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

CPU Engagement Models With Arm

Core License

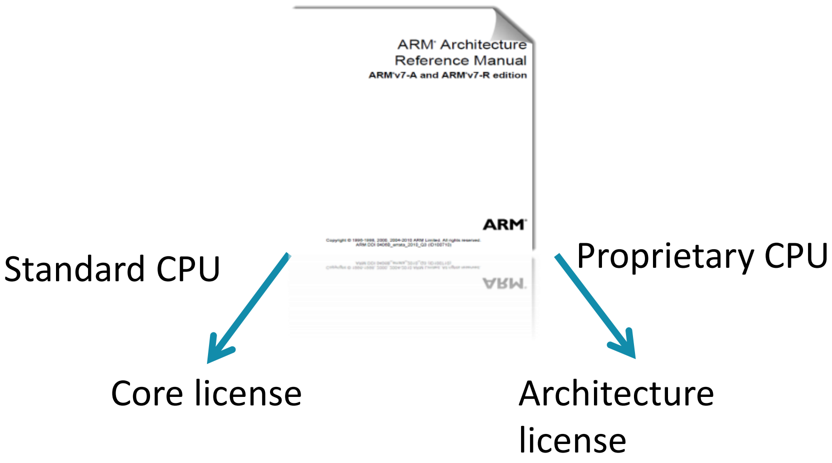
Partner licenses complete microarchitecture design

- Wide choices available
- Many different A, R & M products

CPU differentiation through:

- Flexible configuration options
- Wide implementation envelope with different process technologies

Range of licensing & engagement models possible



ARM Architecture Reference Manual
ARMv7-A and ARMv7-R edition

ARM

Standard CPU

Core license

Proprietary CPU

Architecture license

Architecture License

Partner designs complete CPU microarchitecture from scratch

- Clean room – no reference to Arm core designs

Freedom to develop any design

- Must conform to the rules & programmers model of a given architecture variant
- Must pass Arm architecture validation to preserve software compatibility

Long term strategic investment

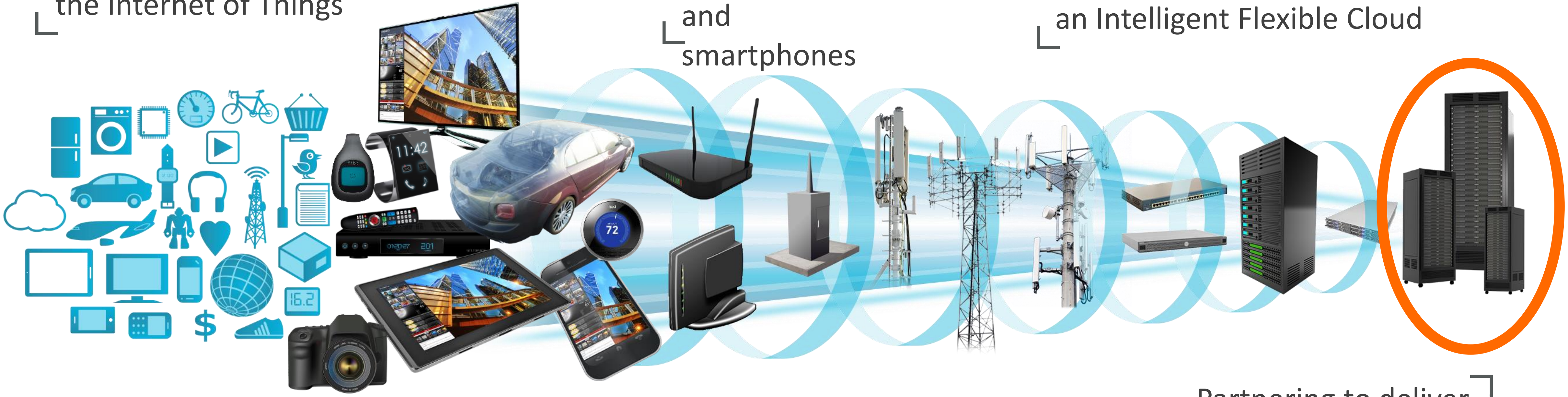
ARM's mission

- Deploy energy-efficient ARM-based technology, wherever computing happens…

Leading in wearables and the Internet of Things

~85% share of laptops, tablets, and smartphones

Driving the transformation of the network and data center to an Intelligent Flexible Cloud

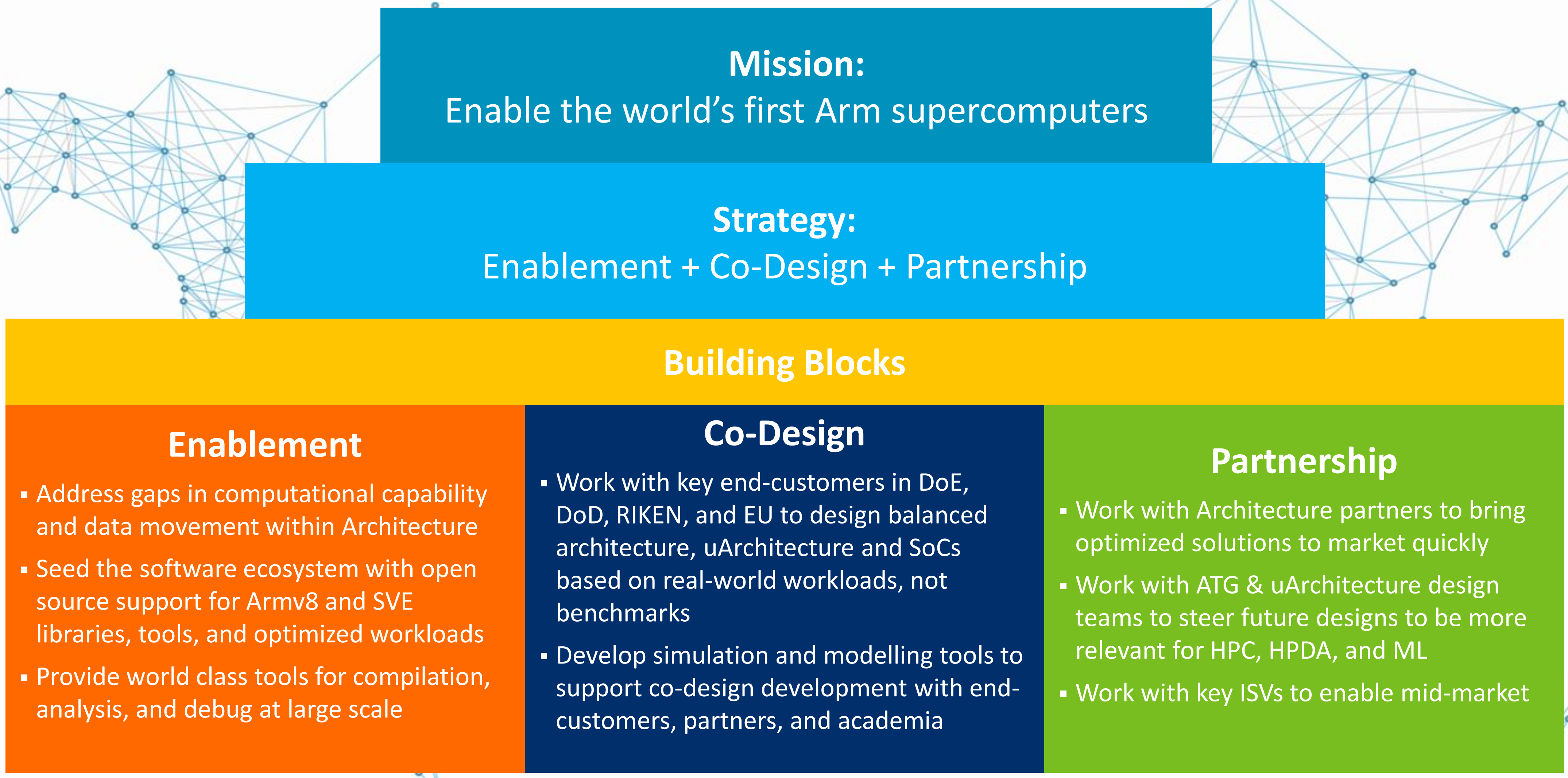


Enabling innovation and creativity with embedded intelligence

Taking mobile computing to the next four billion people

Partnering to deliver data center efficiency

HPC strategy



Mission:
Enable the world’s first Arm supercomputers

Strategy:
Enablement + Co-Design + Partnership

Building Blocks

Enablement

- Address gaps in computational capability and data movement within Architecture
- Seed the software ecosystem with open source support for Armv8 and SVE libraries, tools, and optimized workloads
- Provide world class tools for compilation, analysis, and debug at large scale

