Porting HPC Applications to Arm

Phil Ridley

Phil.Ridley@arm.com

5th September 2018

Topics

- Arm in HPC
- Arm Software for HPC
- Tools
- Things to Consider when Porting
- Building GROMACS
- Conclusions

arm


arm
Arm in HPC

Arm Technology Already Connects the World

Arm is ubiquitous

21 billion chips sold by partners in 2017 alone

Mobile/Embedded/IoT/ Automotive/Server/GPUs

Partnership is key

We design IP, not manufacture chips

Partners build products for their target markets

Choice is good

One size is not always the best fit for all

HPC is a great fit for co-design and collaboration

arm

Deployments: Isambard @ GW4

University of BRISTOL

Isambard

GW4

- Cray XC50 series system
  - Aries Interconnect
- 10,000+ Armv8.1a cores
  - Cavium Thunder X2
  - 2 x 32 cores @ > 2.0GHz
- Cray Programming Environment
- Platform for technology comparison
  - x86, GPU, Armv8.1a
- Arm components arriving soon

I.K.Brunel 1804-1859

arm

Deployments: Catalyst UK

- **HPE**, in conjunction with Arm and SUSE, announced in April the "**Catalyst UK**" program: deployments to accelerate the growth of the Arm **HPC** ecosystem into three universities
- Each machine will have:
- 64 HPE Apollo 70 systems, each with two 32-core Cavium ThunderX2 processors (i.e. 4096 cores per system), 128GB of memory and Mellanox InfiniBand interconnects
- SUSE Linux Enterprise Server for HPC

Bristol: VASP, CASTEP, Gromacs, CP2K, Unified Model, NAMD, Oasis, NEMO, OpenIFS, CASINO, LAMMPS

EPCC: WRF, OpenFOAM, Two PhD candidates

Leicester: Data-intensive apps, genomics, MOAB Torque, DiRAC collab

Department for Business, Energy & Industrial Strategy

arm


arm
Arm Software for HPC

Open source and commercial tools

Arm and partners collaborating to increase end-user performance

Open source

Compiler performance of both GCC and LLVM compilers is enhanced by Arm

OpenHPC 1.3.5 release is now out

- Builds are available for both CentOS and SUSE

Community building for HPC apps porting and performance

- Arm HPC GitLab: https://gitlab.com/arm-hpc/

Arm Allinea Studio

- **Comprehensive and integrated tool suite**
- **Commercially supported** by Arm
- **Frequent releases** with continuous performance improvements
- **Ready for current and future generations** of Arm-based HPC platforms

arm

Software Ecosystem – HPC Applications Porting

GROMACS	LAMMPS	CESM	MrBayes	Bowtie
NAMD	AMBER	Paraview	SIESTA	VMD
WRF	Quantum ESPRESSO	CP2k	MILC	GEANT4
OpenFOAM	GAMESS	VisIT	DL_POLY	NEMO
BLAST	NWCHEM	Abinit	BWA	QMCPACK

Build recipes online at https://gitlab.com/arm-hpc/packages/wikis/home

Chem/Phys | Weather | CFD | Visualization | Genomics


arm
Tools for HPC on Arm

arm COMPILER

Commercial C/C++/Fortran compiler with best-in-class performance

Compilers tuned for Scientific Computing and HPC

Latest features and performance optimizations

Commercially supported by Arm

Tuned for Scientific Computing, HPC and Enterprise workloads

- Processor-specific optimizations for various server-class Arm-based platforms
- Optimal shared-memory parallelism using latest Arm-optimized OpenMP runtime

Linux user-space compiler with latest features

- C++ 14 and Fortran 2003 language support
- Some Fortran 2008 language support
- Fortran has OpenMP 3.1 support and some OpenMP 4.0/4.5 support
- C/C++ has OpenMP 4.0/4.5 support (excluding omp declare simd, device constructs and offloading)
- Support for Armv8-A and SVE architecture extension
- Based on LLVM and Flang, leading open-source compiler projects

Commercially supported by Arm

- Available for a wide range of Arm-based platforms running leading Linux distributions – RedHat, SUSE and Ubuntu

* Without offloading

arm COMPILER

Useful flags for armclang, armclang++ and armflang

Most of the flags for the Arm HPC compilers are the same for GCC

Use the -mcpu=native flag. Then try the following options, in order of lowering the optimization

1. -Ofast (this produces the fastest code)
2. -Ofast -fno-stack-arrays (this forces automatic arrays not to be placed on the stack)
3. -O3 -ffp-contract=fast (still allows fused floating-point operations)
4. -O3
5. -O2

