

An aerial photograph of a city, likely Oxford, featuring a prominent Gothic cathedral (Christ Church) and various university buildings. A grid of small white crosses is overlaid on the image. Three crosses are highlighted: a green one near the top center, a yellow one on the left side, and an orange one near the bottom center.

arm

# Arm in HPC

Contact: [john.linford@arm.com](mailto:john.linford@arm.com)

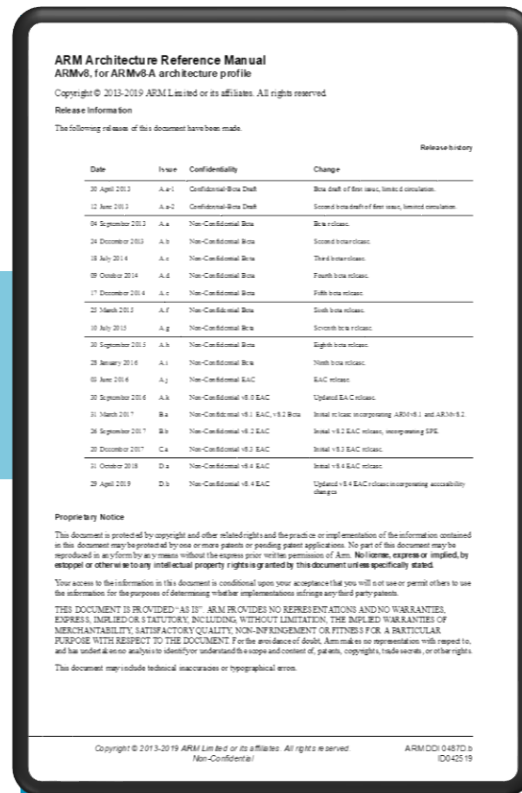
# CPU Engagement Models with Arm

Arm IP is the basic building block for extraordinary solutions.

## Core License

- Partner licenses complete microarchitecture.
- CPU differentiation via:
- Configuration options.
- Wide implementation envelope with different process technologies.

## Arm IP



ARM Architecture Reference Manual  
ARMv8, for ARMv8-A architecture profile  
Copyright © 2013-2019 ARM Limited or its affiliates. All rights reserved.

Release information

The following releases of this document have been made:

Date	Issue	Confidentiality	Change	Release history
20 April 2013	A-1	Confidential/Non-Conf	First draft of first issue, based on consultation.	
12 June 2013	A-2	Confidential/Non-Conf	Second issue draft of first issue, limited permission.	
04 September 2013	A-4	Non-Confidential/Non-Conf	First release.	
24 December 2013	A-9	Non-Confidential/Non-Conf	Second issue release.	
18 July 2014	A-1	Non-Confidential/Non-Conf	Third issue release.	
09 October 2014	A-6	Non-Confidential/Non-Conf	Fourth issue release.	
17 December 2014	A-1	Non-Confidential/Non-Conf	Fifth issue release.	
22 March 2015	A-7	Non-Confidential/Non-Conf	Sixth issue release.	
18 July 2015	A-8	Non-Confidential/Non-Conf	Seventh issue release.	
20 September 2015	A-9	Non-Confidential/Non-Conf	Eighth issue release.	
28 January 2016	A-1	Non-Confidential/Non-Conf	Ninth issue release.	
05 June 2016	A-2	Non-Confidential/Non-Conf	EAC release.	
30 September 2016	A-3	Non-Confidential/Non-Conf	Updated EAC release.	
31 March 2017	B-1	Non-Confidential/Non-Conf	Initial release incorporating ARMv8.1 and ARMv8.1-1.	
24 September 2017	B-2	Non-Confidential/Non-Conf	Initial v1.2 EAC release, incorporating EPC.	
20 December 2017	C-1	Non-Confidential/Non-Conf	Initial v1.3 EAC release.	
21 October 2018	D-1	Non-Confidential/Non-Conf	Initial v1.4 EAC release.	
28 April 2019	D-2	Non-Confidential/Non-Conf	Updated v1.4 EAC release incorporating accessibility changes.	

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise in any intellectual property rights is granted by this document or is specifically excluded.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementation infringes any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

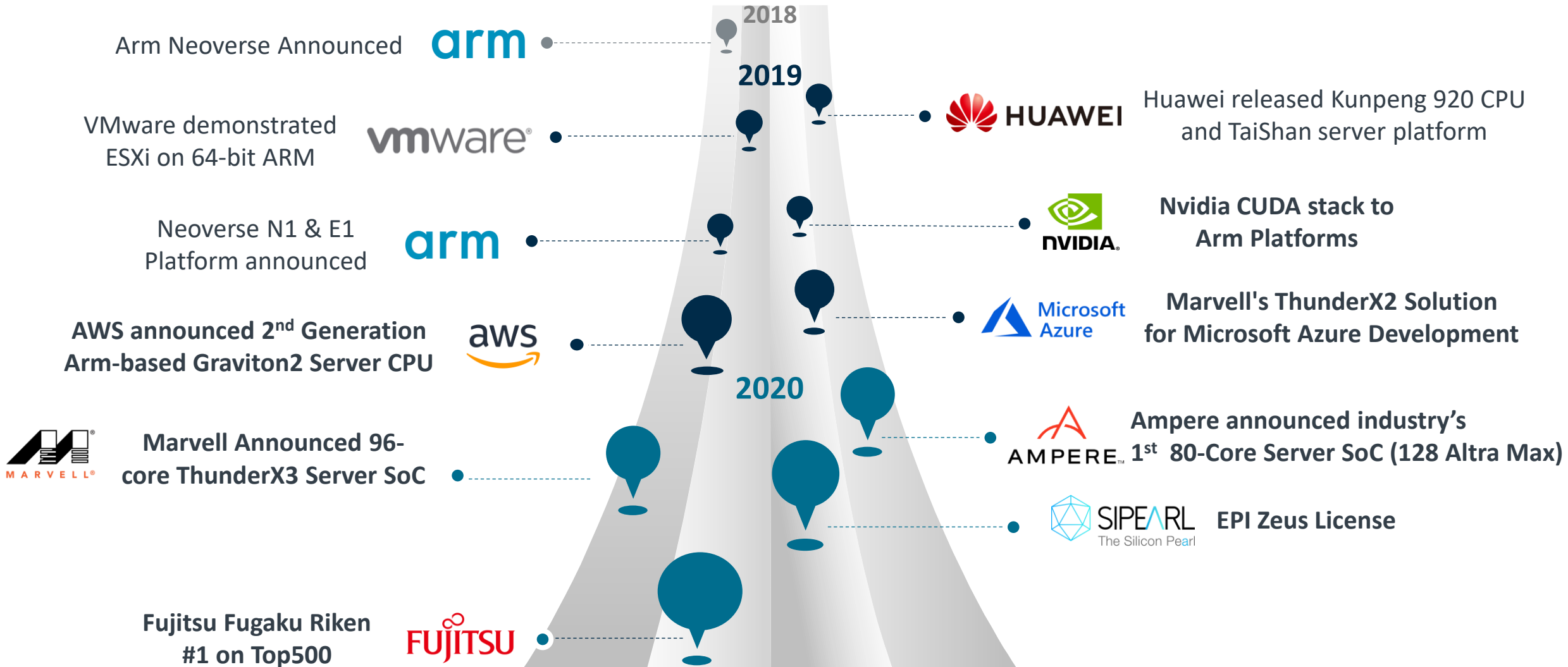
This document may include technical inaccuracies or typographical errors.

Copyright © 2013-2019 ARM Limited or its affiliates. All rights reserved. ARM/DO-0487D-2  
Non-Confidential Non-Confidential

## Architecture License

- Partner designs complete microarchitecture.
- Clean room, scratch.
- Maximum design freedom:
- Directly address needs of the target market.
- Arm architecture validation preserves software compatibility

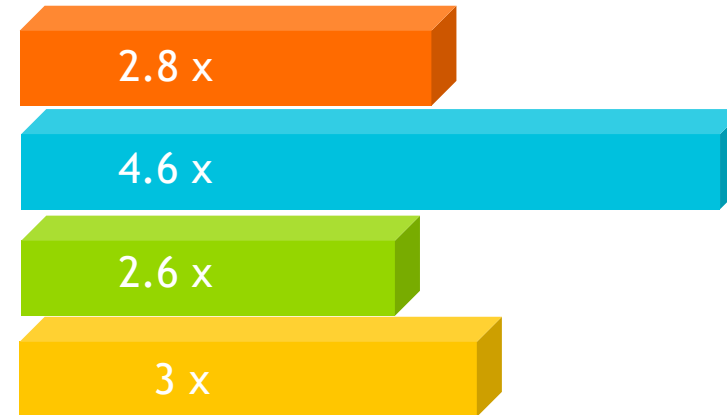
# Arm Neoverse Momentum in Servers & HPC



