and demonstrates and the state of the state 1.5% accessed and the state of the state of the state 1.5% accessed and the state of the state of the state 1.5% accessed and the state of the state of the state of the 1.5% accessed accessed and the state of the state of the 1.5% accessed accessed and the state of the state of the 1.5% accessed accessed accessed and the state of the state of the 1.5% accessed	⊕ advec_cell_driver_module::advec_ce cALL ⊕ advec_cell_driver_module::advec_ce cALL ⊕ advec_mom_driver_module::advec cALL ⊕ advec_mom_driver_module::advec cALL ⊕ advec_mom_driver_module::advec cALL ⊕ update_halo_module::update_halo cALL ⊕ 2 others btimestep_module::ubvec cALL ⊕ pdv_module::pdv cALL ⊕ pdv_module::pdv cALL ⊕ accelerate_module::accelerate cALL	L advec_cell_driver(tile, sweep_number, direction) L advec_cell_driver(tile, sweep_number, direction) L advec_mom_driver(tile, yvel, direction, sweep_number) L advec_mom_driver(tile, yvel, direction, sweep_number) L update_halo(fields, 2) L timestep() L PAV(.TRUE.) L PAV(.TRUE.) L accelerate()	
I \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::timestep pdv_module::pdv pdv_module::pdv accelerate_module::accelerate reset_field_module::reset_field flux_calc_module::flux_calc 	PROGRAM clover_leaf CALL hydro CALL advection() CALL fumestep() CALL PdV(.TRUE.) CALL PdV(.FALSE.) CALL accelerate()	10.9% to be a set of the set of t	⊞ advec_cell_driver_module::advec_ce cALL ⊞ advec_mom_driver_module::advec_ce cALL ⊞ advec_mom_driver_module::advec cALL ⊞ advec_mom_driver_module::advec cALL ⊞ advec_mom_driver_module::advec cALL ⊞ advec_mom_driver_module::advec cALL ⊞ bpdate_halo_module::update_halo cALL ⊞ 2 others Btimestep_module::timestep cALL B pdv_module::pdv cALL B pdv_module::pdv cALL ⊞ accelerate_module::accelerate cALL B rest_field_module::reset_field cALL	L advec_cell_driver(tile,sweep_number,direction) L advec_cell_driver(tile,sweep_number,direction) L advec_mom_driver(tile,yvel,direction,sweep_number) L advec_mom_driver(tile,yvel,direction,sweep_number) L update_halo(fields,2) L timestep() L PAV(.FALSE.) L pav(.FALSE.) L reset_field()	
I \$0MP Do DO k=y_min-2, y_max+2 DO j=x_min, x_max+1 Thread Stacks core time ✓ MPI Core time ✓ MPI Core time O O MPI Core time Core time O O MPI Core time	 clover_leaf [program] clover_leaf hydro advection_module::advection timestep_module::timestep pdv_module::pdv pdv_module::pdv accelerate_module::accelerate reset_field_module::reset_field flux_calc_module::flux_calc 3 others 	PROGRAM clover_leaf CALL hydro CALL advection() CALL timestep() CALL PdV(.TRUE.) CALL PdV(.FALSE.) CALL accelerate() CALL reset_field()	10.9% and demonstrates for the instruments of a state 9.7% accessed and an annual state of the instruments of the state 8.9% and and demonstrates and the state of the state 8.8% and and demonstrates and the state of the state 6.8% and and demonstrates and the state of the state 1.5% accessed and the state of the state of the state 1.5% accessed and the state of the state of the state 1.5% accessed and the state of the state of the state of the 1.5% accessed accessed and the state of the state of the 1.5% accessed accessed and the state of the state of the 1.5% accessed accessed accessed and the state of the state of the 1.5% accessed	⊞ advec_cell_driver_module::advec_ce cALL ⊞ advec_mom_driver_module::advec_ce cALL ⊞ advec_mom_driver_module::advec cALL ⊞ advec_mom_driver_module::advec cALL ⊞ advec_mom_driver_module::advec cALL ⊞ advete_halo_module::update_halo cALL ⊞ 2 others CALL ⊞ pdv_module::pdv cALL B pdv_module::pdv cALL ⊞ accelerate_module::accelerate cALL	L advec_cell_driver(tile, sweep_number, direction) L advec_cell_driver(tile, sweep_number, direction) L advec_mom_driver(tile, yvel, direction, sweep_number) L advec_mom_driver(tile, yvel, direction, sweep_number) L update_halo(fields, 2) L timestep() L PAV(.TRUE.) L PAV(.TRUE.) L accelerate()	