Compiler Flag	Description
--help	Display list of supported options, there are further (non-supported) options available with --help-hidden
-mcpu=thunderx2t99 or -mcpu=native	Optimize for particular CPU
-O3	Very high optimization, the default is -O0 which turns off most optimizations
-Ofast	Everything from -O3 but also -ffp-contract=fast and other more aggressive optimizations
-fopenmp	Enable OpenMP directives (not enabled by default)
-g	Generate source-level debug information
-Rpass=\(loop-vectorize|inline\)	Find out what the compiler has optimized
-S	Outputs assembly code, rather than object code. Produces a text .s file containing annotated assembly code
-v	Show commands to run and use verbose output

arm PERFORMANCE LIBRARIES

Optimized BLAS, LAPACK and FFT

Commercially supported by Arm

Best in class performance

Validated with NAG test suite

Commercial 64-bit Armv8-A math libraries

- Commonly used low-level math routines - BLAS, LAPACK and FFT
- Provides FFTW compatible interface for FFT routines

Best-in-class serial and parallel performance

- Generic Armv8-A optimizations by Arm
- Tuning for specific platforms like Cavium ThunderX2

Validated and supported by Arm

- Validated with NAG's test suite, a de facto standard
- Responsive support team

arm PERFORMANCE LIBRARIES

How to link

```
[phirid01@co-c6-16-1 ~]$ echo $ARMPL_LIBRARIES
/opt/arm/armpl-18.4.0_ThunderX2CN99_RHEL-7_arm-hpc-compiler_18.4_aarch64-linux/lib
[phirid01@co-c6-16-1 ~]$ ls $ARMPL_LIBRARIES
libamath.a   libarmpl_ilp64.a      libarmpl_ilp64.so    libarmpl_int64_mp.so  libarmpl_lp64_mp.a   libarmpl_mp.a
libamath.so  libarmpl_ilp64_mp.a   libarmpl_int64.a     libarmpl_int64.so     libarmpl_lp64_mp.so  libarmpl_mp.so
libarmpl.a   libarmpl_ilp64_mp.so  libarmpl_int64_mp.a  libarmpl_lp64.a       libarmpl_lp64.so     libarmpl.so
[phirid01@co-c6-16-1 ~]$
```

Note: To use Arm PL functions in your code, you need to include the header file <armpl.h>
(in $ARMPL_DIR)

To link

gfortran driver.f90 -L${ARMPL_DIR}/lib -larmpl_lp64
armflang driver.f90 -L${ARMPL_DIR}/lib -larmpl_lp64
armclang driver.c -L${ARMPL_DIR}/lib -larmpl_lp64 -lflang -lflangrti
armclang++ driver.cpp -L${ARMPL_DIR}/lib -larmpl_lp64 -lflang -lflangrti

(for multi-threaded versions use -larmpl_lp64_mp)
Documentation is in $ARMPL_DIR/Doc

arm Compiler and Performance Libraries

Current version:18.4

- Key highlights
  - New Fortran Directives – IVDEP and OMP SIMD
    - The Arm Fortran Compiler now supports the general-purpose IVDEP directive, and partially supports the OpenMP-specific OMP SIMD directive
  - Compiler options update - -fstack-arrays now enabled by default at -Ofast optimization level
  - Math routines – New routines (single precision) sinf, cosf, and optimized (double precision) pow, exp and log - as part of the Arm Performance Libraries
  - New Arm Fortran Compiler Reference Guide [PDF]
  - Compiler bug fixes and improvements

arm Compiler and Performance Libraries

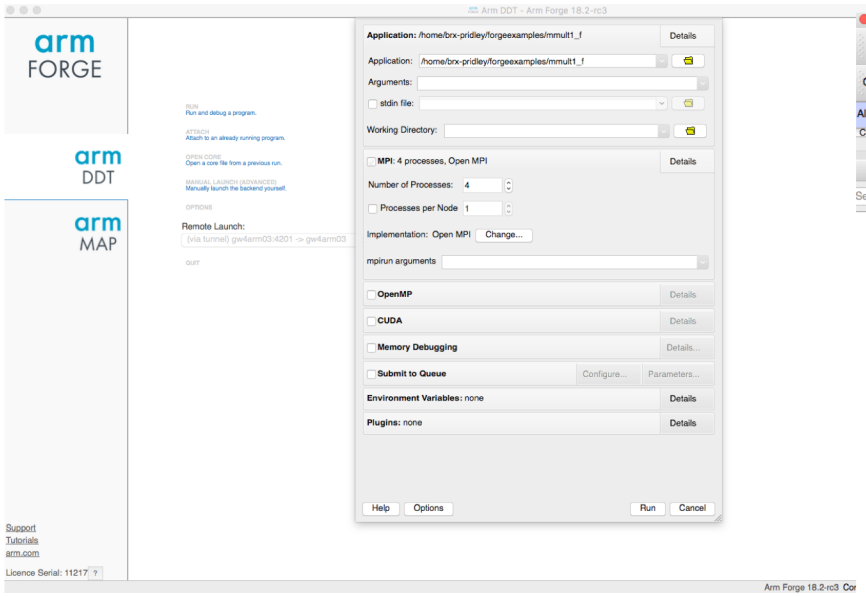
What's coming in version:19.0

- Key highlights
  - Due early Nov
  - Major update for compilers
    - GCC 8.4
    - LLVM 7.0
  - Further performance improvements – better vectorization
  - Fortran 2008 submodules
  - Sparse Matrix Vector Multiplication (SpMV)kernel (needed for HPCG)
  - FFT Guru interface,
  - FFT and BLAS performance improvements
    - CGEMM, SGEMM and ZGEMM
    - Complex-to-real FFTW transforms, especially multidimensional problems