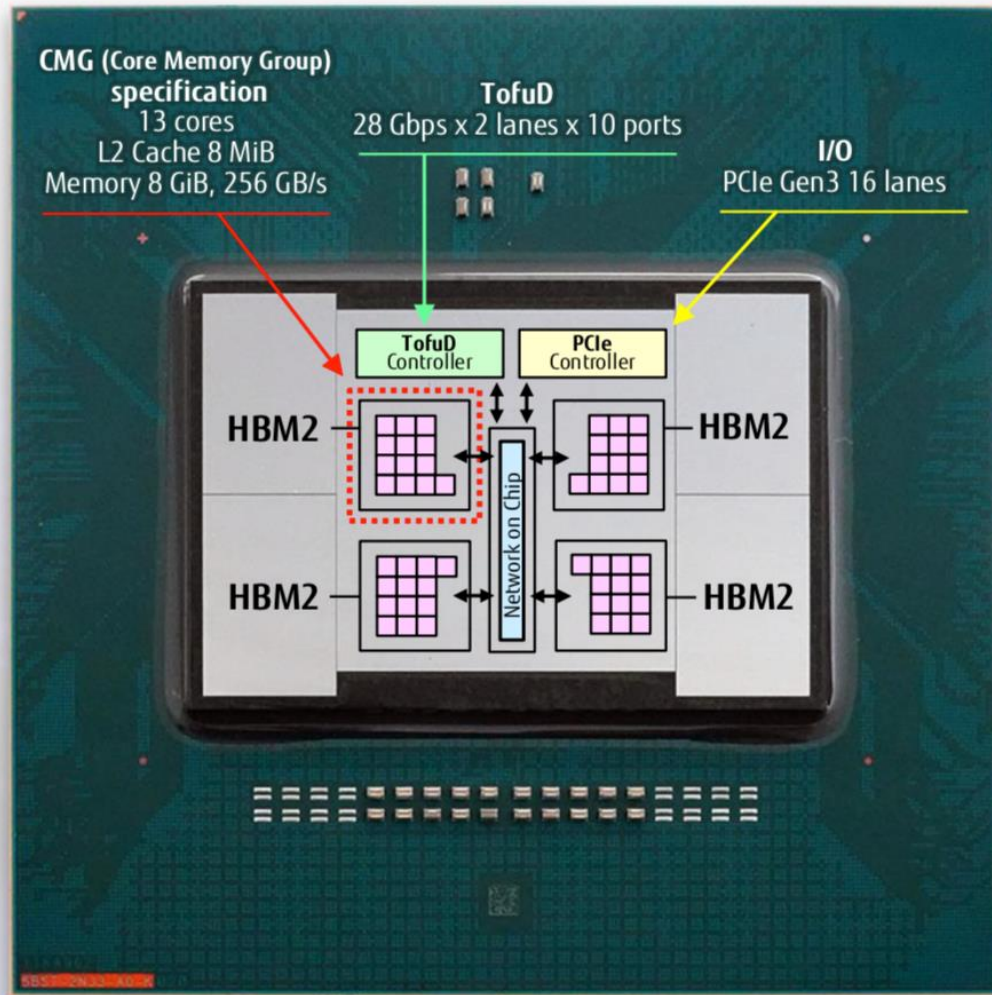
# Fujitsu/RIKEN Fugaku: Fastest Supercomputer in the World

Top place in 4 categories:

Top500 @ 416 Pflop/s  
HPCG @ 13.4 Pflop/s  
HPL-AI @ 1.42 Eflop/s  
Graph 500 @ 70980 GTEPS



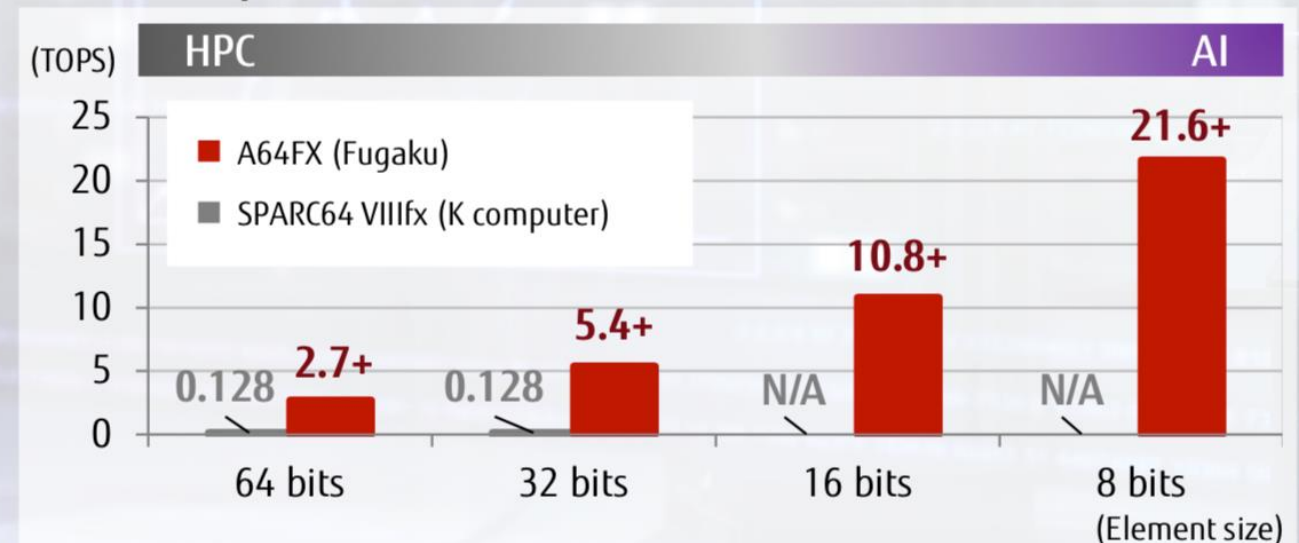
# 1. High-Performance Arm CPU A64FX in HPC and AI Areas



## Architecture features

ISA	Armv8.2-A (AArch64 only) SVE (Scalable Vector Extension)	
SIMD width	512-bit	
Precision	FP64/32/16, INT64/32/16/8	
Cores	48 computing cores + 4 assistant cores (4 CMGs)	
Memory	HBM2: Peak B/W 1,024 GB/s	
Interconnect	TofuD: 28 Gbps x 2 lanes x 10 ports	

## Peak performance (Chip level)



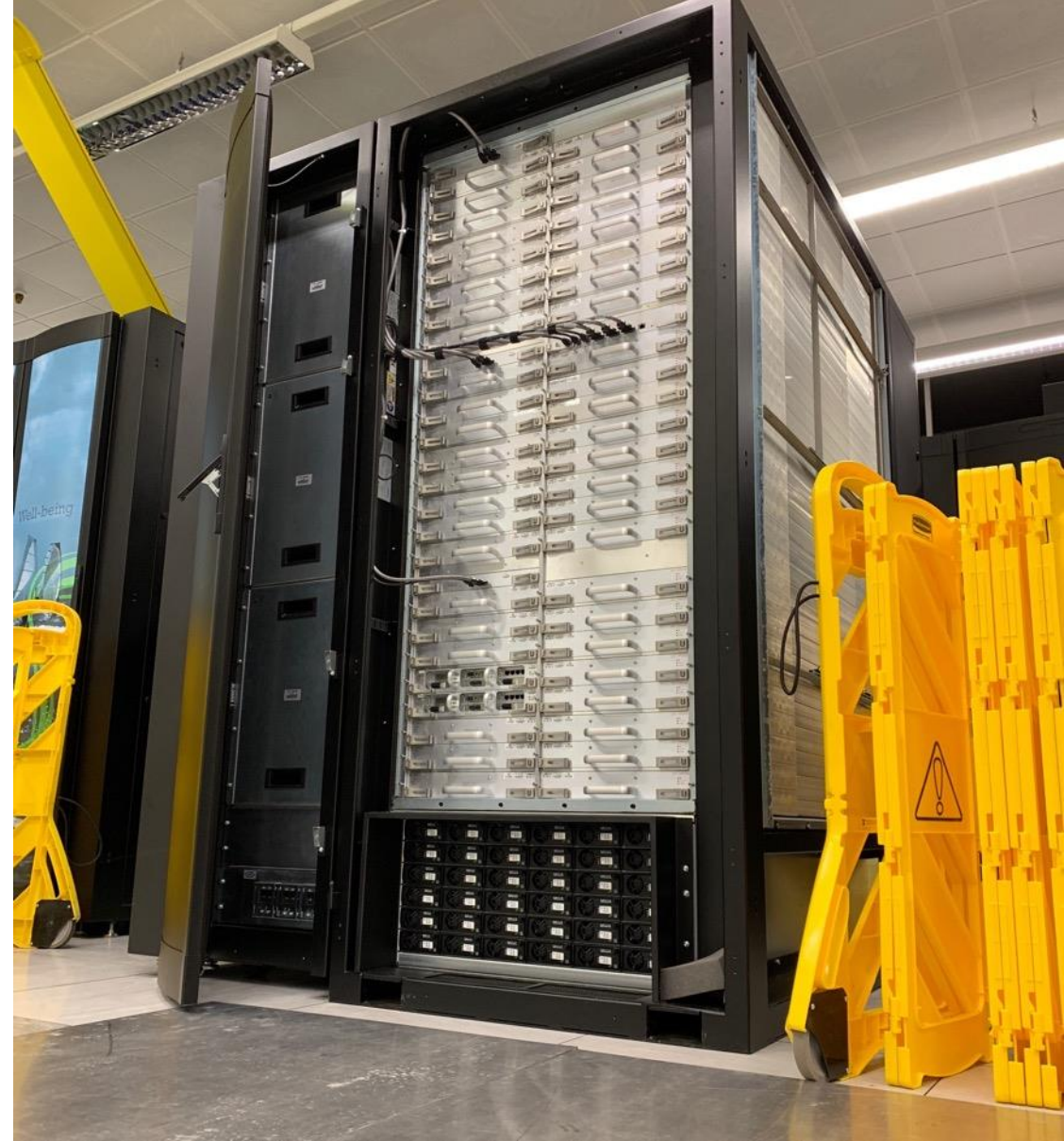
# Vanguard Astra by HPE

- 2,592 HPE Apollo 70 compute nodes
  - 5,184 CPUs, 145,152 cores, 2.3 PFLOPs (peak)
- **Marvell ThunderX2** ARM SoC, 28 core, 2.0 GHz
- Memory per node: 128 GB (16 x 8 GB DR DIMMs)
  - Aggregate capacity: 332 TB, 885 TB/s (peak)
- Mellanox IB EDR, ConnectX-5
  - 112 36-port edges, 3 648-port spine switches
- Red Hat RHEL for Arm
- HPE Apollo 4520 All-flash Lustre storage
  - Storage Capacity: 403 TB (usable)
  - Storage Bandwidth: 244 GB/s



# Isambard system specification

- **10,752** Armv8 cores (168n x 2s x 32c)
  - Cavium ThunderX2 32core 2.1→2.5GHz
- Cray XC50 'Scout' form factor
- High-speed **Aries** interconnect
- Cray HPC optimised software stack
  - CCE, Cray MPI, math libraries, CrayPAT, ...
- **Phase 2 (the Arm part):**
  - Delivered Oct 22<sup>nd</sup>, handed over Oct 29<sup>th</sup>
  - Accepted Nov 9<sup>th</sup>
  - Upgrade to final B2 TX2 silicon, firmware, CPE completed March 15<sup>th</sup> 2019



# Isambard 2 production system

- **21,504** Armv8 cores (168n x 2s x 32c)
  - Marvell ThunderX2 32 core @2.5GHz
- Cray XC50 'Scout' form factor
- High-speed **Aries** interconnect
- Cray HPC optimised software stack
  - Compilers, math libraries, CrayPAT, ...
  - Also comes with all the open source software toolchains: GNU, Clang/LLVM etc.