Different compilers may have different capabilities, but here are guidelines

- Remove conditionals inside loop
- Make sure that loop size is known on entry
- Pay attention to work on contiguous, unit-stride arrays
- Remove data dependencies to enable vectorization
- Use compiler directives to force loop vectorization

Conclusion

Vectorizing an application is a difficult task

Arm Performance Reports and Arm MAP make it easier

- Analyze application efficiency and get advices with Performance Reports
- Identify bottlenecks and line by line performance with MAP

Figure out quickly if your application uses vectorization

Find candidates for vectorization

Inspect vectorization over time

Hands – On

- 2_profiling_compute
- Compile the code
- Is the code well vectorized ? (with Arm Performance Reports)
- Identify where and how it can be improved (with Arm MAP)
- Modify the code and recompile
- Has vectorization increased ? Do you see any speed-up ? (with Arm Performance Reports and Arm MAP)

Hands – On : Workload Imbalance

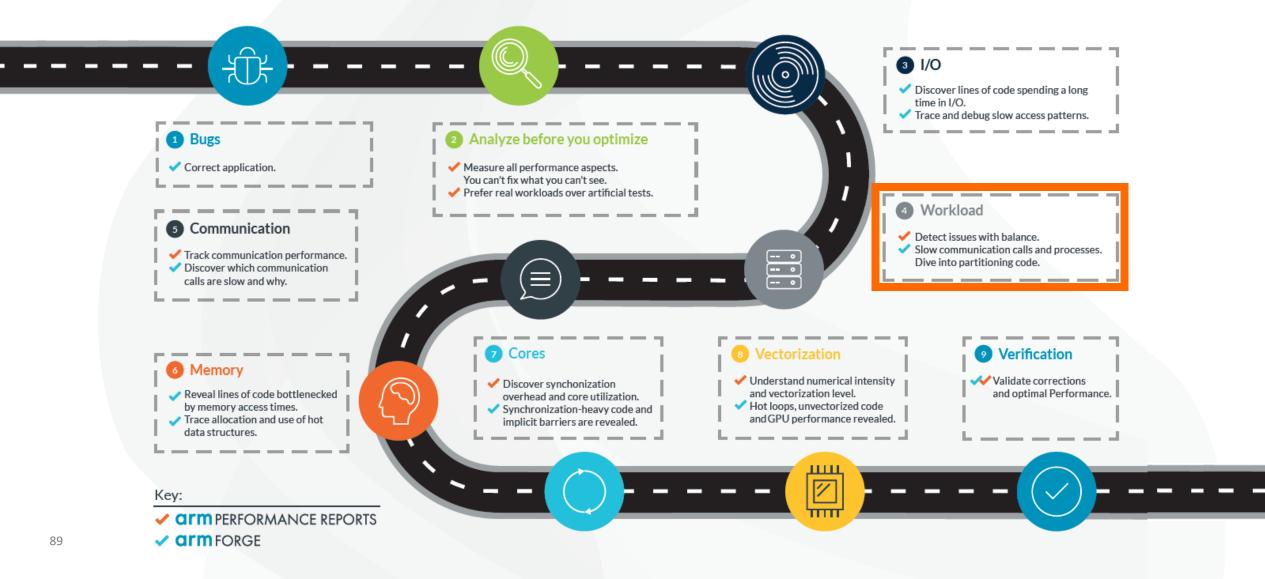


© 2017 Arm Limited

9 Step guide: optimizing high performance applications

arm

Improving the efficiency of your parallel software holds the key to solving more complex research problems faster. This pragmatic, 9 Step best practice guide will help you identify and focus on application readiness, bottlenecks and optimizations one step at a time.