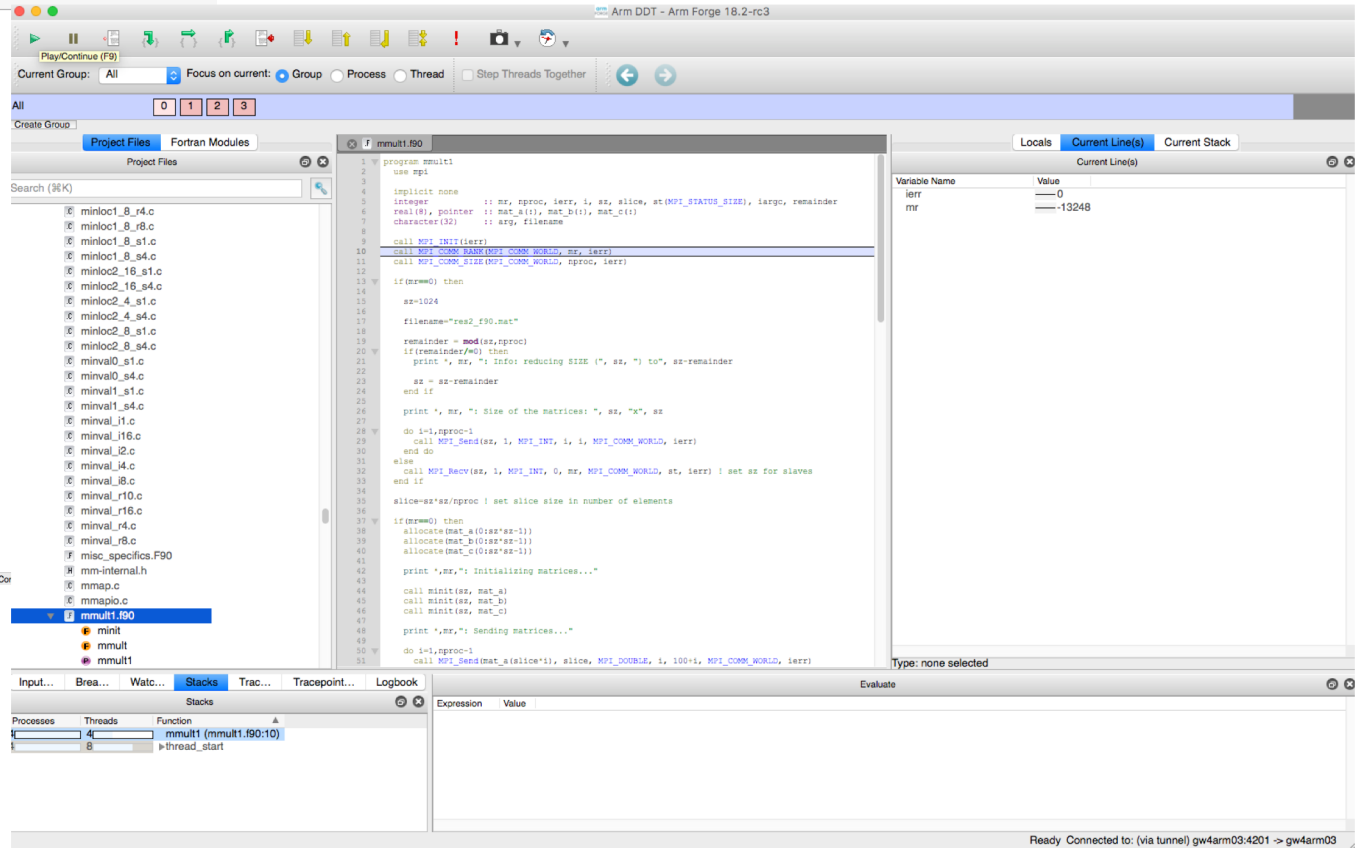
Arm Forge

...debug with DDT

ddt --connect --np 4 ./mmult1_f



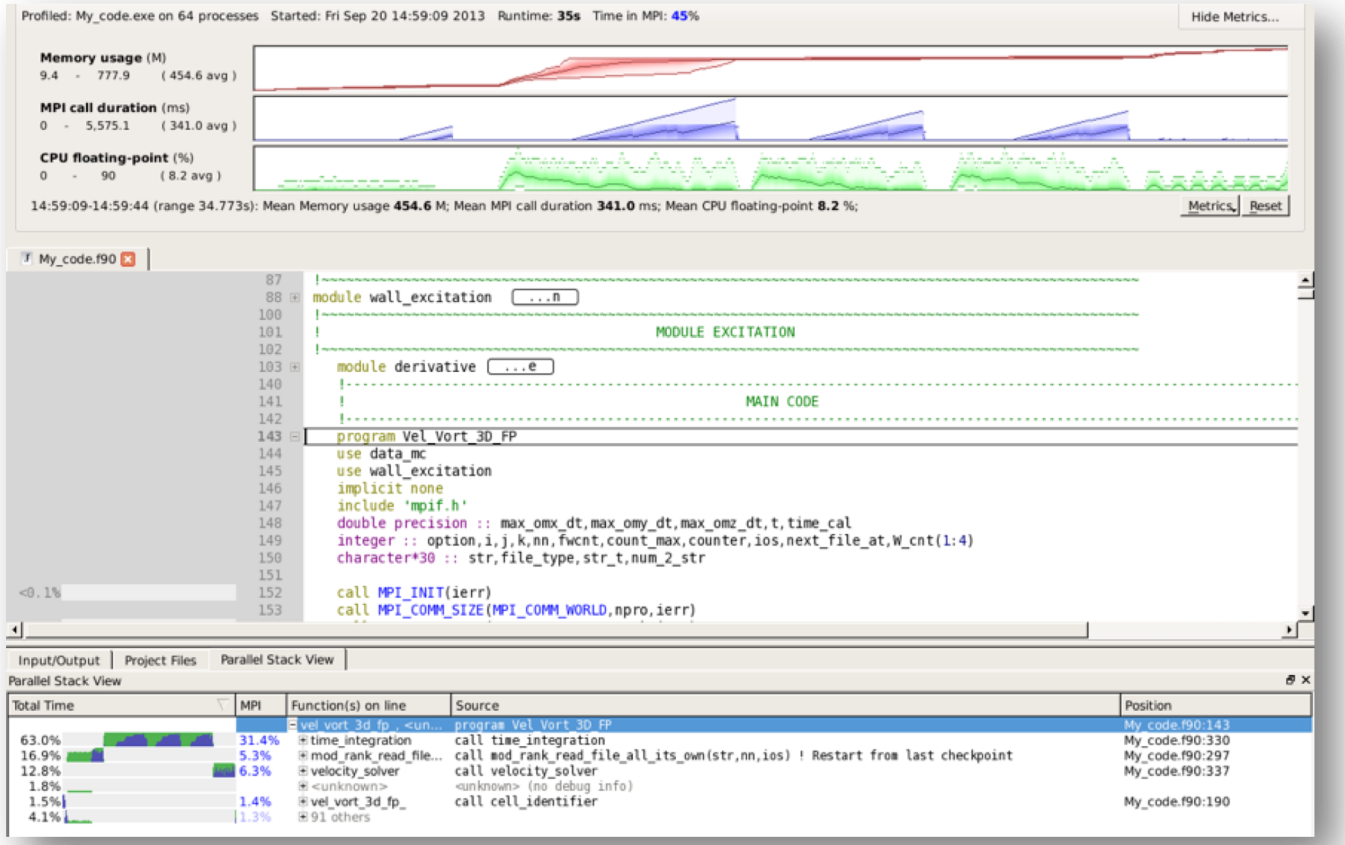
Debug



Arm Forge

...profile with MAP

```
map --profile mpirun –n 48 ./example
```




arm
Things to Consider
When Porting

Surprise!

...I'm relying on a config.guess that's *way* out-of-date!

Often, the config.guess supplied with an application and used by configure will not correctly identify the platform

This can be true for a config.guess already installed on the system and used by some configure scripts

Obtaining up-to-date versions will fix this problem:

```
wget 'http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD' -O config.guess
wget 'http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD' -O config.sub
```

Surprise!

...I'm relying on libtool, but it knows nothing of this "Arm compiler"

configure may not correctly identify the Arm compiler. It may not set the correct flags for libtool to use for position independent code and passing arguments through to the linker. When building libraries, this can cause problems down-the-road

Following **`configure`**, patch libtool as follows:

```
sed -i -e 's#wl=""#wl="-Wl,"#g' libtool
sed -i -e 's#pic_flag=""#pic_flag=" -fPIC -DPIC"#g' libtool
```

Surprise!

...I'm relying on non-standard extensions!

For example ISNAN, COSD ...

Or compiler-specific intrinsics, mm_prefetch, SSE calls etc.

There may be an alternate code path that can be used already. Of possibly the code isn't critical and can be deactivated for now, or an equivalent call can be used, or you could write one?

Surprise!

...OpenMP affinity

cpus might be numbered differently to what you would expect

```
[phirid01@sms09 ~]$ numactl -H
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111
node 0 size: 130235 MB
node 0 free: 111396 MB
node 1 cpus: 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 1
70 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
node 1 size: 130918 MB
node 1 free: 104716 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
[phirid01@sms09 ~]$
```

Bear this in mind when assigning threads to physical cores

Surprise!