# Isambard 2's A64fx Apollo80 system

- 72 nodes, 3,456 cores, 1.8GHz
  - 72 TB/s memory bandwidth
  - 202 TFLOP/s double precision
- Connected with 100Gbps InfiniBand
- Comes with a Cray software stack
  - CCE, Armclang, GNU
- Hope to add the Fujitsu compiler



# CEA : Deployment by ATOS



- 292 **Atos Sequana X1310 compute nodes**
- 584 CPUs, 18,688 cores
- Marvell ThunderX2 **ARM SoC, 32 cores, 2.2 GHz**
- Memory : 8 channels, DDR4 2666, 256 GB
- Mellanox InfiniBand EDR

- ✓ **Peak Performance 329 TFLOPS**
- ✓ **HPL = 84% of efficiency**
- ✓ **HPCG = 3.47 of HPL**



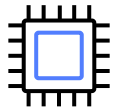
# AWS Graviton2 - an Arm Server Processor



## Graviton Processor



First Arm-based processor available in major cloud



Built on 64-bit Arm Neoverse cores with AWS-designed silicon using 16nm manufacturing technology



Up to 16 vCPUs, 10Gbps enhanced networking, 3.5Gbps EBS bandwidth

## Graviton2 Processor



7x performance, 4x compute cores, and 5x faster memory



Built with 64-bit Arm Neoverse cores with AWS-designed silicon using 7nm manufacturing technology

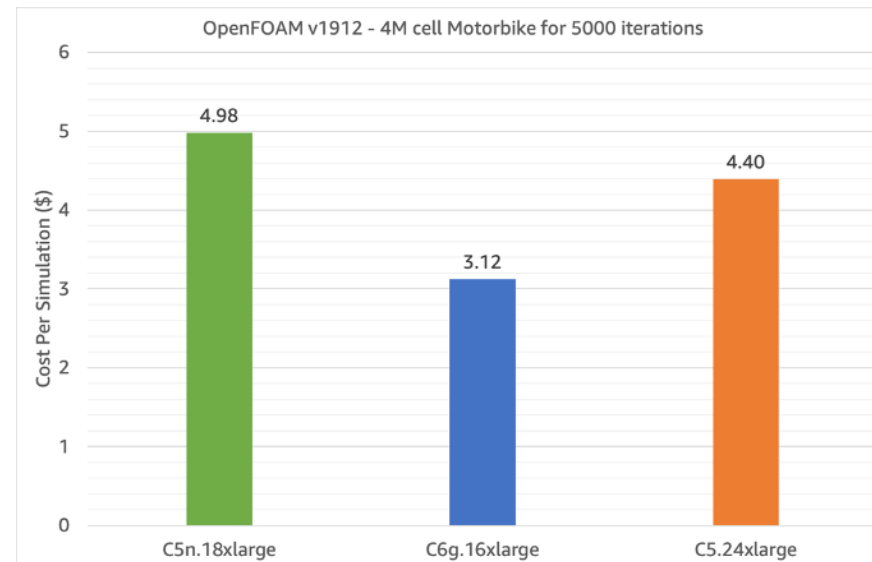
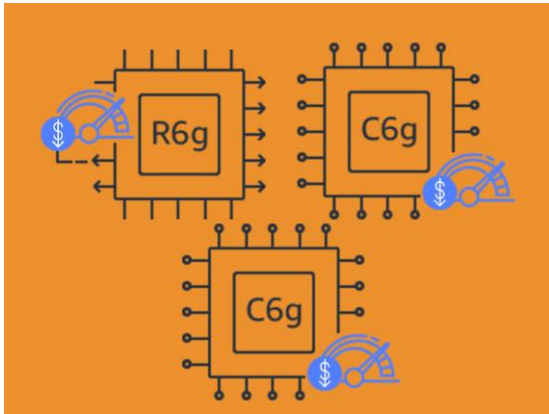


Up to 64 vCPUs, 25Gbps enhanced networking, 18Gbps EBS bandwidth

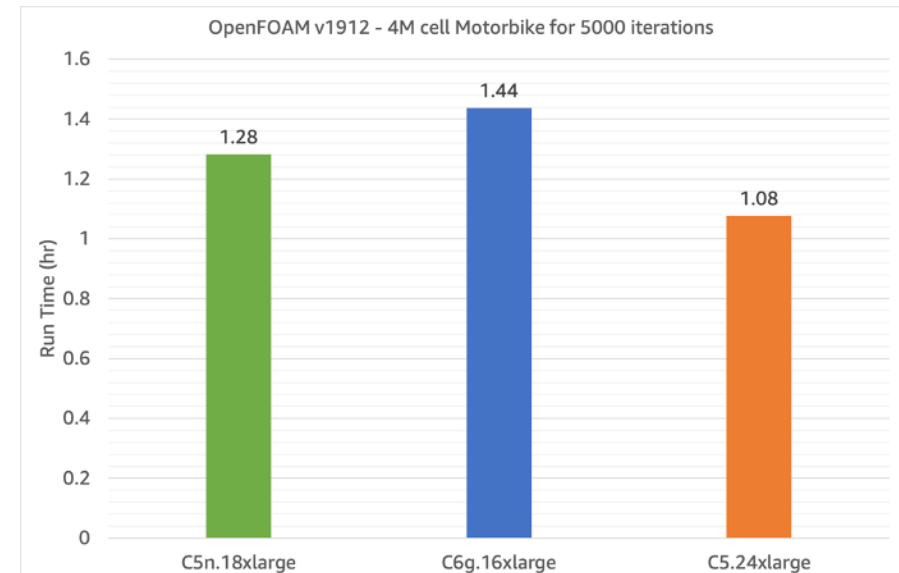
# AWS Graviton 2 for HPC workloads

The c6g instances have outstanding price/performance as compared to similar x86 instances

- The AWS Graviton 2 implements the Arm Neoverse N1
- Up to 40% improved price/performance over x86 instances



Cost: lower is better



Run time: lower is better



arm

Software Ecosystem



## Applications

Open-source, owned, commercial ISV codes, ...

## Debuggers & Profilers

Arm Forge (DDT, MAP),  
Rogue Wave, HPC Toolkit,  
Scalasca, Vampir, TAU, ...

## Containers, Interpreters, etc.

Singularity, PodMan, Docker, Python, ...

## Middleware

Mellanox IB/OFED/HPC-X, OpenMPI, MPICH, MVAPICH2, OpenSHMEM, OpenUCX, HPE MPI

## OEM/ODM's

Cray-HPE, ATOS-Bull,  
Fujitsu, Gigabyte, ...

## Compilers

Arm, GNU, LLVM, Clang, Flang,  
Cray, PGI/NVIDIA, Fujitsu, ...

## Libraries

ArmPL, FFTW, OpenBLAS,  
NumPy, SciPy, Trilinos, PETSc,  
Hypre, SuperLU, ScaLAPACK, ...

## Filesystems

BeeGFS, Lustre, ZFS,  
HDF5, NetCDF, ...

## Silicon Suppliers

Marvell, Fujitsu,  
Mellanox, NVIDIA, ...

## OS

RHEL, SUSE, CentOS, Ubuntu, ...

## Arm Server Ready Platform

Standard firmware and RAS

## Schedulers

SLURM, IBM LSF, Altair PBS Pro, ...

## Cluster Management

Bright, HPE CMU, xCat, Warewulf, ...