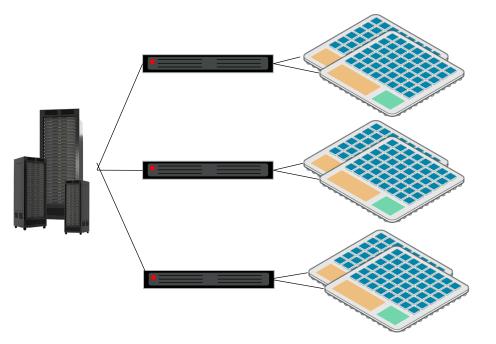


Workload balancing: definition

• *"Aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource."*

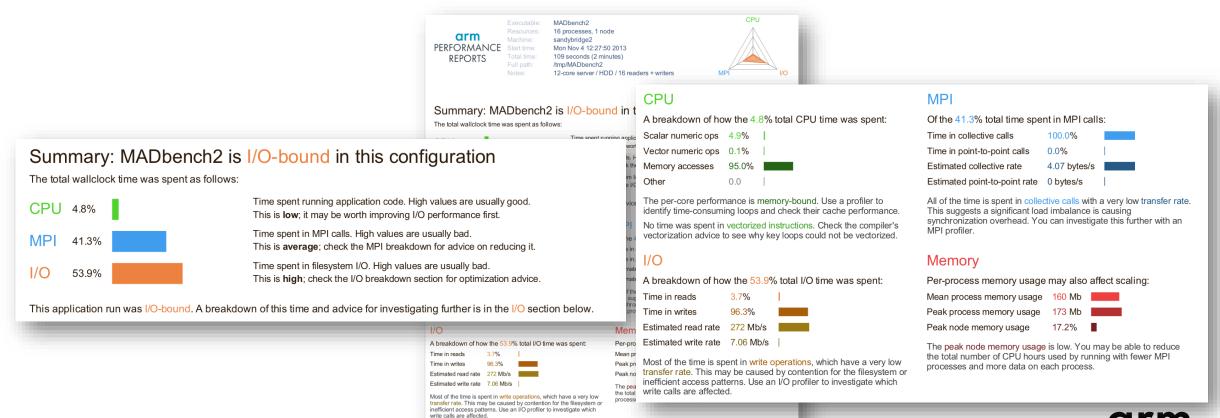
(Wikipedia)

- In HPC, a well balanced workload across:
 - Multiple nodes over a high-speed network,
 - Multiple sockets,
 - Multiple NUMA systems
 - Multiple cores,
 - Multiple accelerators,
 - Multiple disk drives,
- Is critical for application performance



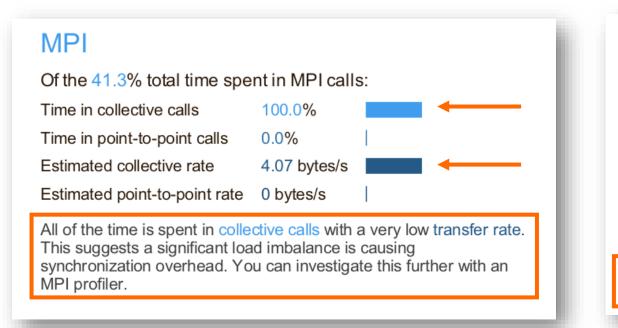
Identify workload imbalance

- Arm Performance Reports is an application reporting tool for HPC
 - Easy to use: no re-compiling required
 - Gives a comprehensible and readable summary of the application behavior



MPI and OpenMP imbalance

- Clues: excessive synchronization
 - MPI collective calls with no actual data transfer
 - Idle cores where threads are stuck in locks/mutexes



OpenMP

A breakdown of the 74.5% time in OpenMP regions:

Computation	53.6 %	
Synchronization	46.4%	
Physical core utilization	100.0%	
System load	78.0%	

Significant time is spent synchronizing threads in parallel regions. Check the affected regions with a profiler.