...I can use KMP_AFFINITY with my OpenMP code

```
phirid01@avantek-1-1:~/OMP_TUTORIAL$ export KMP_AFFINITY=verbose,compact
phirid01@avantek-1-1:~/OMP_TUTORIAL$ ./omp_example_k_armflang.exe 52 52 52 52
   nx= 52    ny= 52    nz= 52  nvar=  9
 Footprint of grid =         11071  kbytes
 Footprint of one var of grid=          1230  kbytes
OMP: Info #208: KMP_AFFINITY: parsing /proc/cpuinfo.
OMP: Info #148: KMP_AFFINITY: Affinity capable, using cpuinfo file
OMP: Info #154: KMP_AFFINITY: Initial OS proc set respected: {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,4
4,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95}
OMP: Info #156: KMP_AFFINITY: 96 available OS procs
OMP: Info #157: KMP_AFFINITY: Uniform topology
OMP: Info #179: KMP_AFFINITY: 2 packages x 48 cores/pkg x 1 threads/core (96 total cores)
OMP: Info #206: KMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: KMP_AFFINITY: OS proc 0 maps to package 0 core 0
OMP: Info #171: KMP_AFFINITY: OS proc 1 maps to package 0 core 1
OMP: Info #171: KMP_AFFINITY: OS proc 2 maps to package 0 core 2
OMP: Info #171: KMP_AFFINITY: OS proc 3 maps to package 0 core 3
OMP: Info #171: KMP_AFFINITY: OS proc 4 maps to package 0 core 4
OMP: Info #171: KMP_AFFINITY: OS proc 5 maps to package 0 core 5
OMP: Info #171: KMP_AFFINITY: OS proc 6 maps to package 0 core 6
OMP: Info #171: KMP_AFFINITY: OS proc 7 maps to package 0 core 7
OMP: Info #171: KMP_AFFINITY: OS proc 8 maps to package 0 core 8
OMP: Info #171: KMP_AFFINITY: OS proc 9 maps to package 0 core 9
OMP: Info #171: KMP_AFFINITY: OS proc 10 maps to package 0 core 10
OMP: Info #171: KMP_AFFINITY: OS proc 11 maps to package 0 core 11
OMP: Info #171: KMP_AFFINITY: OS proc 12 maps to package 0 core 12
OMP: Info #171: KMP_AFFINITY: OS proc 13 maps to package 0 core 13
OMP: Info #171: KMP_AFFINITY: OS proc 14 maps to package 0 core 14
OMP: Info #171: KMP_AFFINITY: OS proc 15 maps to package 0 core 15
OMP: Info #171: KMP_AFFINITY: OS proc 16 maps to package 0 core 16
```

Surprise!

...Integer divide by zero is zero

```
#include<stdio.h>

int main (int argc, char** argv)
{

  int x  = 0;

  printf("%d\n", 1/x);

return 0;
}
```

```
[phirid01@sms09 ~]$ gcc -o int int.c -O0
[phirid01@sms09 ~]$ ./int
0
```

AArch64

```
pridley@mars:~$ gcc -o int int.c -O0
pridley@mars:~$ ./int
Floating point exception
pridley@mars:~$
```

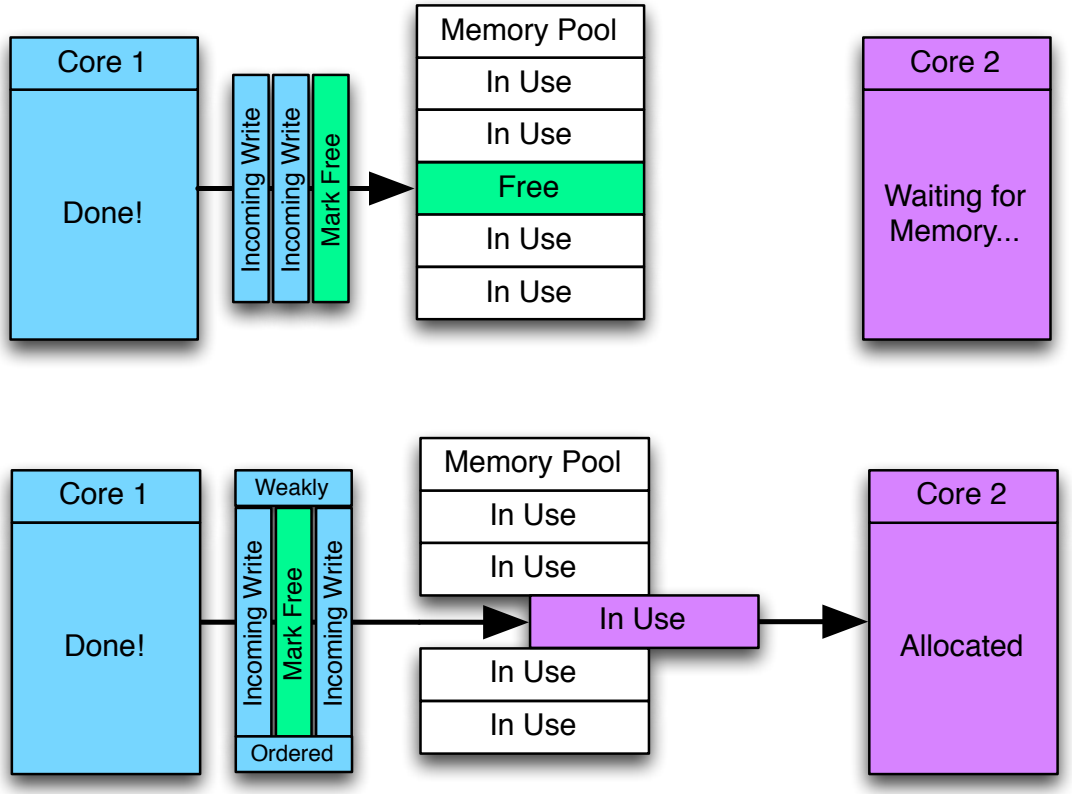
x86

Important note: FP division behaves the same, this is only for integer division

Surprise!