# arm

# A Rich and Growing Application Ecosystem

GROMACS	LAMMPS	CESM2	MrBayes	Bowtie	DeepBench
NAMD	TensorFlow	ParaView	SIESTA	UM	AMBER
WRF	Quantum ESPRESSO	VASP	Torch	MILC	GEANT4
OpenFOAM	GAMESS	Mahout	VisIt	DL-Poly	NEMO
Weka	BLAST	NWCHEM	Abinit	BWA	QMCPACK
<i>Chem/Phys</i>	<i>Weather</i>	<i>CFD</i>	<i>Visualization</i>	<i>Genomics</i>	<i>AI/ML</i>



# GNU and LLVM Toolchains

Toolchains for all Arm cores – supported at release

## Status:

- LTS Linux distributions support Arm CPU features when a CPU becomes generally available
- Improve performance for key user workloads and industry benchmarks

## GNU Toolchain (compilers, debuggers, libraries, etc.)

- Default compiler in Linux distributions like RedHat, SUSE, Ubuntu
- Key segments: Cloud, networking and HPC

## LLVM Toolchain (compilers, debuggers, libraries, etc.)

- Default compiler in Android and the basis for commercial compilers (including Arm and Cray compilers)
- Key segments: Mobile (Android/iOS), Cloud



# Example: SVE Support

Over four years of active, ongoing development

- **Arm actively posting SVE open source patches upstream**
  - Beginning with first public announcement of SVE at HotChips 2016
- **Available upstream**
  - [Since GNU Binutils-2.28](#) Released Feb 2017, includes SVE assembler & disassembler
  - Since GCC 8: Full assembly, disassembly and basic auto-vectorization
  - Since LLVM 7: Full assembly, disassembly
  - Since QEMU 3: User space SVE emulation
  - Since GDB 8.2: HPC use cases fully included
- **Constant upstream review**
  - [LLVM](#): Since Nov 2016, as presented at LLVM conference
  - [Linux kernel](#): Since Mar 2017, LWN article on SVE support



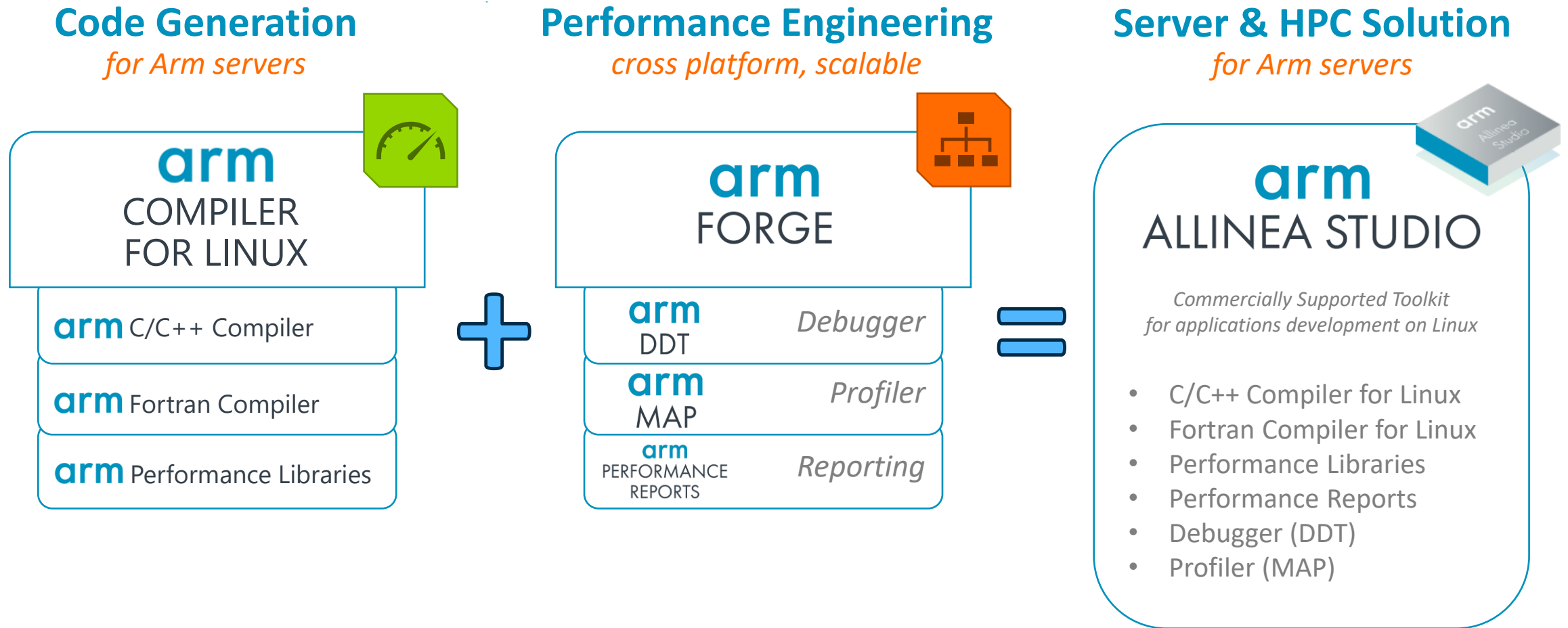
**Automatic Arm support in latest version of all tools – peer to x86**

# Example: Auto-vectorization in LLVM

- Auto-vectorization via LLVM **vectorizers**:
  - Use cost models to drive decisions about what code blocks can and/or should be vectorized.
  - Since **October 2018**, two different vectorizers used from LLVM: [Loop Vectorizer](#) and [SLP Vectorizer](#).
- Loop Vectorizer support for SVE and NEON:
  - Loops with unknown trip count
  - Runtime checks of pointers
  - Reductions
  - Inductions
  - “If” conversion
  - Pointer induction variables
  - Reverse iterators
  - Scatter / gather
  - Vectorization of mixed types
  - Global structures alias analysis

# Server & HPC Development Solutions from Arm

Commercially supported tools for Linux and high performance computing



# Arm Compiler for Linux

a.k.a Arm Compiler for HPC, a.k.a. Arm Alinea Compiler



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

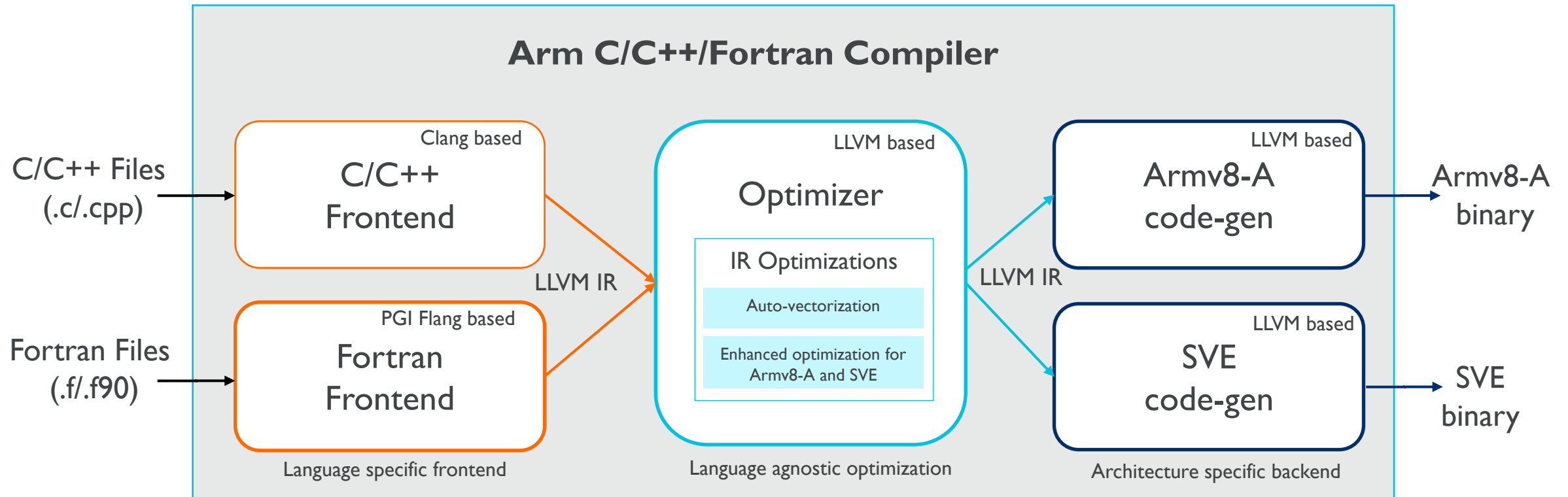
## Linux user-space compiler with latest features

- C++ 14 and Fortran 2003 language support with OpenMP 4.5
- 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

# Building on LLVM, Clang and Flang projects



# 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
- Sparse linear algebra and batched BLAS support
- libamath gives high-performing math.h functions implementations