This may be a sign of overly fine-grained parallelism (OpenMP regions in tight loops) or workload imbalance.

Locate imbalance in your code

arm MAP

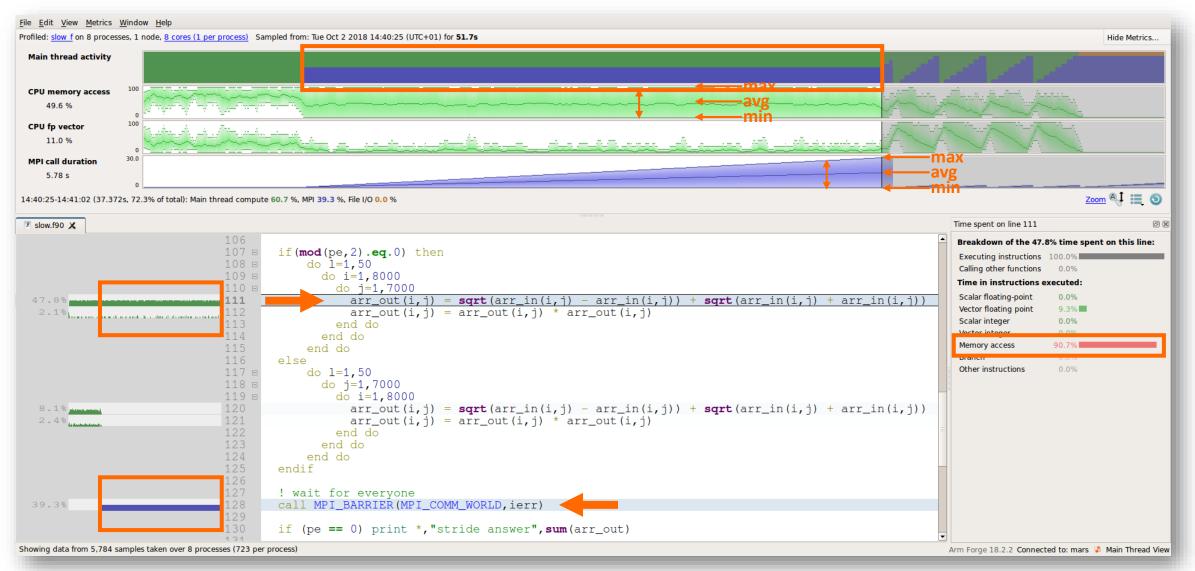
Arm MAP is a lightweight multi-node profiling tool

- Compiling with debugging flag required
- Shows processes and threads activity over time
- Source code is annotated
- Information aggregated by stacks and function