...AArch64 uses a weak memory model

- For nearly all HPC codes this will not be relevant
- Only applies to codes that use their own implementation of shared memory parallelization
- Symptom will be a weird race condition
  - Usually caused by a lock-free thread interaction
  - The implementation relies upon a TSO (stronger) memory model
  - Will behave differently on a weakly ordered memory system



arm


arm
Building GROMACS

GROMACS

GROningen MAchine for Chemical Simulations

- Versatile open-source code that can be used to perform molecular dynamics simulations
- Used on large HPC systems worldwide e.g. ARCHER, NERSC (US), CSC (Finland) and Piz Daint (Switzerland)
- Supported by many developers and contributors
- C++ with OpenMP and MPI
- SIMD intrinsics for a range of instruction sets, including Arm (NEON)
- Built in capability to control threads regarding hardware locality

GROMACS

Building

- Check if there's info on the Arm website

https://gitlab.com/arm-hpc/packages/wikis/packages/gromacs

- GROMACS uses CMake, so check CMakeLists.txt: Are CMAKE_C_FLAGS_RELEASE / CMAKE_CXX_FLAGS_RELEASE set with best choice for optimizations?

```
set(CMAKE_C_FLAGS_RELEASE "-Ofast -DNDEBUG")
set(CMAKE_CXX_FLAGS_RELEASE "-Ofast -DNDEBUG")
```

- Try (building own FFTW)

```
cmake -DCMAKE_INSTALL_PREFIX=${gromacs_install}  -DBUILD_SHARED_LIBS=off  -DCMAKE_C_COMPILER=`which mpicc`
-DCMAKE_CXX_COMPILER=`which mpicxx` -DGMX_BUILD_OWN_FFTW=on  -DGMX_SIMD=ARM_NEON_ASIMD
-DGMX_DOUBLE=off -DGMX_EXTERNAL_BLAS=on -DGMX_EXTERNAL_LAPACK=on -DGMX_FFT_LIBRARY=fftw3
-DGMX_BLAS_USER=${ARMPL_DIR}/lib/libarmpl_lp64.so -DGMX_LAPACK_USER=${ARMPL_DIR}/lib/libarmpl_lp64.so
-DGMX_GPU=off -DGMX_MPI=on -DGMX_OPENMP=on  -DGMX_X11=off ..
```

(may also need -DGMX_HWLOC=off)