Co-Design

- Work with key end-customers in DoE, DoD, RIKEN, and EU to design balanced architecture, uArchitecture and SoCs based on real-world workloads, not benchmarks
- Develop simulation and modelling tools to support co-design development with end-customers, partners, and academia

Partnership

- Work with Architecture partners to bring optimized solutions to market quickly
- Work with ATG & uArchitecture design teams to steer future designs to be more relevant for HPC, HPDA, and ML
- Work with key ISVs to enable mid-market

Deployment

arm

Arm CPU vendors for server and HPC segment

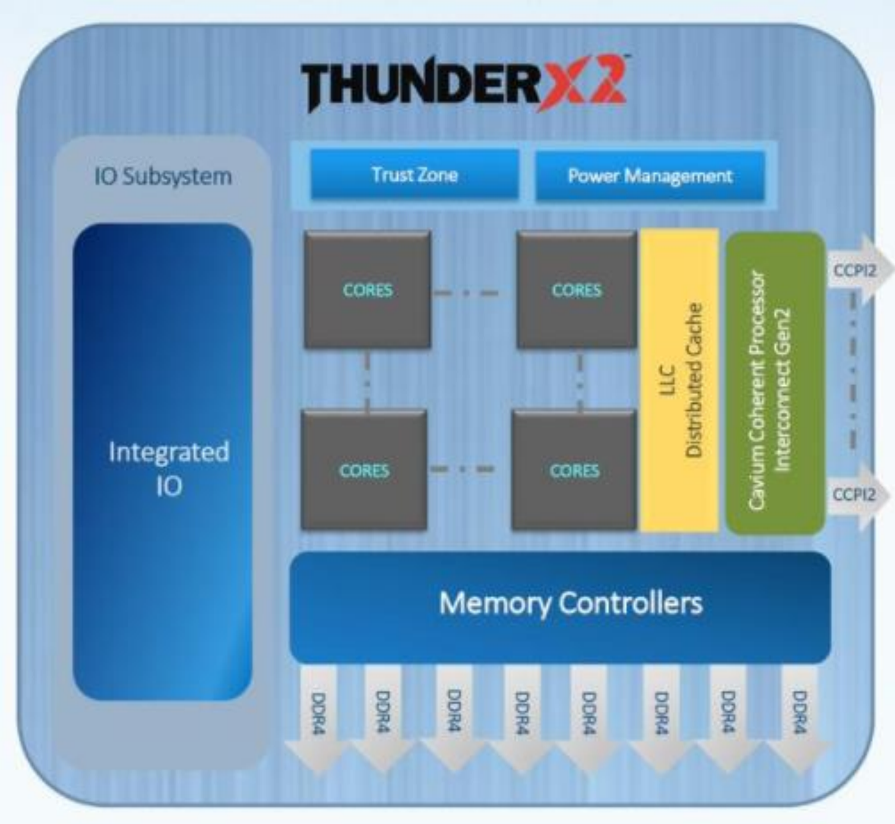
Publicly announced

Vendor	Type of License	HPC Focus	Current CPU	Future CPUs
Cavium/Marvell	Architecture License	Yes	Cavium ThunderX2	Available under NDA
Ampere	Architecture License	No	eMAG Skylark (14nm)	Available under NDA
Fujitsu	Architecture License	Yes	A64FX (7nm) (SVE 512bit)	Public commitment to develop future Arm CPUs
Amazon	Architecture License	No	Graviton	Available under NDA
Arm	Architecture License	Yes	Neoverse N1	Public commitment to develop future Arm CPUs

Other vendors – HXT (China), Phytium (China). Potential future vendors EPI (European processor initiative)

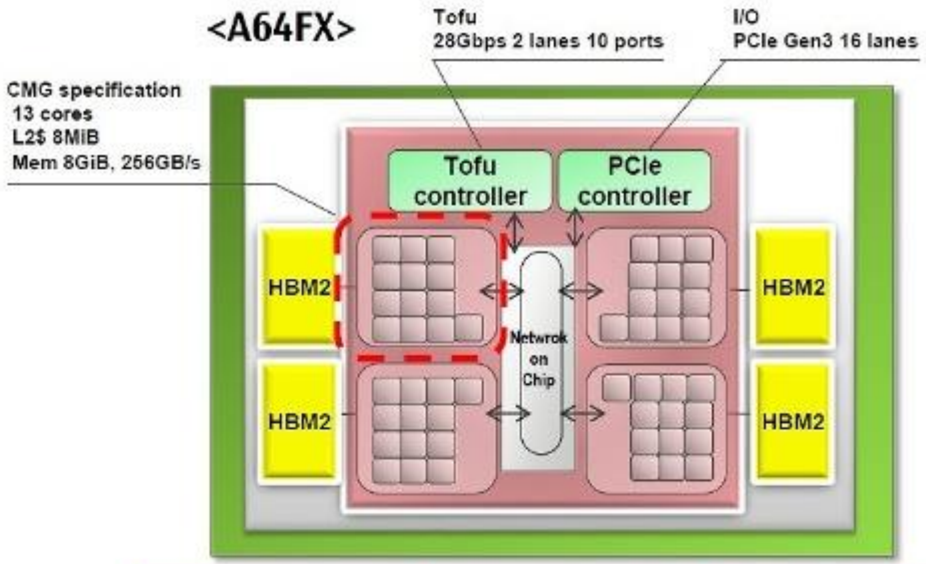
Cavium ThunderX2 CN99XX

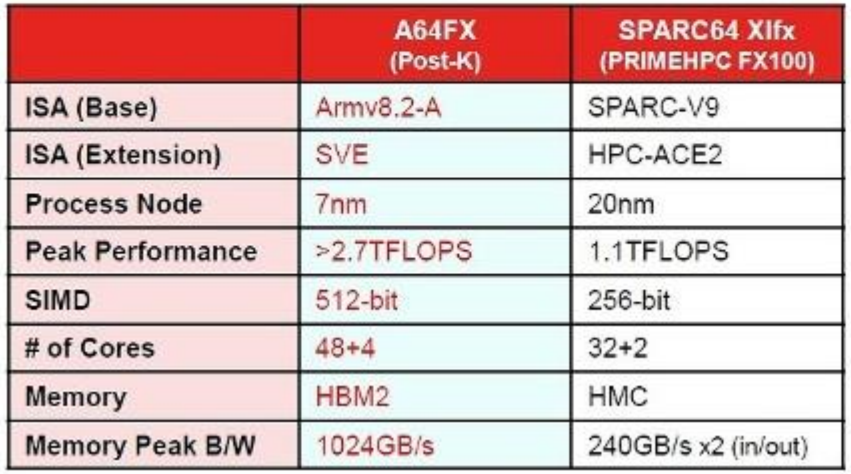
- Cavium’s next generation 64-bit Arm processor
  - Taken from Broadcom Vulcan
- 32 cores @ 2.2 GHz (other SKUs available)
  - 4 Way SMT
  - Fully out of order execution
  - 8 Memory channels
  - 1 DPC - DDR4 – 2667MT/s
  - => 512 GB / socket
- Available in dual SoC configurations
  - CCPI2 interconnect
  - 108x PCIe 3.0 lanes (dual socket)
  - 180-200w / socket
- No SVE vectorisation
  - But still 128-bit NEON vectorisation



Fujitsu A64FX

- Chip designed for RIKEN POST-K machine
- 48 core 64-bit Armv8 processor
  - + dedicated OS cores
- With SVE vectorisation
  - 512 bit vector length
- 32 GB HBM
  - No DDR
  - 1 TB/s bandwidth
- TOFU 3 interconnect