## Best-in-class serial and parallel performance

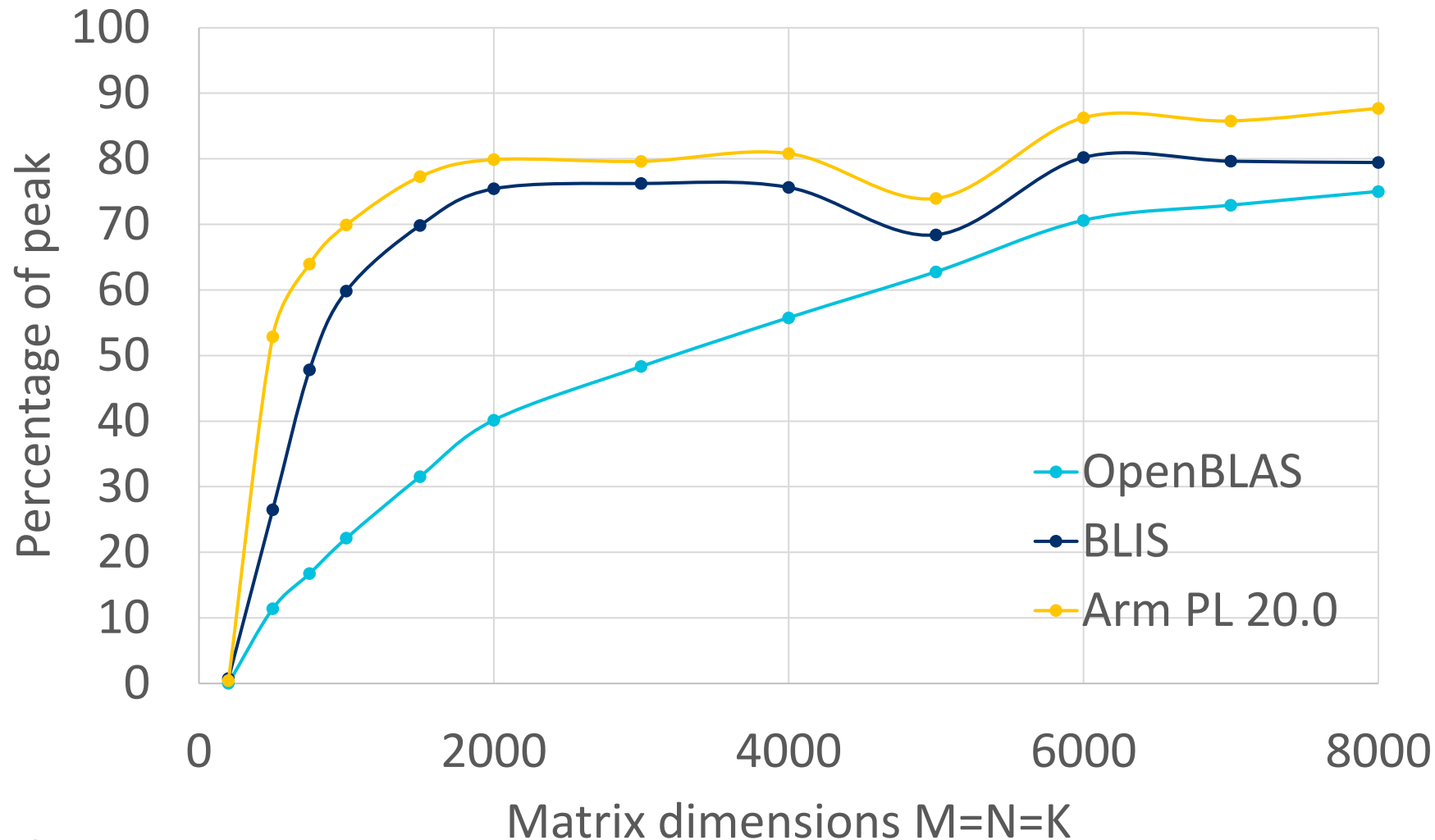
- Generic Armv8-A optimizations by Arm
- Tuning for specific platforms like Marvell ThunderX2 in collaboration with silicon vendors

## Validated and supported by Arm

- Available for a wide range of server-class Arm-based platforms
- Validated with NAG's test suite, a de-facto standard

# Arm Performance Libraries – Leading BLAS performance

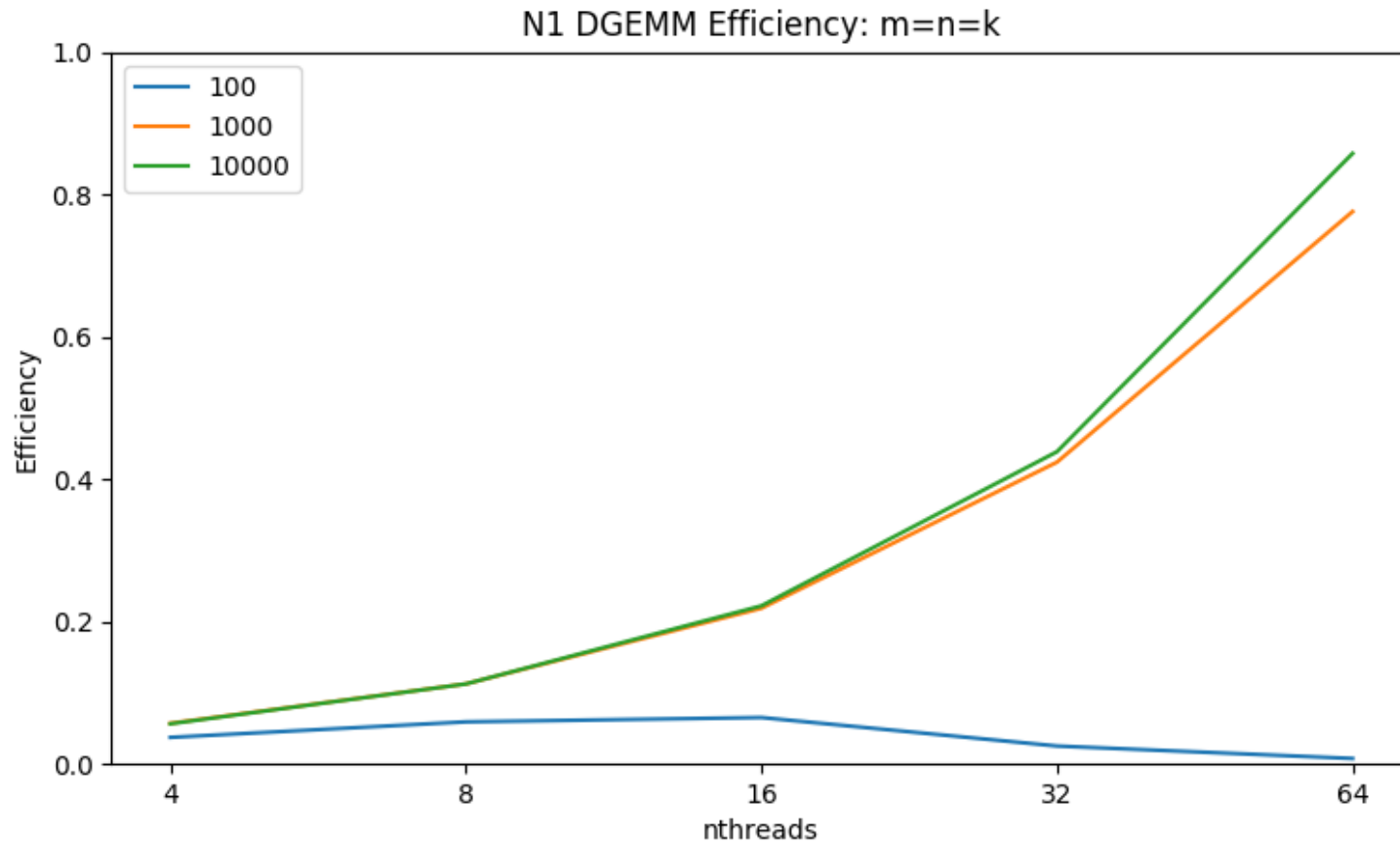
Arm Compiler for Linux 20.0 vs latest OpenBLAS vs latest BLIS



- High serial performance for BLAS level 3 routines, such as GEMMs also have class-leading parallel performance
- Shown is DGEMM on square matrices using 56 threads on a ThunderX2

# Arm Performance Libraries: OpenMP Scaling on N1

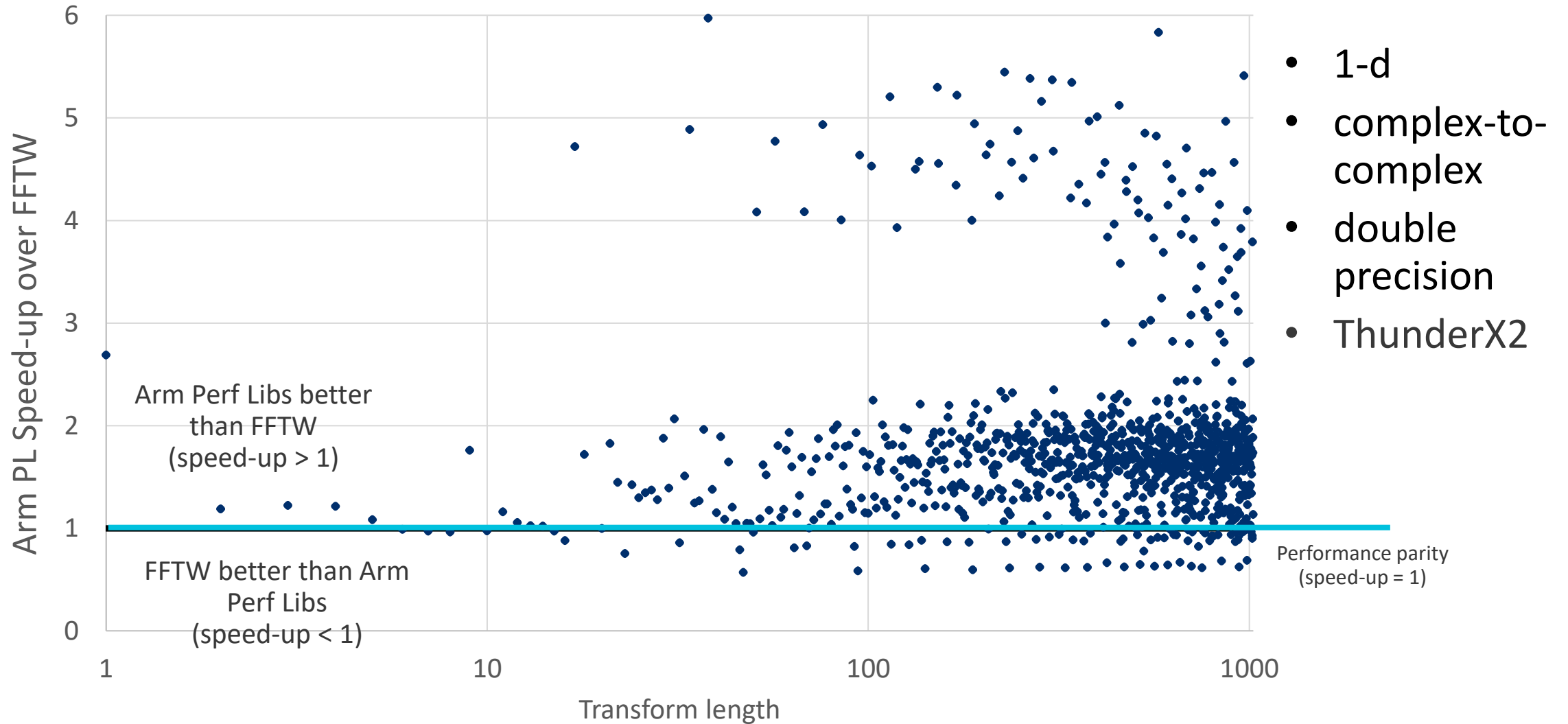
Run on AWS Graviton2



- Shown is DGEMM on square matrices using 64 threads on an AWS Graviton2
- Shown for matrix sizes of 100, 1,000 and 10,000
- Shows up to 85.7% efficiency for large matrices