arm

GROMACS

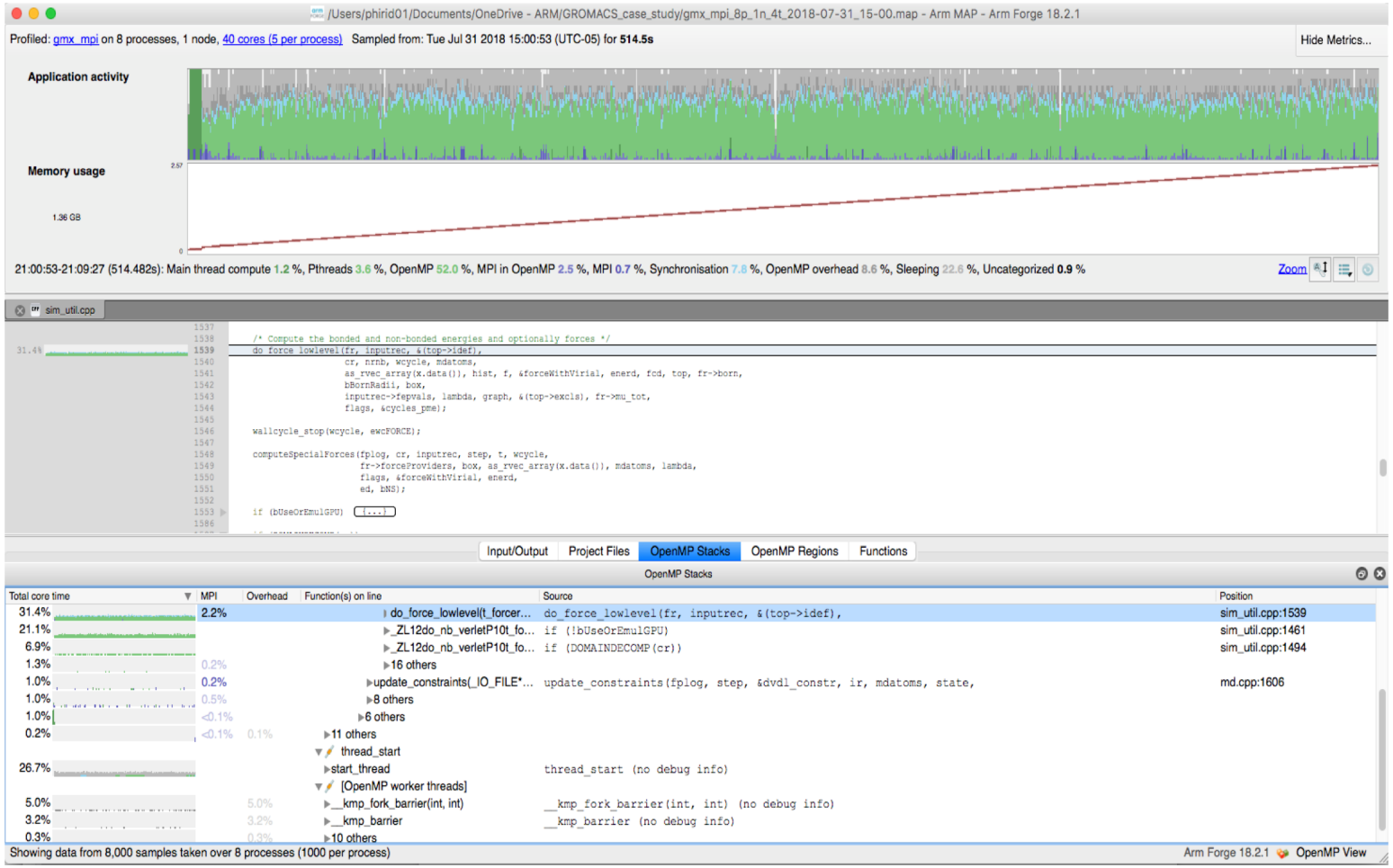
Performance

- Build separate versions for both GCC and armclang++
  - Double check which flags are actually being used
- Look at performance for both versions
- Check thread to core affinity and task placement
  - On a TX2 you typically have either 28 or 32 physical cores per socket, on a dual socket node
  - Each physical core can be configured with SMT=4, thus giving 112 or 128 logical cores per socket
  - Several different ways of achieving this, e.g. OpenMPI --report-bindings
  - Simpler when SMT=1
- Find out optimal number of OpenMP threads to use, e.g. export OMP_NUM_THREADS=4

GROMACS

Investigate Performance

- Profile with Arm MAP
  - Use -g compiler flag so that MAP can resolve required symbols and debug info
  - May need to use compatibility launch
  - Determine where code is spending the most time