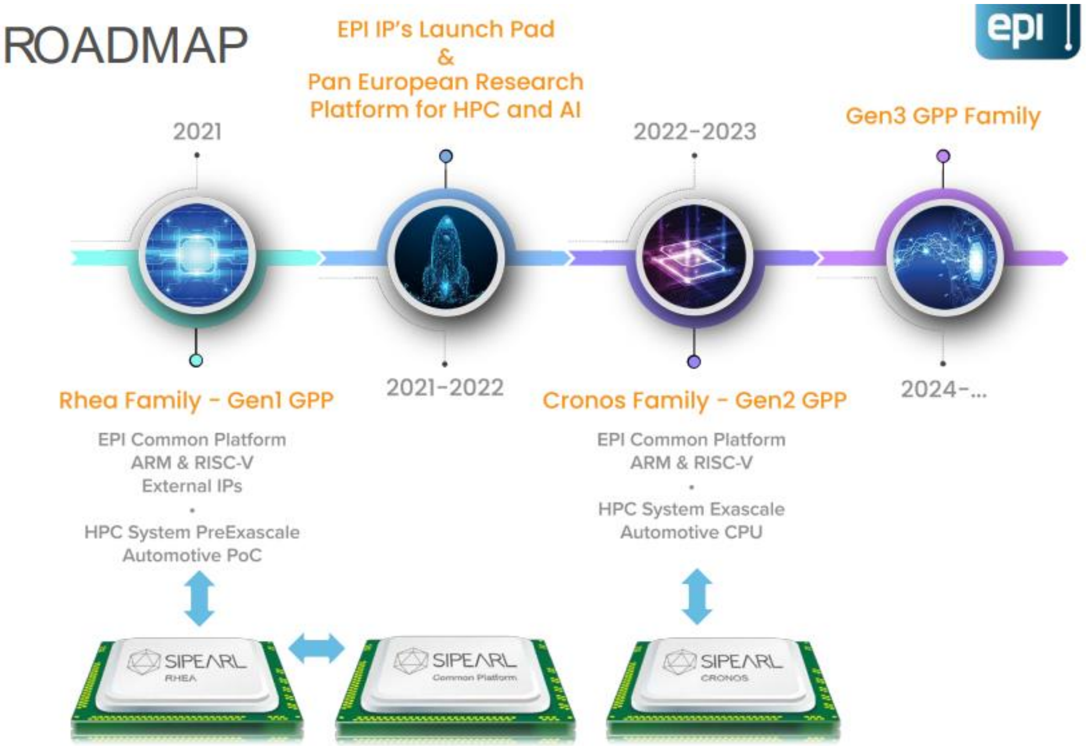




	A64FX (Post-K)	SPARC64 XIfx (PRIMEHPC FX100)
ISA (Base)	Armv8.2-A	SPARC-V9
ISA (Extension)	SVE	HPC-ACE2
Process Node	7nm	20nm
Peak Performance	>2.7TFLOPS	1.1TFLOPS
SIMD	512-bit	256-bit
# of Cores	48+4	32+2
Memory	HBM2	HMC
Memory Peak B/W	1024GB/s	240GB/s x2 (in/out)

EPI

European Processor Initiative



ROADMAP

EPI IP's Launch Pad & Pan European Research Platform for HPC and AI

Gen3 GPP Family

2021

2021-2022

2022-2023

2024-...

Rhea Family - Gen1 GPP

EPI Common Platform
ARM & RISC-V
External IPs

HPC System PreExascale
Automotive PoC

Cronos Family - Gen2 GPP

EPI Common Platform
ARM & RISC-V

HPC System Exascale
Automotive CPU

2017: AToS, HPE and Cray Announce Products

AToS Sequana (ISC 2017)

HPE Apollo 70 (SC 2017)

Cray XC50 (SC 2017)

Deployments: Astra at Sandia

Mapping performance to real-world mission applications

- 2592 HPE Apollo 70 nodes with 145,152 cores in total
- 2.3 theoretical peak petaflops @1.2MW power
- Cavium ThunderX2 processors
- Mellanox EDR InfiniBand
- 8 Memory Channels per socket (332 TB memory with 885 TB/s bandwidth)

Deployments: HPE’s Comanche Collaboration

Early access to Cavium ThunderX2 systems that became Apollo 70

Engagements in HPE Comanche program have accelerated adoption

- We have been able to assess the state of fundamental software stacks, such as MPI and NUMA capabilities
  - Collaborative work here especially great with all partners focusing on interoperability issues
  - Examples include fixing bugs with kernels, MPI drivers and OpenMP thread placement
  - Optimization of packages, environment and execution configurations

Hewlett Packard Enterprise

Argonne NATIONAL LABORATORY

arm

Lawrence Livermore National Laboratory

CAVIUM

Los Alamos NATIONAL LABORATORY EST. 1943

Mellanox TECHNOLOGIES

AMD

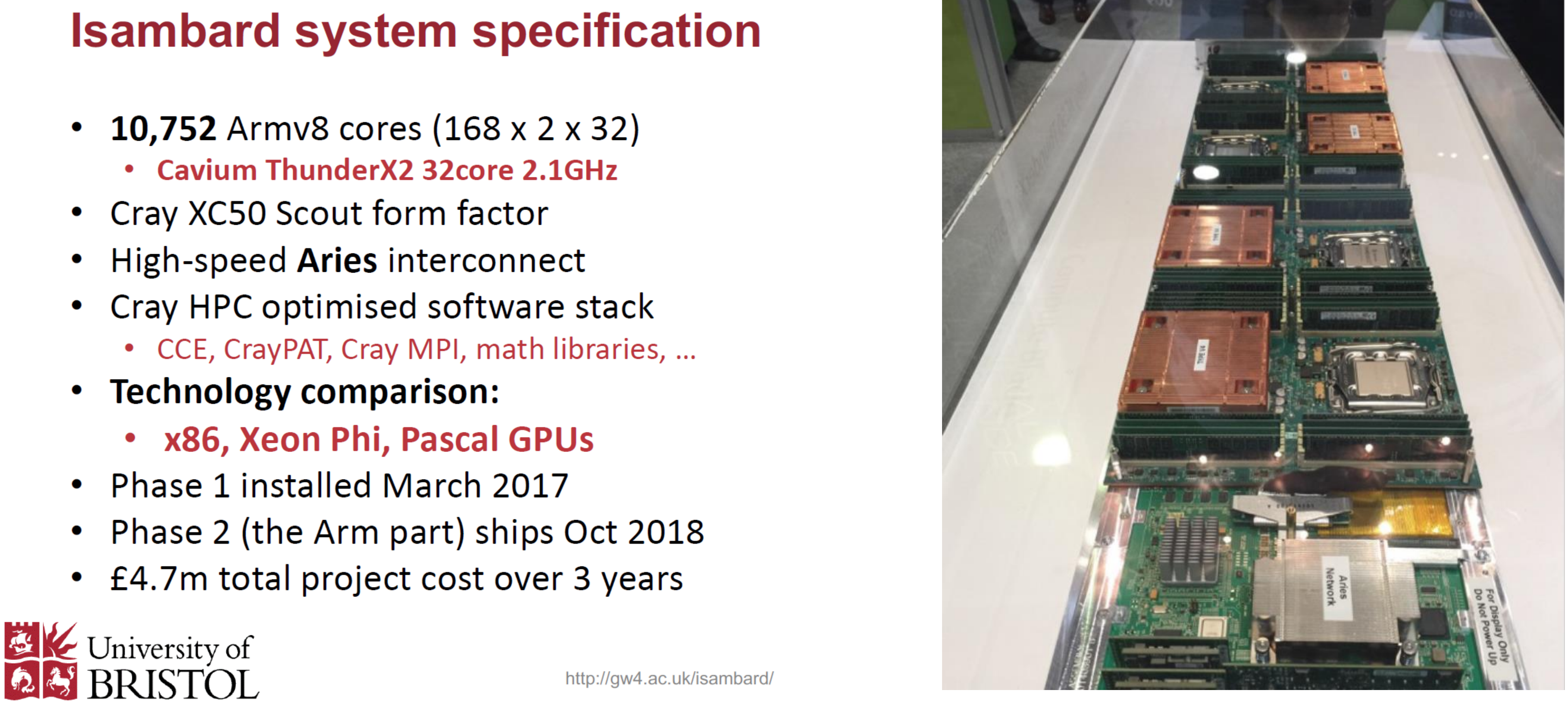
OAK RIDGE National Laboratory

redhat

Sandia National Laboratories

Over 1,000 processors delivered | LLNL TOSS stack ported and demoed | InfiniBand optimized

Deployments: Isambard @ GW4



Isambard system specification

- **10,752** Armv8 cores (168 x 2 x 32)
  - **Cavium ThunderX2 32core 2.1GHz**
- Cray XC50 Scout form factor
- High-speed **Aries** interconnect
- Cray HPC optimised software stack
  - CCE, CrayPAT, Cray MPI, math libraries, …
- **Technology comparison:**
  - **x86, Xeon Phi, Pascal GPUs**
- Phase 1 installed March 2017
- Phase 2 (the Arm part) ships Oct 2018
- £4.7m total project cost over 3 years

University of BRISTOL

http://gw4.ac.uk/isambard/

Source: http://uob-hpc.github.io/assets/Isambard_ECMWF_HPC_WS_Sep_2018.pdf

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:

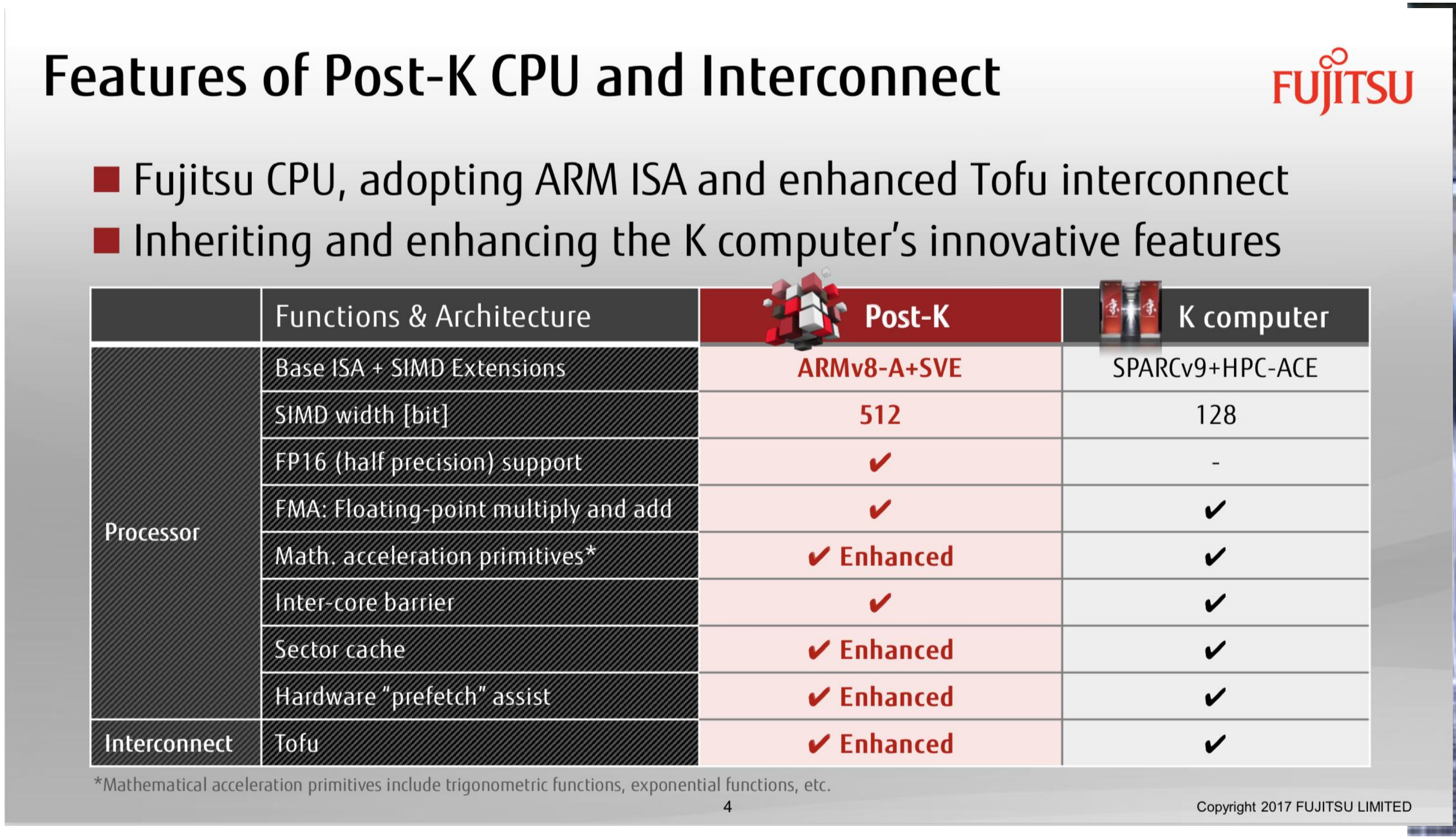
- 64 HPE Apollo 70 systems, each with two 32-core Cavium ThunderX2 processors (i.e. 4096 cores per system), 128GB of memory and Mellanox InfiniBand interconnects
- SUSE Linux Enterprise Server for HPC

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

RIKEN Post-K



Features of Post-K CPU and Interconnect

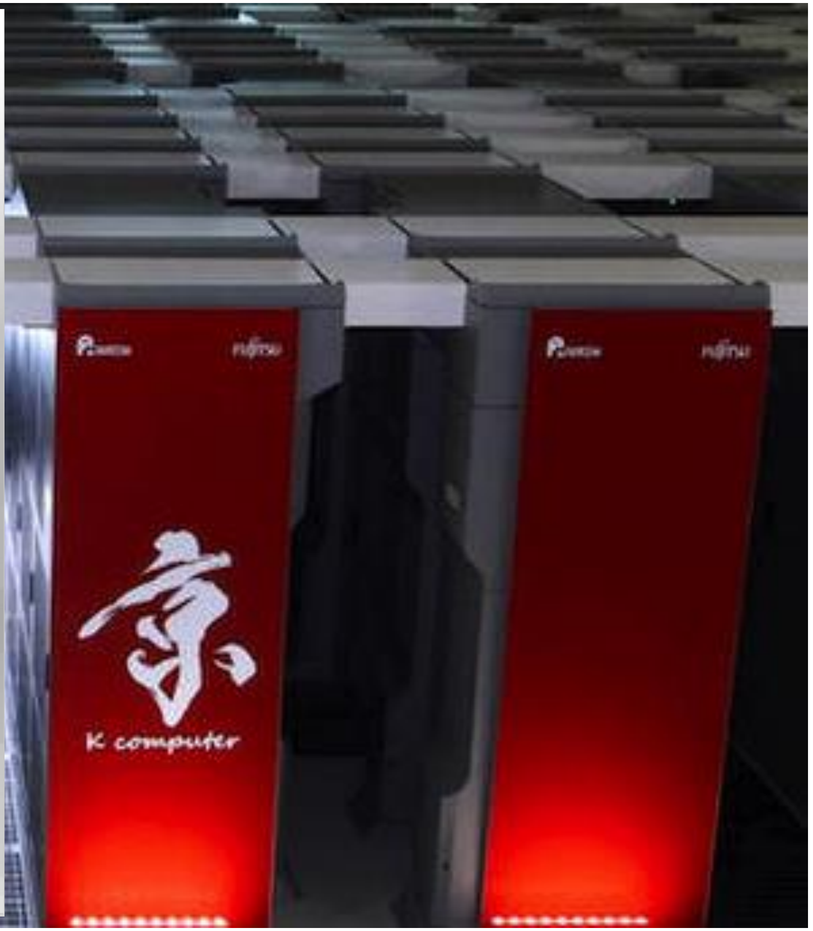
FUJITSU

- Fujitsu CPU, adopting ARM ISA and enhanced Tofu interconnect
- Inheriting and enhancing the K computer's innovative features

	Functions & Architecture	Post-K	K computer
Processor	Base ISA + SIMD Extensions	ARMv8-A+SVE	SPARCv9+HPC-ACE
	SIMD width [bit]	512	128
	FP16 (half precision) support	✔	-
	FMA: Floating-point multiply and add	✔	✔
	Math. acceleration primitives*	✔ Enhanced	✔
	Inter-core barrier	✔	✔
	Sector cache	✔ Enhanced	✔
	Hardware "prefetch" assist	✔ Enhanced	✔
Interconnect	Tofu	✔ Enhanced	✔

*Mathematical acceleration primitives include trigonometric functions, exponential functions, etc.