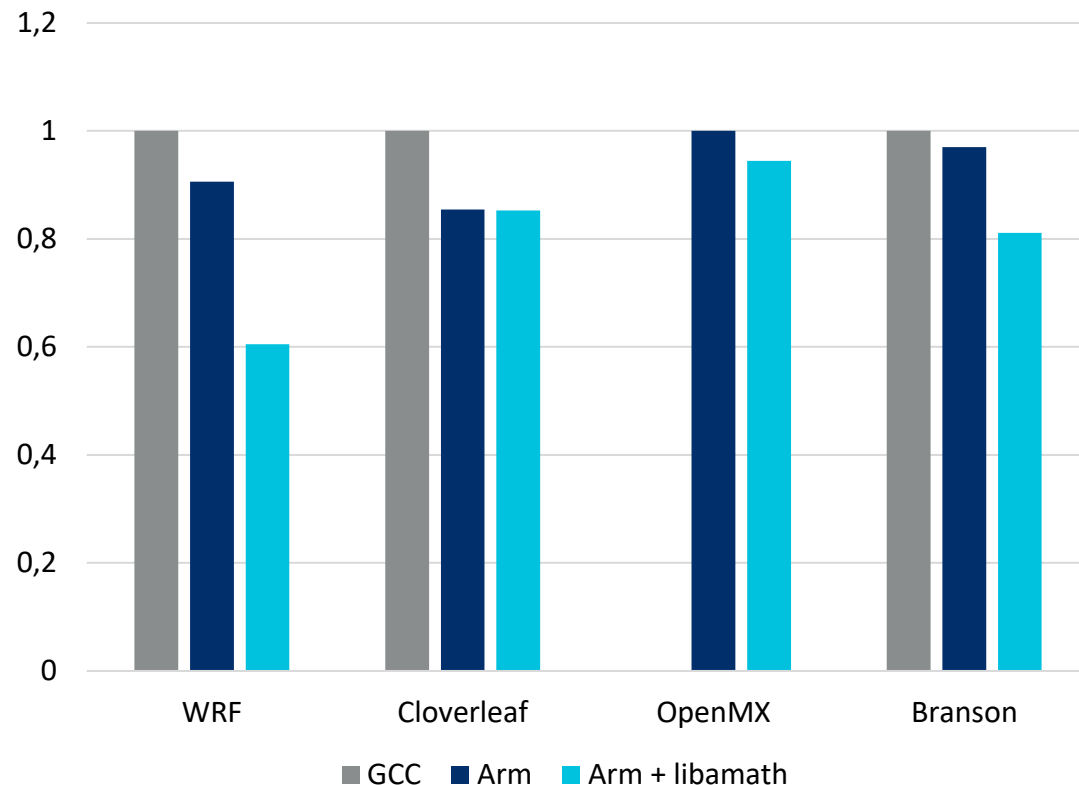
# ArmPL 20.0 FFT vs FFTW 3.3.8



# Arm Performance Libraries – Optimized Math Routines

Open Source: <https://github.com/ARM-software/optimized-routines>

## Normalised runtime



## ArmPL includes libamath and libastring

- Algorithmically better performance than standard library calls
- No loss of accuracy
- Enabled by default with Arm Compiler for Linux
- Double precision implementations of:
  - `erf()`, `erfc()`
  - single and double precision implementations of:
    - `exp()`, `pow()`, `log()`, `log10()`
    - single precision implementations of:
      - `sin()`, `cos()`, `sincos()`
  - Efficient **memory/string** functions from `string.h`
  - Enable **autovectorization** of math and string routines by adding `-armpl` or `-fsimdmath`

*...more to come.*

# Build Tools

All popular build tools are supported on Arm

## Support

- All major build systems and tools:
  - CMake, Make, GNUMake, Spack etc.
  - Spack used internally at Arm.
- Arm supports KitWare etc. to ensure build tools like CMake are stable and supported.
- Arm upstreams any necessary changes to support Arm's commercial tools.
  - e.g. CMake toolchain files for Arm Compilers.

## Compilation Performance

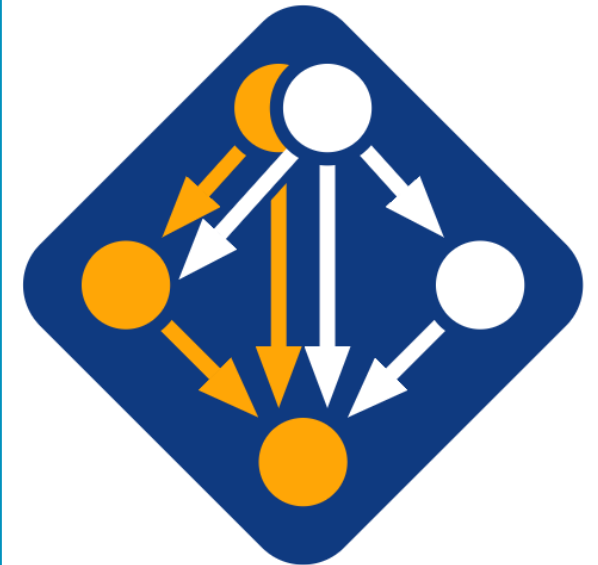
- A data point: ThunderX2 compilation of large code bases is on-par with Intel Skylake
  - Usually faster due to higher core counts.
- GNU compilers run faster than LLVM, but that's not aarch64-specific; same on any arch.



# Application Build Recipes and Spack

Spack is used extensively by Arm

- Multiple places for recipes
  - <https://gitlab.com/arm-hpc/packages/wikis/packages>
  - <https://developer.arm.com/hpc/hpc-software/categories/applications>
  - <https://github.com/UoB-HPC/benchmarks>
- Want to move our knowledge base into Spack
  - <https://github.com/spack/spack>
  - Would like customers to also contribute to Spack
- Ideally get package owners to update their code



# MPI Implementations

Out-of-the-box support for Arm in the latest versions of...

## OpenMPI

- Out-of-the-box support since 3.1.2 (currently 4.0.4)
- [developer.arm.com](https://developer.arm.com) guide
- Upstream contributions
- Used inhouse
- Basis of Bull, Mellanox and Fujitsu K implementations
- Active development from Arm and Arm partners

## MPICH

- Basis of Cray and Intel implementations  
...and MVAPICH

## MVAPICH

- [developer.arm.com](https://developer.arm.com) guide
- Upstream contributions
- Used inhouse
- Basis of Sunway TaihuLight implementation
- Arm investment in OSU
  - Arm hardware & tools

# Parallel Runtime Environments

## Threading, thread placement, and affinity

- POSIX threads 2.0 fully supported.
- Thread placement, pinning, affinity via hwloc, numactl, etc.
- Most SoCs support a simple memory hierarchy partitioned into a minimal number of NUMA nodes, e.g. one NUMA node per CPU socket.
- The goal is to minimize code refactoring for performance and eliminate “guess and check” data movement optimization strategies.

## Dynamically linked libraries and page size

- Users do not need to change anything in their execution environment or workflow to achieve good performance.
  - Demonstrated at multiple application scales at several sites including Sandia and Bristol.
- Tools like [LLNL's Spindle](#) are supported to reduce I/O pressure when loading dynamically linked applications.

# Scientific Computing Libraries

<https://gitlab.com/arm-hpc/packages/wikis/categories/library>

## Package Support

- Trilinos, PETSc, Hypre, SuperLU, ScaLAPACK, NetCDF, HDF5, BLIS, etc.
- Tested to work well with Arm and GNU compilers.
- 54+ packages in Arm's Community Packages Wiki