Compute, IO and MPI

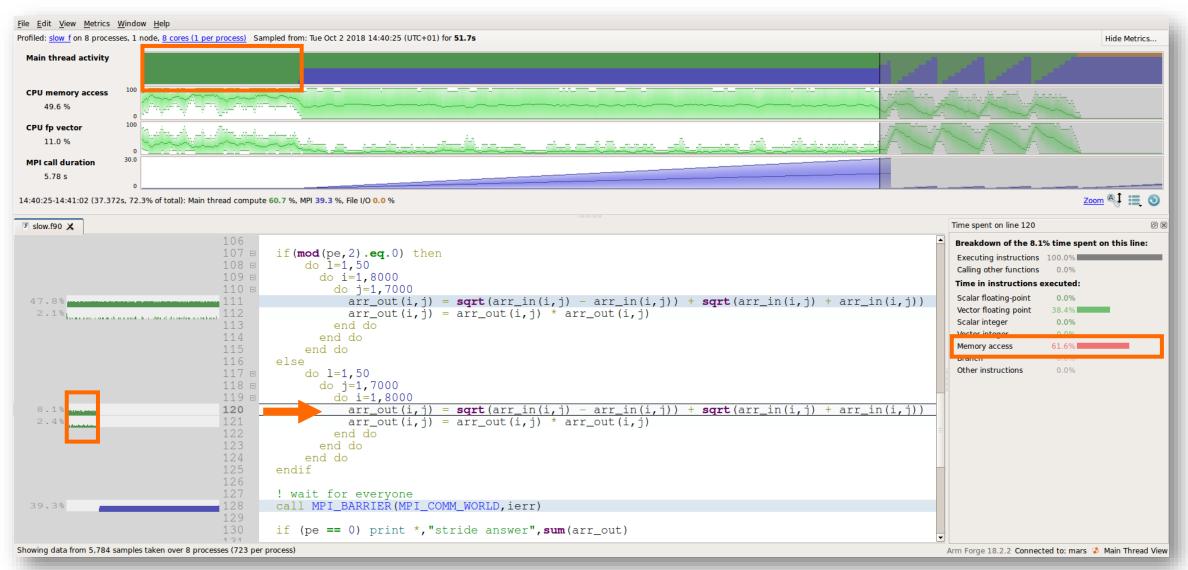
	sses, 2 nodes Samp	pled from: Thu Jul 9 2015 10:32:13 for 164	.95
Main thread activity	and of the	and the first of the second	a na ing ing ing mang na ing
CPU floating-point	100		
12.3 %	A		หม่นในการสารสารสารสารที่ได้นี้ที่ได้ได้ได้ได้ได้เป็นหม่าส่วนส่วนไปแต่ได้ได้ได้ได้ได้เป็นและการสารสาร
Memory usage	35.0		
30.3 MB			
50.5 MB	0		
10:32:13-10:34:57 (164.9	11s): Main thread co	ompute 0.3 %, OpenMP 19.6 %, MPI 63.2 %	6, File I/O 16.2 %, OpenMP overhead 0.5 %, Sleeping 0.2 %
🕫 main.c 🗙			
	275	}	
		#endif	
		$\frac{1}{2}$ // dt = 1.e-3;	
	278		stdout, "Hydro starts godunov.\n");
	280 日	if ((H.nstep % 2) == 0) {	cuole, nyulo starts goullov. (n /)
11.6%	281	hydro_godunov(1, dt, H, &Hv	r, &Hw_godunov, &Hvw_godunov);
	282	// hydro_godunov	(2, dt, H, &Hv, &Hw, &Hvw);
4.2	283 🕀	} else {	
13.3%	284		r, &Hw_godunov, &Hvw_godunov); r(1, dt, H, &Hv, &Hw, &Hvw);
		1 Injuro_gouunov	(1, dL, n, anv, anw, anvw),
		end_iter = dcclock();	
	288	cellPerCycle = (double) (H.gl	.obnx * H.globny) / (end_iter - start_iter) / 1000000.0L;
	289	avgCellPerCycle += cellPerCyc	:1e;
		nbCycle++;	
	291 292		
		U notoritt.	
		H.nstep++; H.t += dt:	
	292 293 E 294	H.nstep++; H.t += dt;	
Input/Output Project F	293 🗉 294	H.t += dt;	annama
	293 🗉 294	H.t += dt;	annama
Main Thread Stacks	293 E 294 Files Main Thread S	H.t += dt;	Source
Main Thread Stacks	293 E 294 Files Main Thread S	H.t += dt; Stacks Functions Dverhead Function(s) on line $\ominus \ \ $ hydro (program)	
Main Thread Stacks Total core time	293 E 204 Files Main Thread S MPI O	H.t += dt; Stacks Functions Overhead Function(s) on line ⊖ ♦ hydro [program] ⊖ ✓ main	Source main(int argc, char **argv) {
Main Thread Stacks Total core time	293 E 294 Files Main Thread S MPI Or MPI Or Mainschool of 29,9%	H.t += dt; Stacks Functions Overhead Function(s) on line □	<pre>Source main(int argc, char **argv) { MPI_Allreduce(sflopsAri, sflopsAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD);</pre>