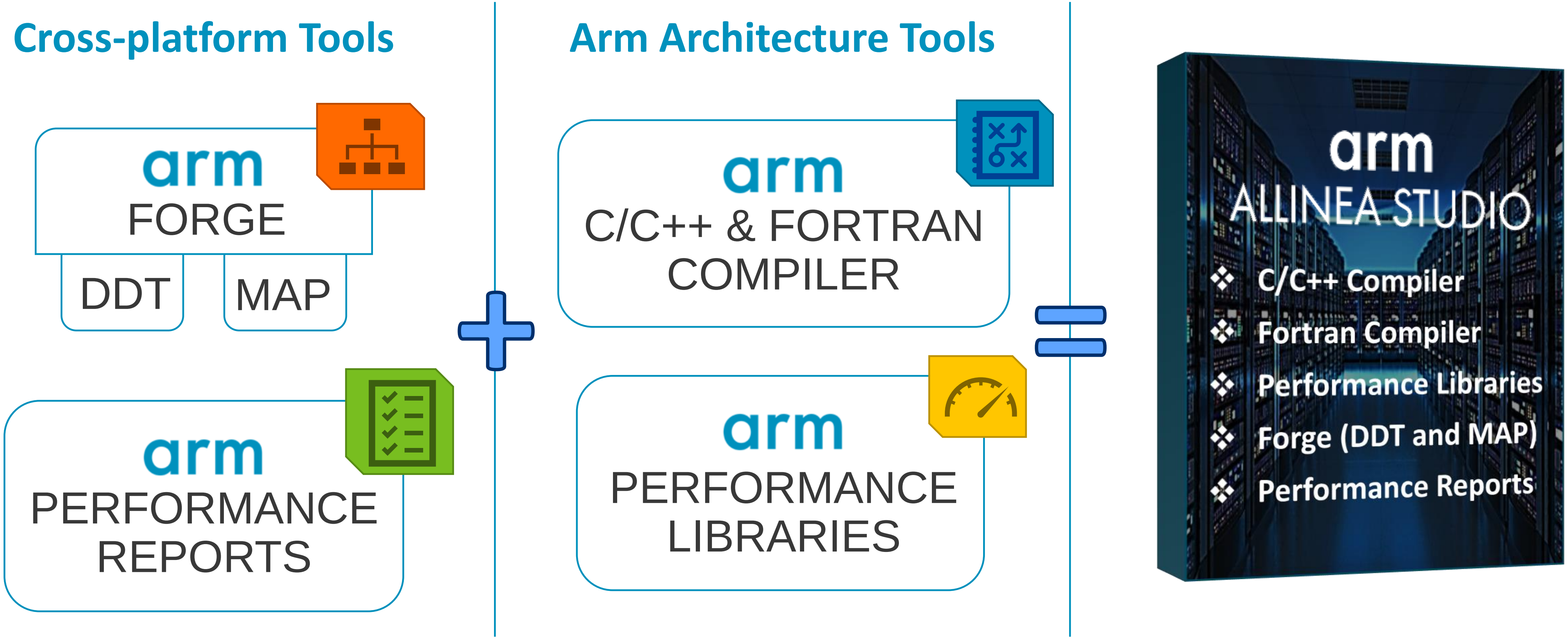
4



arm

Arm’s solution for HPC application development and porting

Combines cross-platform tools with Arm only tools for a comprehensive solution



Cross-platform Tools

arm FORGE

DDT

MAP

arm PERFORMANCE REPORTS

Arm Architecture Tools

arm C/C++ & FORTRAN COMPILER

arm PERFORMANCE LIBRARIES

arm ALLINEA STUDIO

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

Tools

Arm Allinea Studio

A quick glance at what is in Arm Allinea Studio

C++

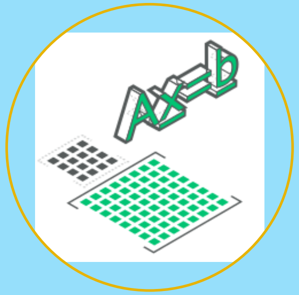
C/C++ Compiler

- C++ 14 support
- OpenMP 4.5 without offloading
- SVE ready

Fortran

Fortran Compiler

- Fortran 2003 support
- Partial Fortran 2008 support
- OpenMP 3.1
- SVE ready



Performance Libraries

- Optimized math libraries
- BLAS, LAPACK and FFT
- Threaded parallelism with OpenMP

Forge (DDT and MAP)

- Profile, Tune and Debug
- Scalable debugging with DDT
- Parallel Profiling with MAP

Performance Reports

- Analyze your application
- Memory, MPI, Threads, I/O, CPU metrics

Tuned by Arm for a wide-range of server-class Arm-based platforms

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 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

Arm Performance Libraries

Optimized BLAS, LAPACK and FFT

Commercial 64-bit Armv8-A math libraries

- Commonly used low-level math routines - BLAS, LAPACK and FFT
- Validated with NAG’s test suite, a de-facto standard

Best-in-class performance with commercial support

- Tuned by Arm for Cortex-A72, Cortex-A57 and Cortex-A53
- Maintained and supported by Arm for a wide range of Arm-based SoCs
  - Including Cavium ThunderX and ThunderX2 CN99 cores

Silicon partners can provide tuned micro-kernels for their SoCs

- Partners can contribute directly through open source route
- Parallel tuning within our library increases overall application performance

Performance on par
with best-in-class math libraries

Commercially Supported
by Arm

Validated with
NAG test suite

Arm Allinea Studio

A quick glance at what is in Arm Allinea Studio

C++

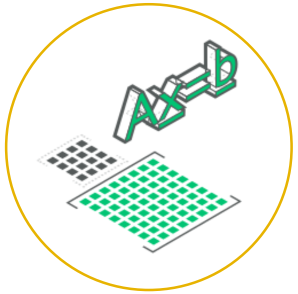
C/C++ Compiler

- C++ 14 support
- OpenMP 4.5 without offloading
- SVE ready

Fortran

Fortran Compiler

- Fortran 2003 support
- Partial Fortran 2008 support
- OpenMP 3.1
- SVE ready



Performance Libraries

- Optimized math libraries
- BLAS, LAPACK and FFT
- Threaded parallelism with OpenMP

Forge (DDT and MAP)

- Profile, Tune and Debug
- Scalable debugging with DDT
- Parallel Profiling with MAP

Performance Reports

- Analyze your application
- Memory, MPI, Threads, I/O, CPU metrics

Tuned by Arm for a wide-range of server-class Arm-based platforms

Arm Forge

An interoperable toolkit for debugging and profiling

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 x86, IBM Power, Nvidia GPUs, etc.

State-of-the art debugging and profiling capabilities

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

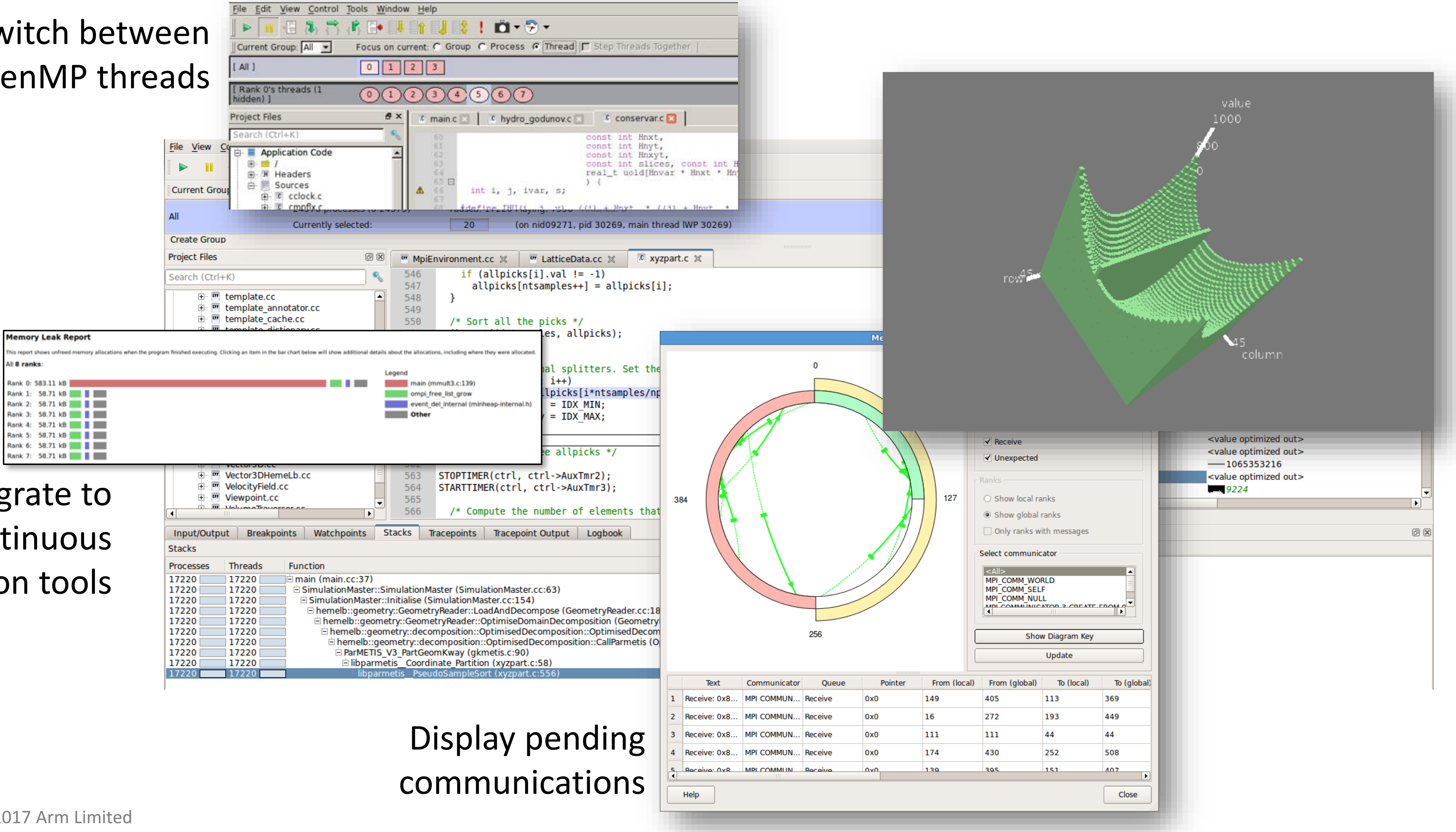
Easy to use by everyone

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

Migrate and debug application

Switch between OpenMP threads

Visualise data structures



```
const int Hnxt,
const int Hnyt,
const int Hnxyt,
const int slices, const int H
real_t uold[Hnvar * Hnxt * Hn
) {
int i, j, ivar, s;
```

```
      if (allpicks[i].val != -1)
        allpicks[ntsamples++] = allpicks[i];
    }

  /* Sort all the picks */
```

```
  STOPTIMER(ctrl, ctrl->AuxTmr2);
  STARTTIMER(ctrl, ctrl->AuxTmr3);

  /* Compute the number of elements that
```

```
<value optimized out>
<value optimized out>
1065353216
<value optimized out>
9224
```

Memory Leak Report

This report shows unfreed memory allocations when the program finished executing. Clicking an item in the bar chart below will show additional details about the allocations, including where they were allocated.