## Testing and Development

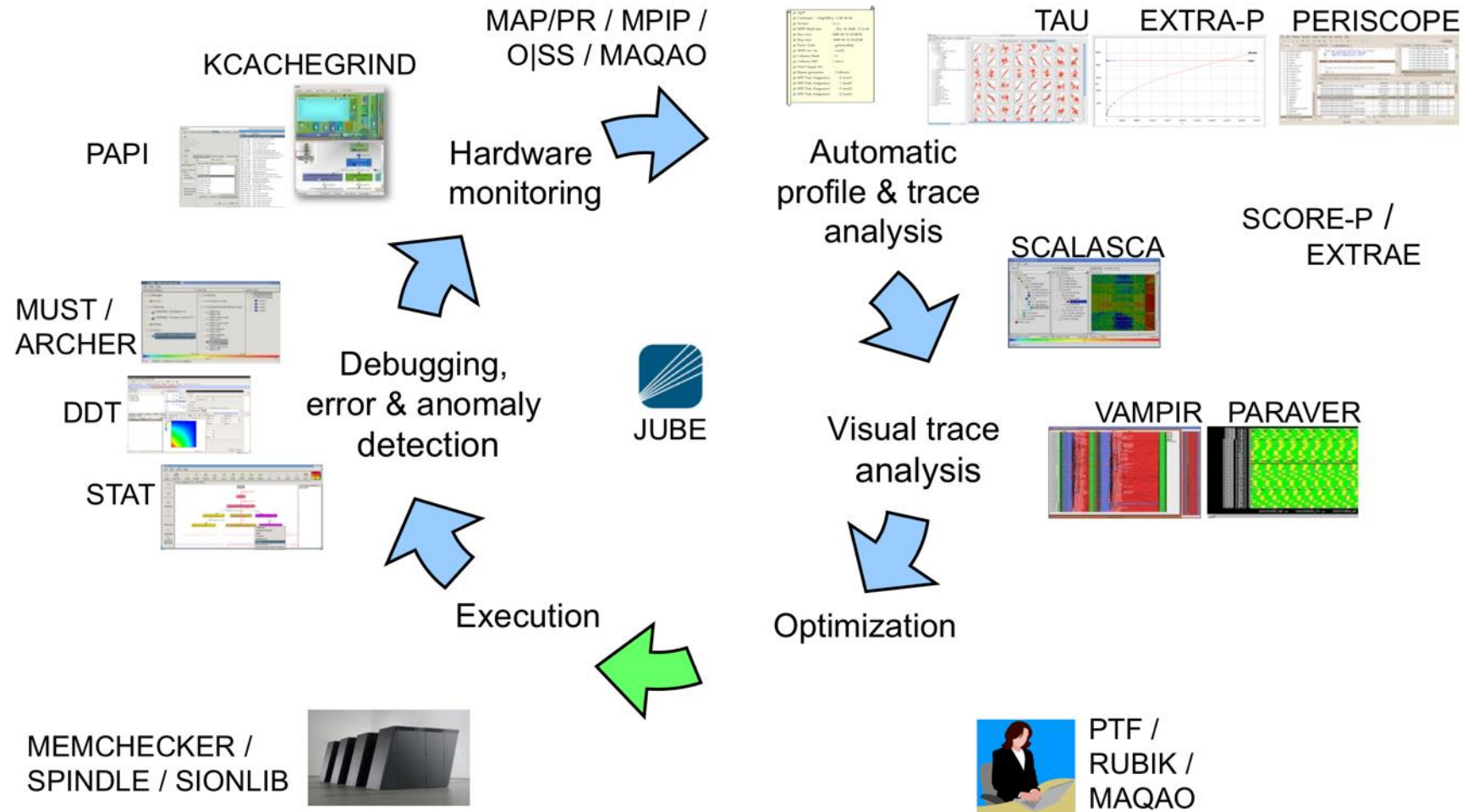
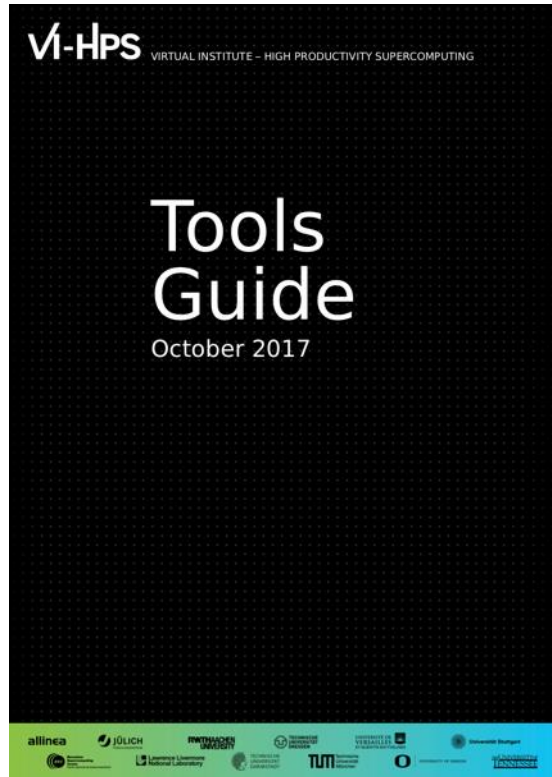
- ThunderX2 access freely available for open source project CI/CD
  - packet.net
  - Verne Global

## Resourcing

- Arm supports communities as part of broader NRE and commercial projects
- Arm provides reactive support to users at key HPC sites worldwide

# Arm Performance Engineering Tools Ecosystem

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





# Hardware Performance Counter Support

Hardware performance counter APIs are fully supported

## PAPI

- Support for many aarch64 server-class CPUs:
  - e.g. ThunderX2
- Marvell planning support for future CPUs e.g. ThunderX4

## perf\_events

- Native HPM API is fully supported
- User applications may:
  - Initialize the HPM
  - Initiate and reset counters
  - Read counters
  - Generate interrupts on counter overflow
  - Register interrupt handlers from each process and thread independently

## Documentation and Tools

- Arm MAP, HPCToolkit, IPM, TAU, ScoreP, etc.
  - HPM values can be accessed by non-privileged users in a secure manner
- Performance metrics derived from multiple counters:
  - Partners provide their own PMU/HPM documentation

The ARM logo is displayed in a white, lowercase, sans-serif font. The background of the slide is a dark grey grid of small white plus signs. Three plus signs are highlighted in color: a green one at the top right, an orange one in the middle, and a yellow one at the bottom right.

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

[www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks)

# Arm Forge Ultimate

A cross-platform toolkit for debugging, profiling and performance analysis



Commercially supported  
by Arm



Fully Scalable



Very user-friendly

## The de-facto standard for HPC development

- Available on the vast majority of the Top500 machines in the world
- Fully supported by Arm on Arm servers, x86, IBM Power, Nvidia GPUs, etc.

## State-of-the art debugging and profiling capabilities

- Powerful and in-depth error detection mechanisms (including memory debugging)
- Sampling-based profiler to identify and understand bottlenecks
- Available at any scale (from serial to petaflop applications)

## Easy to use by everyone

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

# Arm Forge – DDT Parallel Debugger

Switch between MPI ranks and OpenMP threads

Analyze memory usage

Visualize data structures

File View Control Search Tools Window Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

All 24576 processes (0-24575) Paused: 17220 Playing: 7356 Finished: 20 (on nid09271, pid 30269)

Create Group

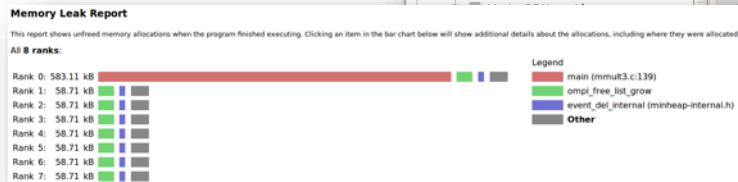
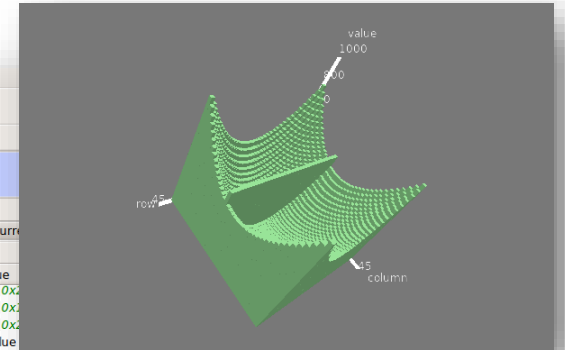
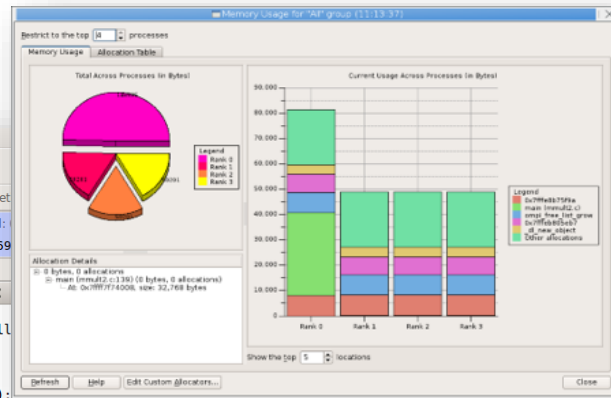
Project Files

Search (Ctrl+K)

- template.cc
- template\_annotator.cc
- template\_cache.cc
- template\_dictionary.cc
- template\_modifiers.cc
- template\_nameLIST.cc
- template\_paths.cc
- template\_string.cc
- timer.c
- timers.c
- timing.c
- UnitConverter.cc
- util.c
- utilityFunctions.cc
- Vector3D.cc

```

546 if (allpicks[i].val != -1)
547     allpicks[ntsamples++] = allpicks[i];
548 }
549
550 /* Sort all the picks */
551 ikvsortii(ntsamples, allpicks);
552
553
554 /* Select the final splitters. Set the boundaries to simplify coding */
555 for (i=1; i<nps; i++)
556     mypicks[i] = allpicks[i*ntsamples/nps];
557 mypicks[0].key = IDX_MIN;
558 mypicks[nps].key = IDX_MAX;
559
560
561 WCOREPOP; /* free allpicks */
562
563
564 STOPTIMER(ctrl, ctrl->AuxTmr2);
565 STARTTIMER(ctrl, ctrl->AuxTmr3);
566
567 /* Compute the number of elements that belong to each bucket */
    
```