Main Thread Stacks Total core time 29.9% second distance of the 23.7% second distance of the	293 E Files Main Thread S MIP Or 10 (29.9%) 10 (29.9%) 10 (29.9%)	B.t. += dt; Stacks Functions Dverhead Function(s) on line Dverhead Function(s) on line Dverh	<pre>Source main(int argc, char **argv) { MPI_Alireduce(sflopAri, sflopAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_NORLD); vtkfle(+Hvk, H, sHv);</pre>
Main Thread Stacks Total core time 29.9% https://docs.org/officers/ 23.7% anticipation/file/indianality/	293 704 Files Main Thread S MPI O 29.9% 11/4 (2)/46 7.9% 2.8% 0.	H.t += dt; Stacks Functions Stacks Function(s) on line ⊕ ⊕ hydro [program] ⊕ main HMP_Alfreduce ⊕ vtkfile ⊕ ydro godunov	<pre>Source main(int argc, char **argv) { MPI_Allreduce(&flopsAri, &flopsAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); vtkfile(++nvtk, H, 6Hv); hydro_godunov(2, dt, H, SHv, &FMw_godunov, &FHvw_godunov);</pre>
Main Thread Stacks Total core time 29.9% 23.7% and international factors 13.3%	223 704 Files Main Thread S MPI O 29.9% 10/4779/000 2.8% 0. 2.8% 0. 2.8% 0.	H.t += dt; Stacks Functions Dverhead Function(s) on line	<pre>Source main(int argc, char **argv) { MPI_Allreduce(&flopsAri, &flopsAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); vtkfile(++nvtk, H, &Hv); hydrc_godunov(2, dt, H, &Hv, &Hw_godunov, &Hvw_godunov); hydrc_godunov(1, dt, H, &Hv, &Hv, @dw_godunov, &Hvw_godunov); </pre>
Main Thread Stacks Total core time 29.9% 23.7% 13.3% 11.6%	293 794 Files Main Thread S MPI 0 104670306 2.8% 0. 2.8% 0. 2.5% 0. 2.5% 0. 2.5% 0. 2.5% 0. 2.5% 0. 2.5% 0. 2.5% 0. 2.5% 0. 2.8% 0. 2.	H.t += dt; Stacks Functions Dverhead Function(s) on line	<pre>Source main(int argc, char **argv) { MPI_Allreduce(sflopsAri, sflopsAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); vtkfle(+hrwtk, H, sHW); hydro_godunov(2, dt, H, sHW, sHW_godunov, sHWw_godunov); hydro_godunov(1, dt, H, sHW, sHW_godunov, sHWw_godunov); MPI_Allreduce(sflopsSqr_t, sflopsSqr_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); </pre>
Main Thread Stacks Total core time 29.9% sector time 23.7% sector time 13.3% 11.6% 8.0%	223 704 Files Main Thread S MPI Or Call States 7,9% duke States 7,9% 28.5% 0, 8.0% States 1,1% Call States 1,	H.t += dt; Stacks Functions Dverhead Function(s) on line	<pre>Source main(int argc, char **argv) { MPI_Allreduce(&flopsAri, &flopsAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); vtkfile(++nvtk, H, &Hv); hydro_godunov(2, dt, H, &Hv, &Hw_godunov, &Hvw_godunov); hydro_godunov(1, dt, H, &Hv, &Hw_godunov, &Hvw_godunov); MPI_Allreduce(&flopsEgr, &flopsEgr_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); MPI_Allreduce(&dt, &dtmin, 1, MPI_COMBLE, MPI_MIN, NPI_COMM_WORLD); </pre>
Main Thread Stacks Total core time 29.9% 23.7% 23.7% 11.6% 8.0% 7.1% 2.9%	293 E 704 Files Main Thread S MPI O 400000000 29.9% 100000 2.8% 0.0 8.0% 8.0% 7.1% 2.9% 2.9% 0.00000 2.9% 0.0000 2.9% 0.0000 2.9% 0.0000 0.000	H.t. += dt; Stacks Functions Dverhead Function(s) on line □	<pre>Source main(int argc, char **argv) { MPI_Alireduce(sflopsAri, sflopsAri_t, 1, NPI_LONG, NPI_SUM, MPI_COMM_WORLD); vtkfile(++nvtk, H, sHv); hydrc_godunov(2, dt, H, sHv, sHw_godunov, sHvw_godunov); hydrc_godunov(2, dt, H, SHv, sHw_godunov, sHvw_godunov); hydrc_arodunov(4, dt, H, SHv, SHw_godunov, SHvw_godunov); hyl_alireduce(sflopsAri, sflopsAri_t, 1, NPI_LONE, NPI_SUM, MPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri_t, 1, NPI_LONE, NPI_SUM, NPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri, 1, NPI_NTI, NPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri, sflopsAri, 1, NPI_NTI, NPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri, sflopsAri, 1, NPI_NTI, NPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri, sflopsAri, 1, NPI_NTI, NPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri, 1, NPI_NTI, NPI_NTI, NPI_COMM_WORLD); MPI_Alireduce(sflopsAri, sflopsAri, 1, NPI_NTI, NP</pre>
Main Thread Stacks Total core time 29.9% sector and the 13.3% 11.6% 8.0% 7.1%	293 Files Main Thread S MPI O 29.9% 10.0779% 2.5% 0 2.5% 0 0 0 0 0 0 0 0 0 0 0 0 0 0	H.t += dt; 5tacks Functions 2verhead Function(s) on line B	<pre>Source main(int argc, char **argv) { MPI_Allreduce(&flopsAri, &flopsAri_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); vtkfile(++nvtk, H, &Hv); hydro_godunov(2, dt, H, &Hv, &Hw_godunov, &Hvw_godunov); hydro_godunov(1, dt, H, &Hv, &Hw_godunov, &Hvw_godunov); MPI_Allreduce(&flopsEgr, &flopsEgr_t, 1, MPI_LONG, MPI_SUM, MPI_COMM_WORLD); MPI_Allreduce(&dt, &dtmin, 1, MPI_COMBLE, MPI_MIN, NPI_COMM_WORLD); </pre>