All 8 ranks:

Rank 0:	583.11 kB
Rank 1:	58.71 kB
Rank 2:	58.71 kB
Rank 3:	58.71 kB
Rank 4:	58.71 kB
Rank 5:	58.71 kB
Rank 6:	58.71 kB
Rank 7:	58.71 kB

Legend
- 🟥 main (mmult3.c:139)
- 🟩 ompi_free_list_grow
- 🟦 event_del_internal (minheap-internal.h)
- ⬛ **Other**

Stacks

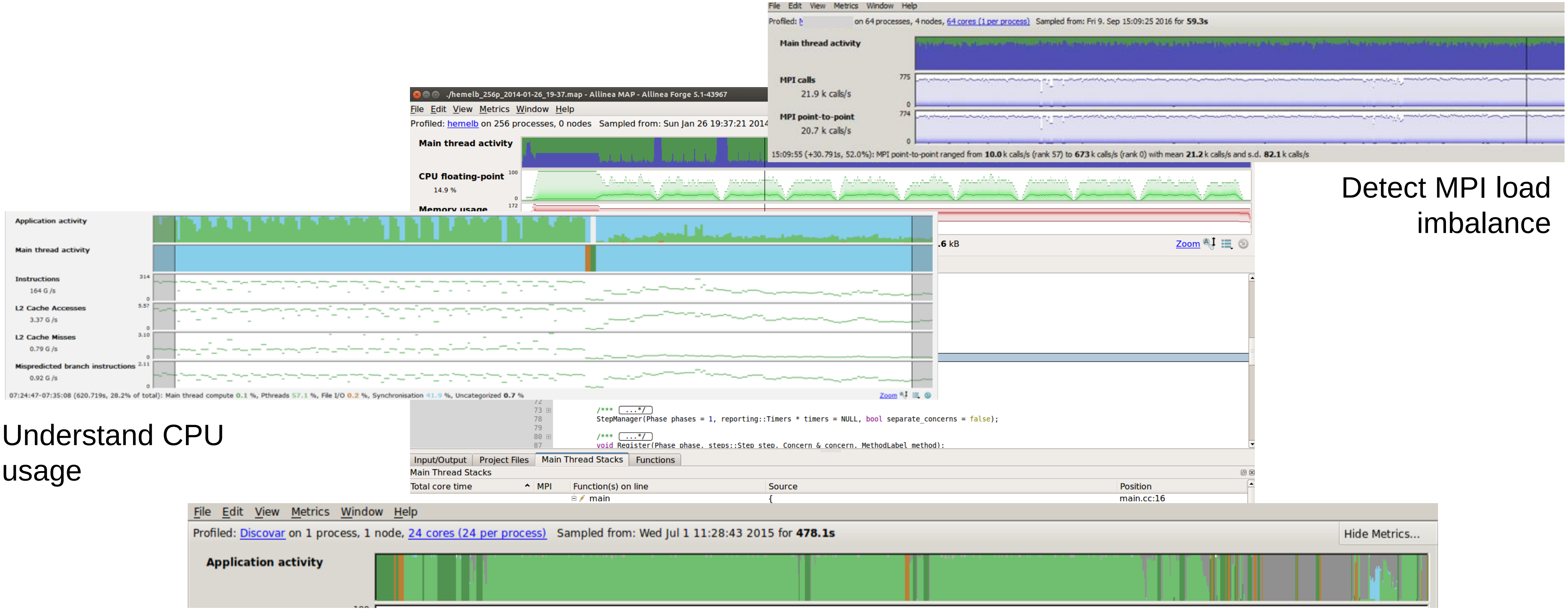
```
Processes  Threads  Function
17220      17220    main (main.cc:37)
17220      17220      SimulationMaster::SimulationMaster (SimulationMaster.cc:63)
17220      17220        SimulationMaster::Initialise (SimulationMaster.cc:154)
17220      17220          hemelb::geometry::GeometryReader::LoadAndDecompose (GeometryReader.cc:18
17220      17220            hemelb::geometry::GeometryReader::OptimiseDomainDecomposition (Geometry
17220      17220              hemelb::geometry::decomposition::OptimisedDecomposition::OptimisedDecom
17220      17220                hemelb::geometry::decomposition::OptimisedDecomposition::CallParmetis (O
17220      17220                  ParMETIS_V3_PartGeomKway (gkmetis.c:90)
17220      17220                    libparmetis__Coordinate_Partition (xyzpart.c:58)
17220      17220                      libparmetis__PseudoSampleSort (xyzpart.c:556)
```

	Text	Communicator	Queue	Pointer	From (local)	From (global)	To (local)	To (global)
1	Receive: 0x8...	MPI COMMUN...	Receive	0x0	149	405	113	369
2	Receive: 0x8...	MPI COMMUN...	Receive	0x0	16	272	193	449
3	Receive: 0x8...	MPI COMMUN...	Receive	0x0	111	111	44	44
4	Receive: 0x8...	MPI COMMUN...	Receive	0x0	174	430	252	508
5	Receive: 0x8	MPI COMMUN	Receive	0x0	139	395	151	407

Integrate to continuous integration tools

Display pending communications

Optimize on Arm



File Edit View Metrics Window Help

Profiled: [redacted] on 64 processes, 4 nodes, 64 cores (1 per process) Sampled from: Fri 9. Sep 15:09:25 2016 for **59.3s**

15:09:55 (+30.791s, 52.0%): MPI point-to-point ranged from **10.0** k calls/s (rank 57) to **673** k calls/s (rank 0) with mean **21.2** k calls/s and s.d. **82.1** k calls/s

./hemelb_256p_2014-01-26_19-37.map - Allinea MAP - Allinea Forge 5.1-43967

File Edit View Metrics Window Help

Profiled: hemelb on 256 processes, 0 nodes Sampled from: Sun Jan 26 19:37:21 2014

07:24:47-07:35:08 (620.719s, 28.2% of total): Main thread compute 0.1 %, Pthreads 57.1 %, File I/O 0.2 %, Synchronisation 41.9 %, Uncategorized **0.7** %

```
        /*** ...*/
        StepManager(Phase phases = 1, reporting::Timers * timers = NULL, bool separate_concerns = false);

        /*** ...*/
        void Register(Phase phase, steps::Step step, Concern & concern, MethodLabel method);
```

Total core time	MPI	Function(s) on line	Source	Position
		main	{	main.cc:16

File Edit View Metrics Window Help

Profiled: Discovar on 1 process, 1 node, 24 cores (24 per process) Sampled from: Wed Jul 1 11:28:43 2015 for **478.1s**