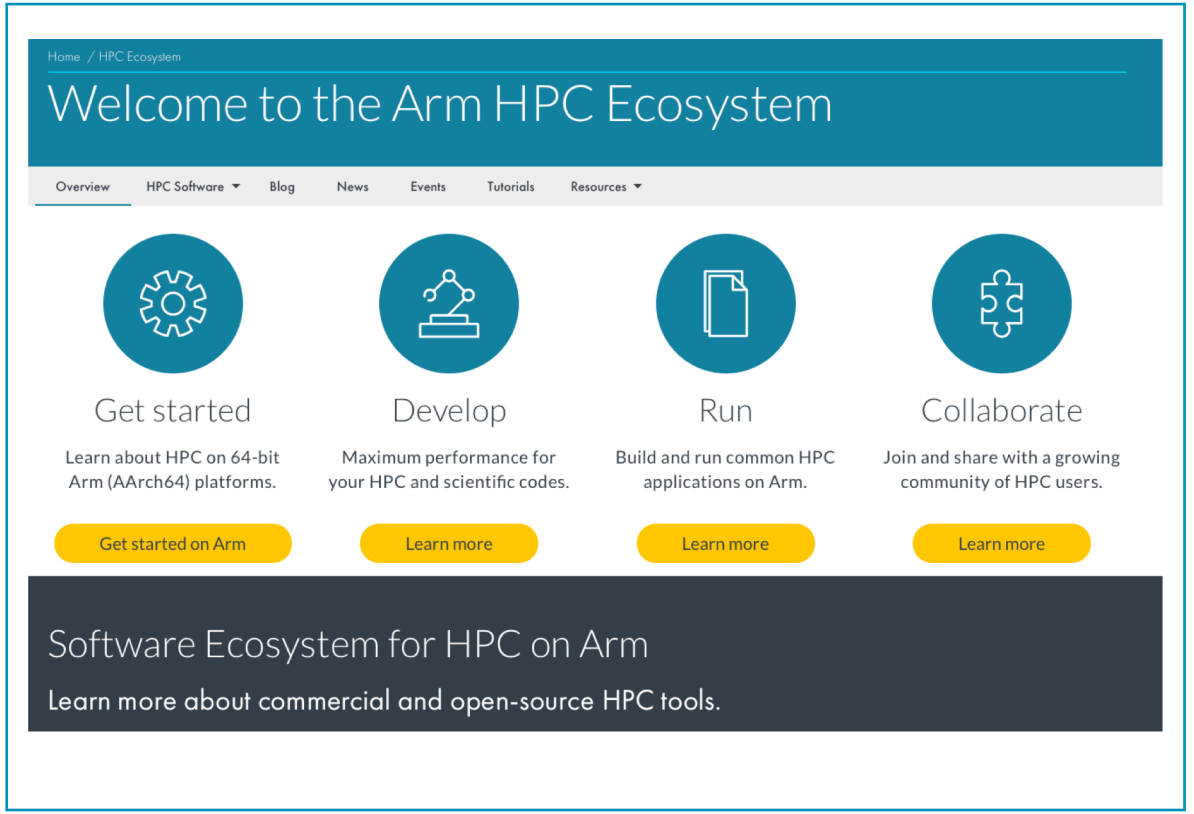


Conclusions

Arm HPC Ecosystem website: https://developer.arm.com/hpc

Clearinghouse for Arm's HPC ecosystem, information channels, and collaboration

- Latest events, news, blogs, and collateral including whitepapers, webinars, and presentations
- Links to HPC open-source & commercial SW packages
- Recipes for porting HPC apps
- New Arm HPC User Group Forum
- Curated and moderated by Arm



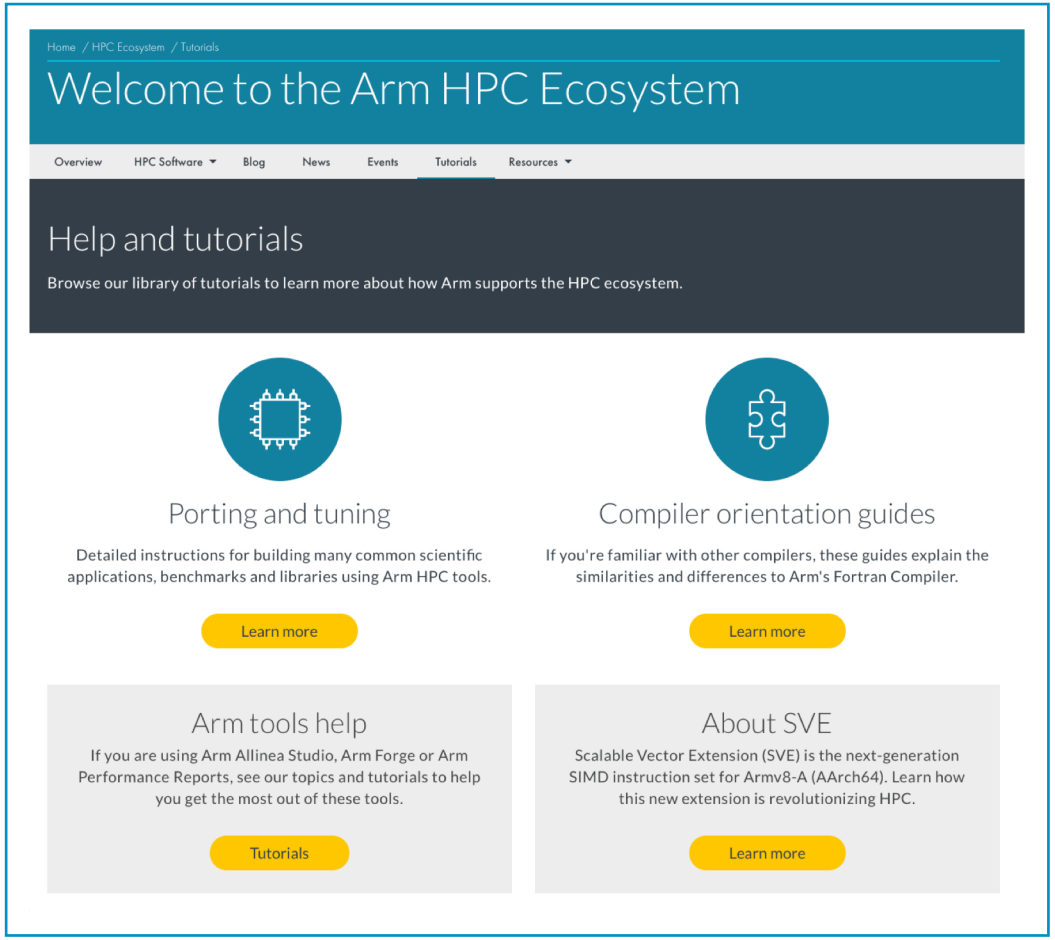
Participate, share progress, and expertise

arm

Porting to Arm website: https://developer.arm.com/hpc/tutorials

Useful for reference when porting your application

- Tips on how to port to Arm
- Tips on using the compilers and performance libraries
- Tips on using Arm DDT and Arm MAP
- How to build some widely-used open-source packages
- Questions, comments, ideas or problems? Please get in touch with the Arm support team



https://developer.arm.com/products/software-development-tools/hpc/get-support

arm

Supporting our users – You're not on your own!

Arm Professional Services: Increasing scientific code performance

- In addition to developing software we are here to help users
- Work is now extended to helping users port and optimize their codes on Arm HPC systems
- We are already working with users to get best performance out of Arm deployments

arm