MPI imbalance: barrier



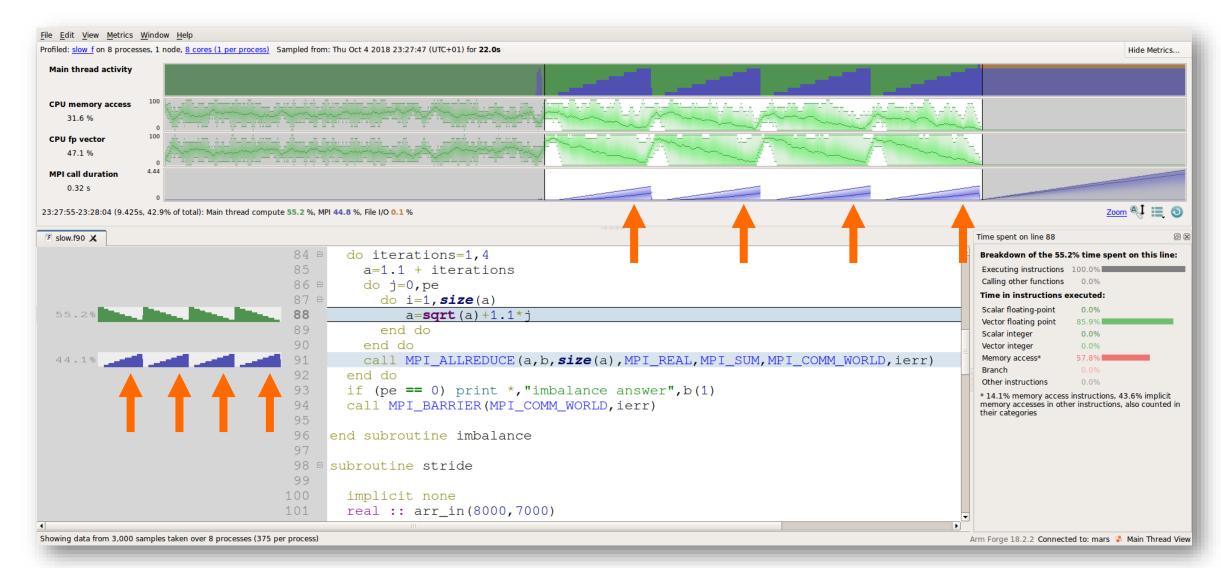
arm

MPI imbalance: barrier



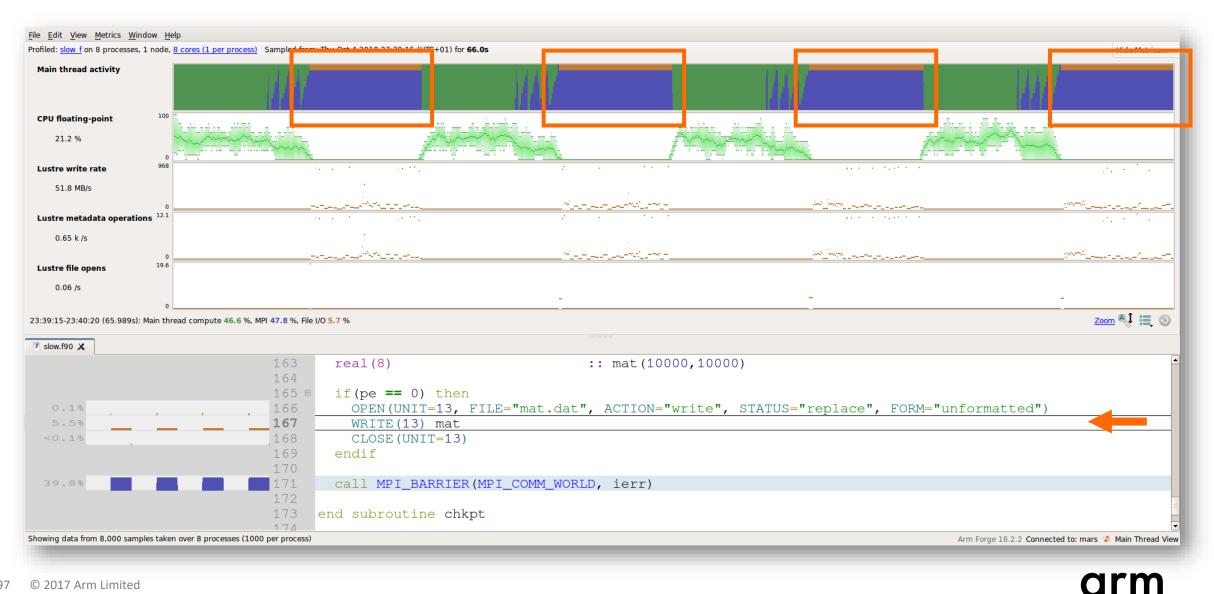
arm

MPI imbalance: all reduce



arm

IO imbalance



Hands – On

- 4_profiling_imbalance
- Compile the code
- Are the MPI communications heavy ? (with Arm Performance Reports)
- Are the IOs efficient ? (with Arm Performance Reports)
- Identify where and how it can be improved (with Arm MAP)
- Modify the code and recompile
- Are the performances better ? (with Arm Performance Reports and Arm MAP)

Contact Support



© 2017 Arm Limited

Issues with Arm Forge ? Our support team is here to help !

For any questions : <u>support-hpc-sw@arm.com</u>

CC : <u>conrad.hillairet@arm.com</u>

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