Detect MPI load imbalance

Understand CPU usage

Identify regions of high OpenMP synchronisation

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

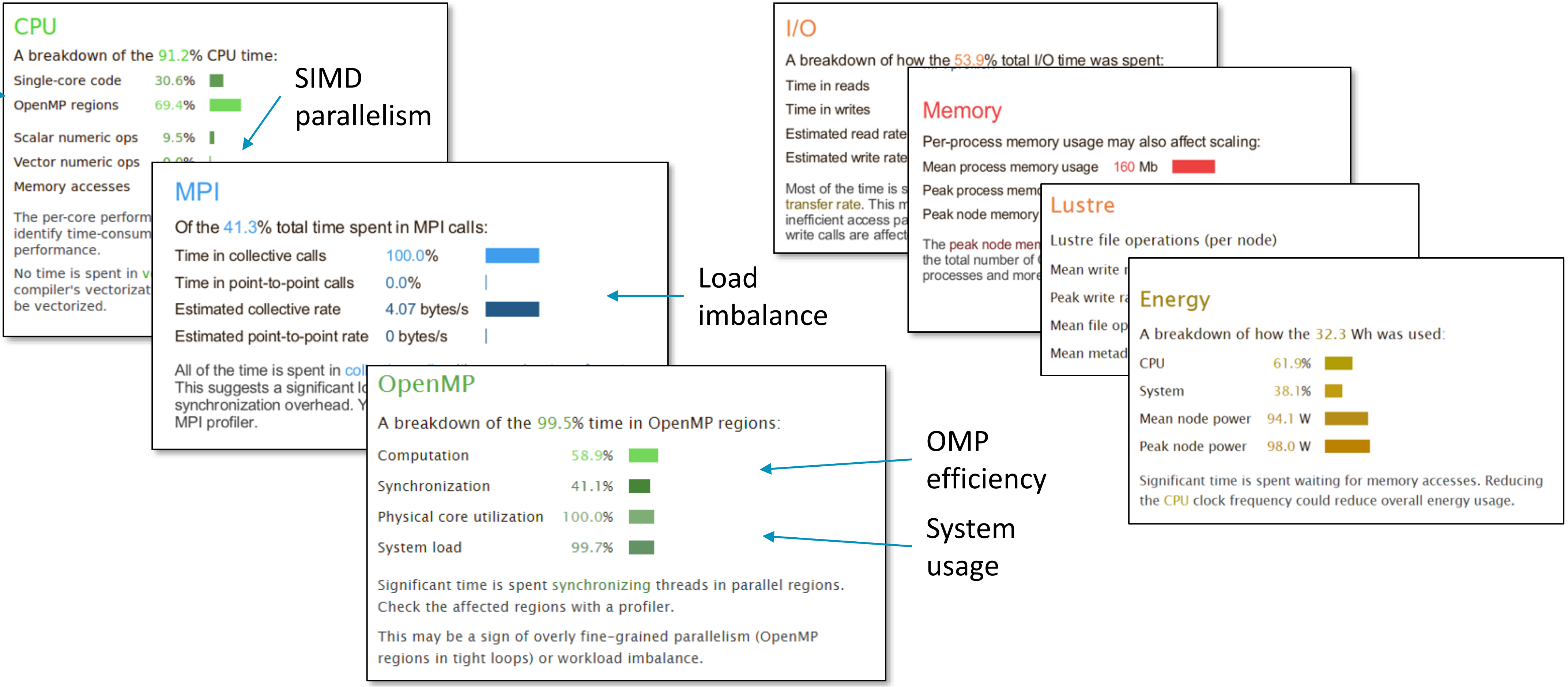
Build a culture of application performance & efficiency awareness

- 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)

Maximize efficiency



Multi-threaded parallelism

SIMD parallelism

Load imbalance

OMP efficiency

System usage

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		

The per-core perform
identify time-consum
performance.

No time is spent in v
compiler's vectorizat
be vectorized.

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%	
Estimated collective rate	4.07 bytes/s	🟦
Estimated point-to-point rate	0 bytes/s	

All of the time is spent in col
This suggests a significant lo
synchronization overhead. Y
MPI profiler.

OpenMP

A breakdown of the 99.5% time in OpenMP regions:

Computation	58.9%	🟩
Synchronization	41.1%	🟩
Physical core utilization	100.0%	🟩
System load	99.7%	🟩

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.

I/O

A breakdown of how the 53.9% total I/O time was spent:

Time in reads

Time in writes

Estimated read rate

Estimated write rate

Most of the time is s
transfer rate. This m
inefficient access pa
write calls are affect

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	160 Mb	🟥
Peak process memo		
Peak node memory		

The peak node mem
the total number of
processes and more

Lustre

Lustre file operations (per node)

Mean write r

Peak write ra

Mean file op

Mean metad

Energy

A breakdown of how the 32.3 Wh was used:

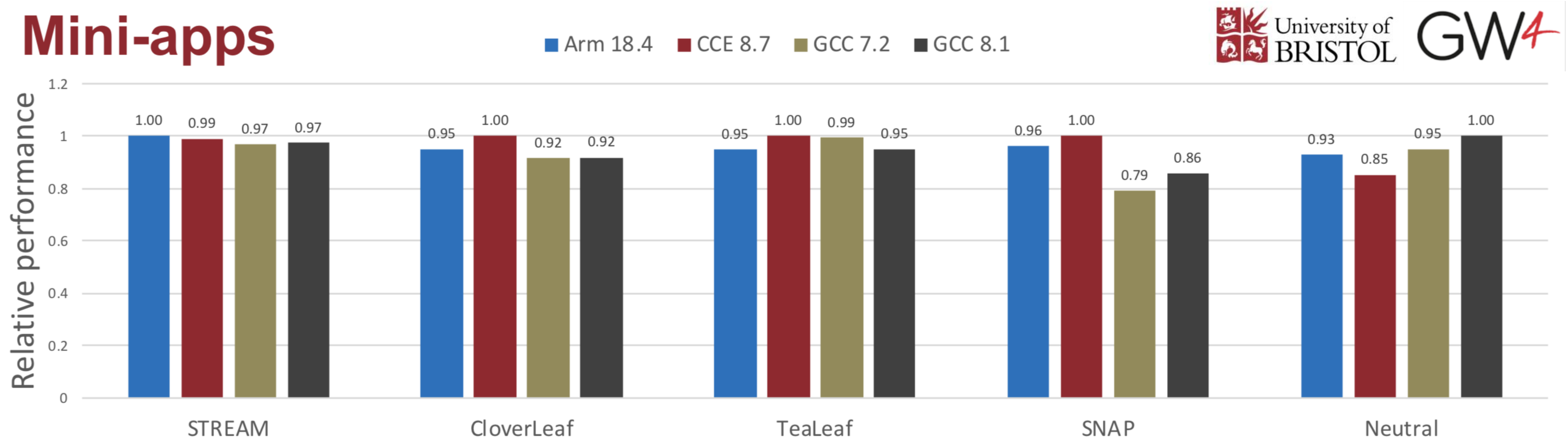
CPU	61.9%	🟨
System	38.1%	🟨
Mean node power	94.1 W	🟨
Peak node power	98.0 W	🟧

Significant time is spent waiting for memory accesses. Reducing the CPU clock frequency could reduce overall energy usage.

Applications

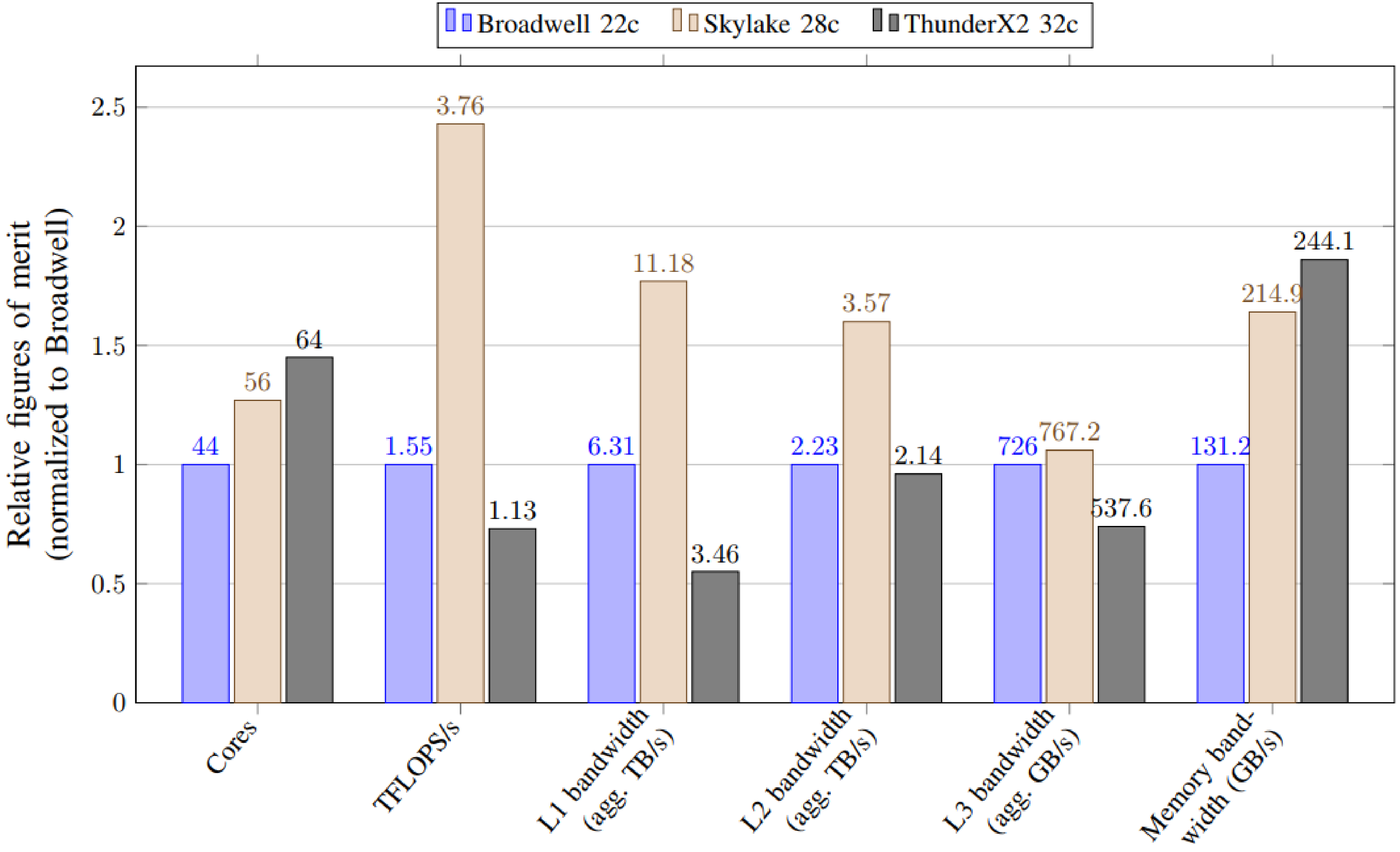
Application performance – GW4

Results presented at the Arm Research Summit, 2018 (Higher is better)