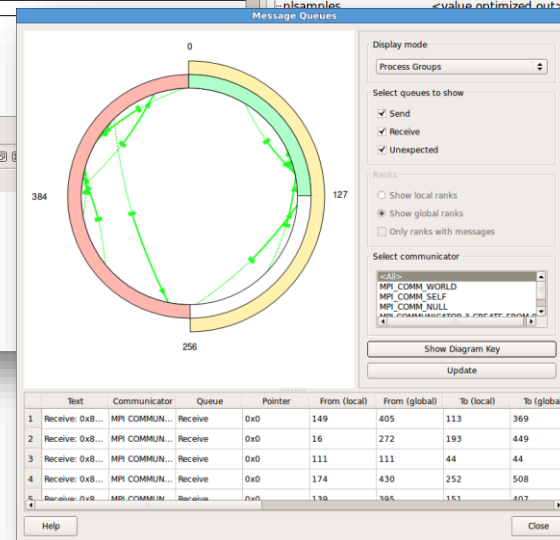


Export data and connect to continuous integration

Tracepoints Tracepoint Output Logbook

```

onMaster (SimulationMaster.cc:63)
se (SimulationMaster.cc:154)
hemelb::geometry::GeometryReader::LoadAndDecompose (GeometryReader.cc:188)
hemelb::geometry::GeometryReader::OptimiseDomainDecomposition (GeometryReader.cc:809)
hemelb::geometry::decomposition::OptimisedDecomposition::OptimisedDecomposition (OptimisedDecomposition.cc:65)
hemelb::geometry::decomposition::OptimisedDecomposition::CallParmetis (OptimisedDecomposition.cc:181)
ParMETIS_V3_PartGeomKWay (gkmetis.c:90)
libparmetis::CoordinatePartition (xyzpart.c:58)
libparmetis::PseudoSampleSort (xyzpart.c:56)
    
```



Display pending communications

# Arm Forge – MAP Multi-node Low-overhead Profiler

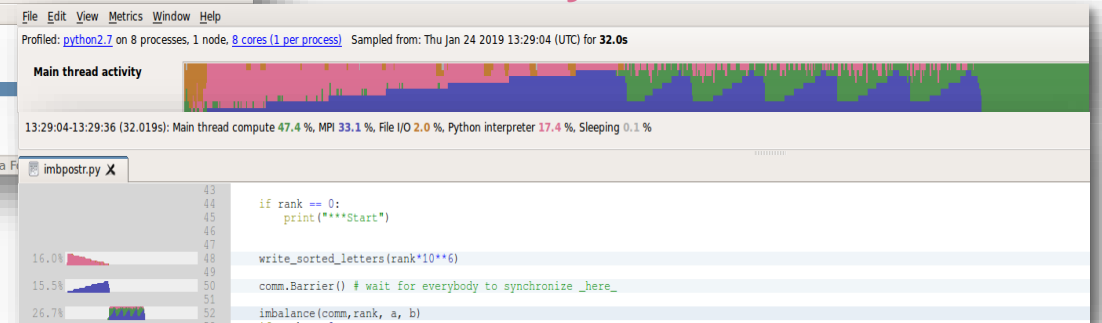
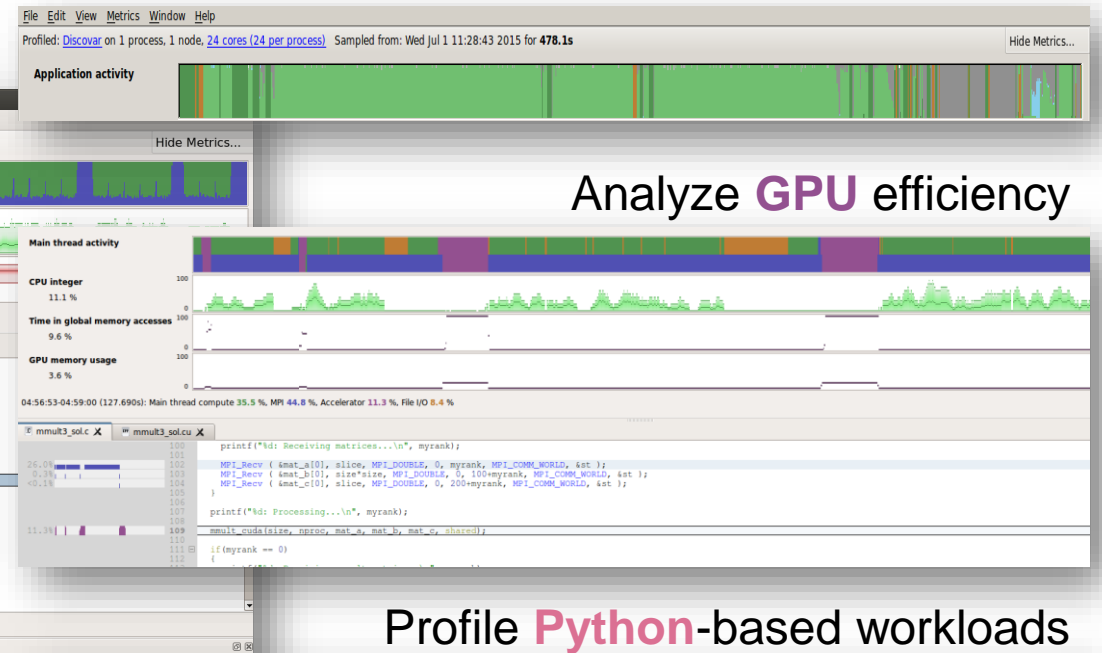
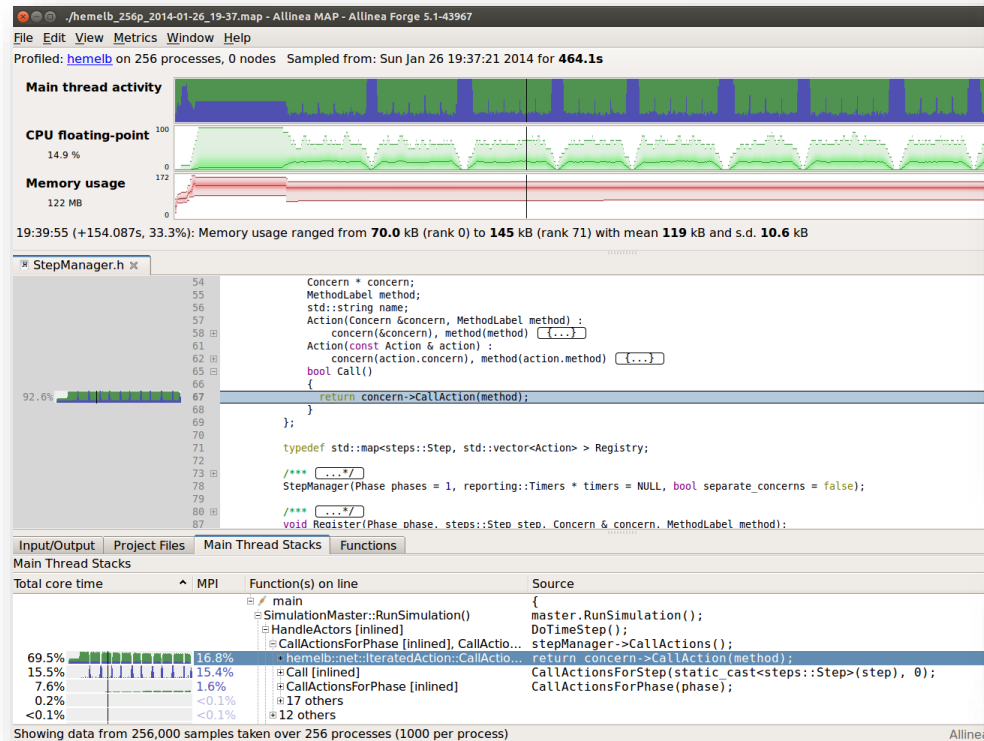
Understand **MPI/CPU/IO** operations thanks to timelines and metrics

Inspect **OpenMP** activity

Analyze **GPU** efficiency

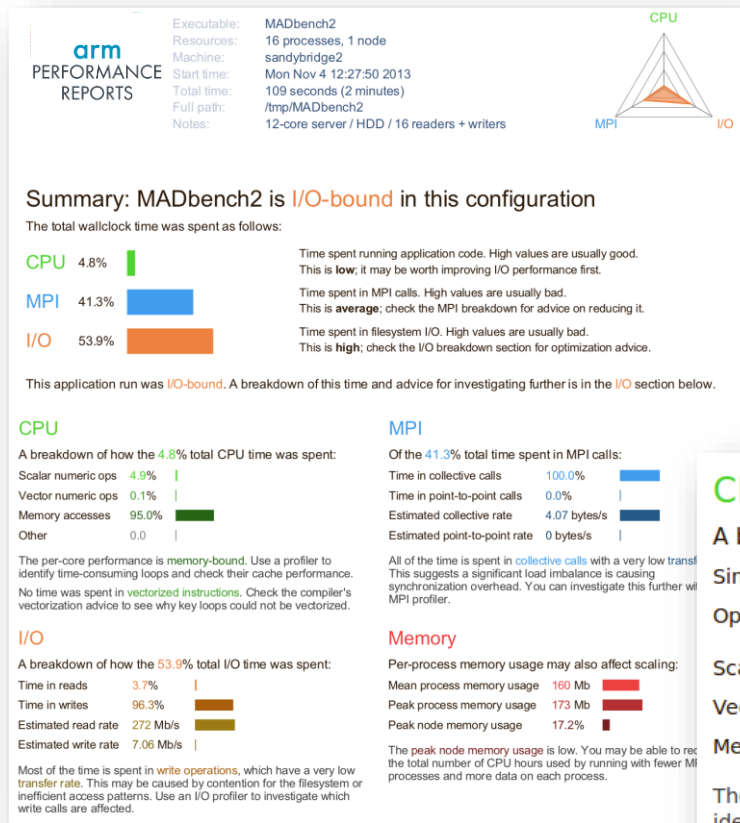
Profile **Python**-based workloads

Investigate annotated source code and stack view



# Arm Performance Reports Application Analysis Tool

Analyze all performance aspects in a single HTML or TXT file



Qualify the type of workload

Inspect key metrics on SIMD, multithreading, IO, MPI efficiency and many more...

## CPU

A breakdown of the 91.2% CPU time:

Single-core code	30.6%
OpenMP regions	69.4%
Scalar numeric ops	9.5%
Vector numeric ops	0.0%
Memory accesses	78.1%

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

No time is spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

Follow guidance advices for your next steps and maximize output