- For each application, results are relative to best configuration
- CCE/Arm fastest in all but one case

Application performance – GW4



Application performance – GW4

Figure 3. CloverLeaf scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems

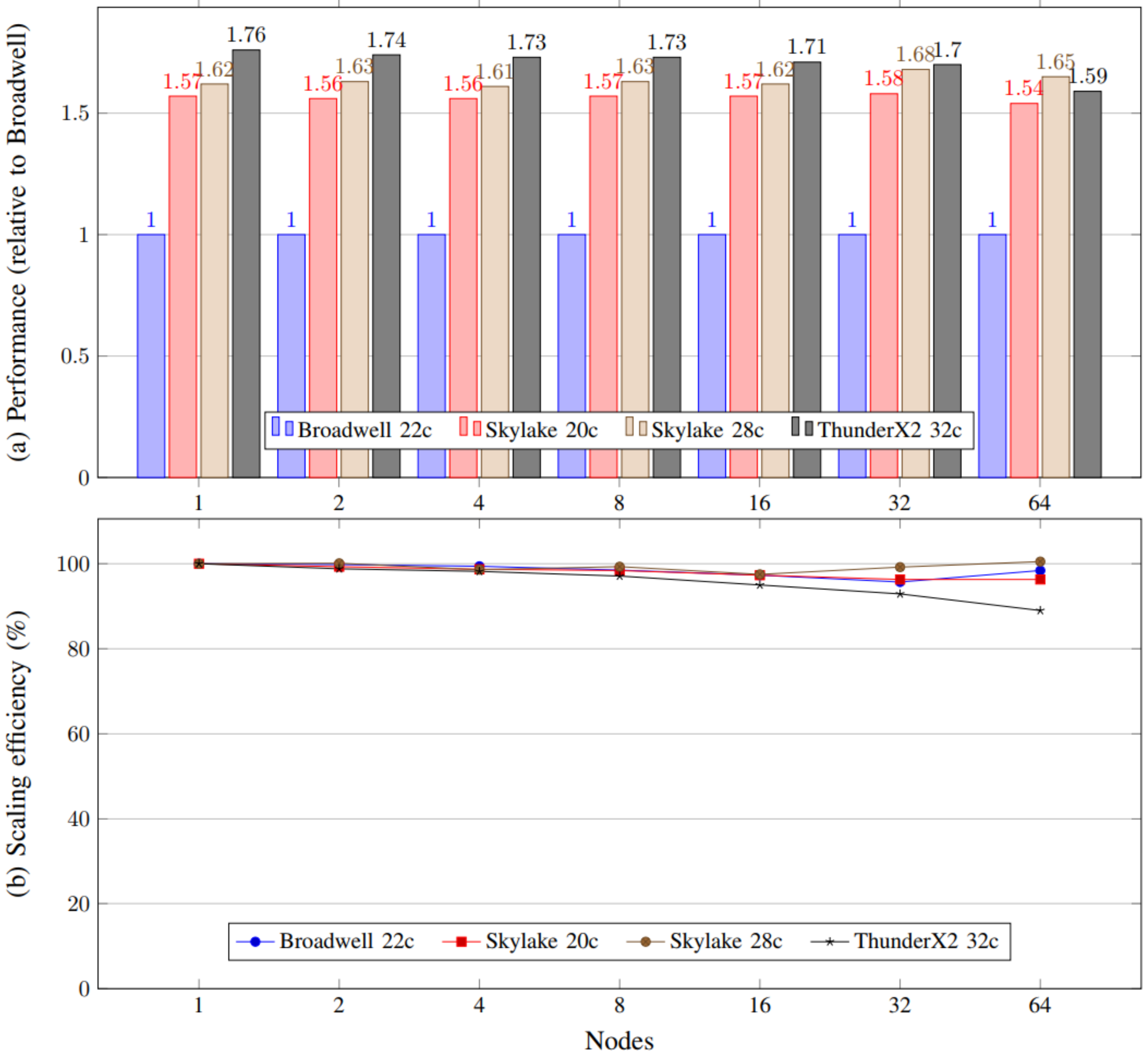
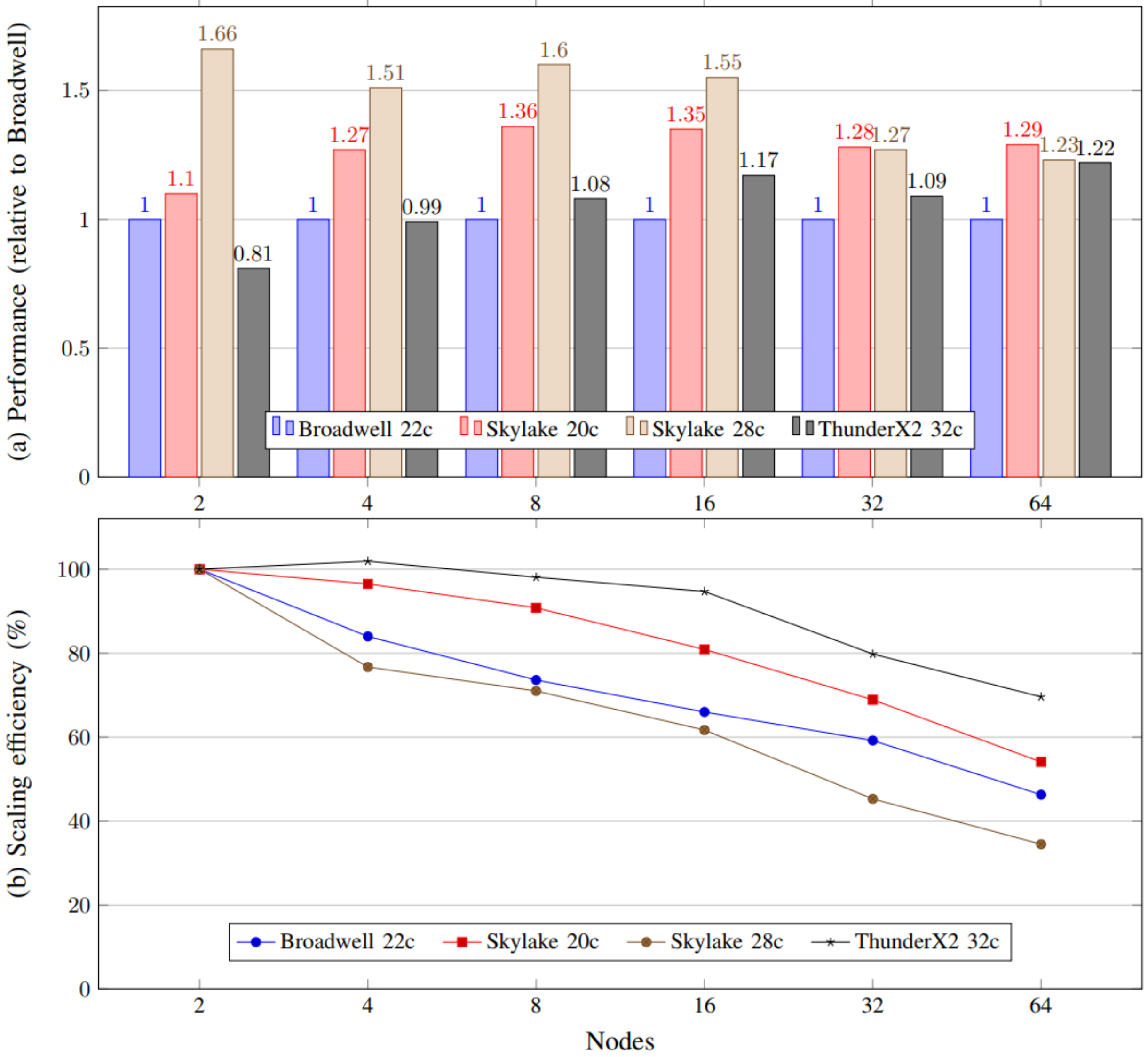
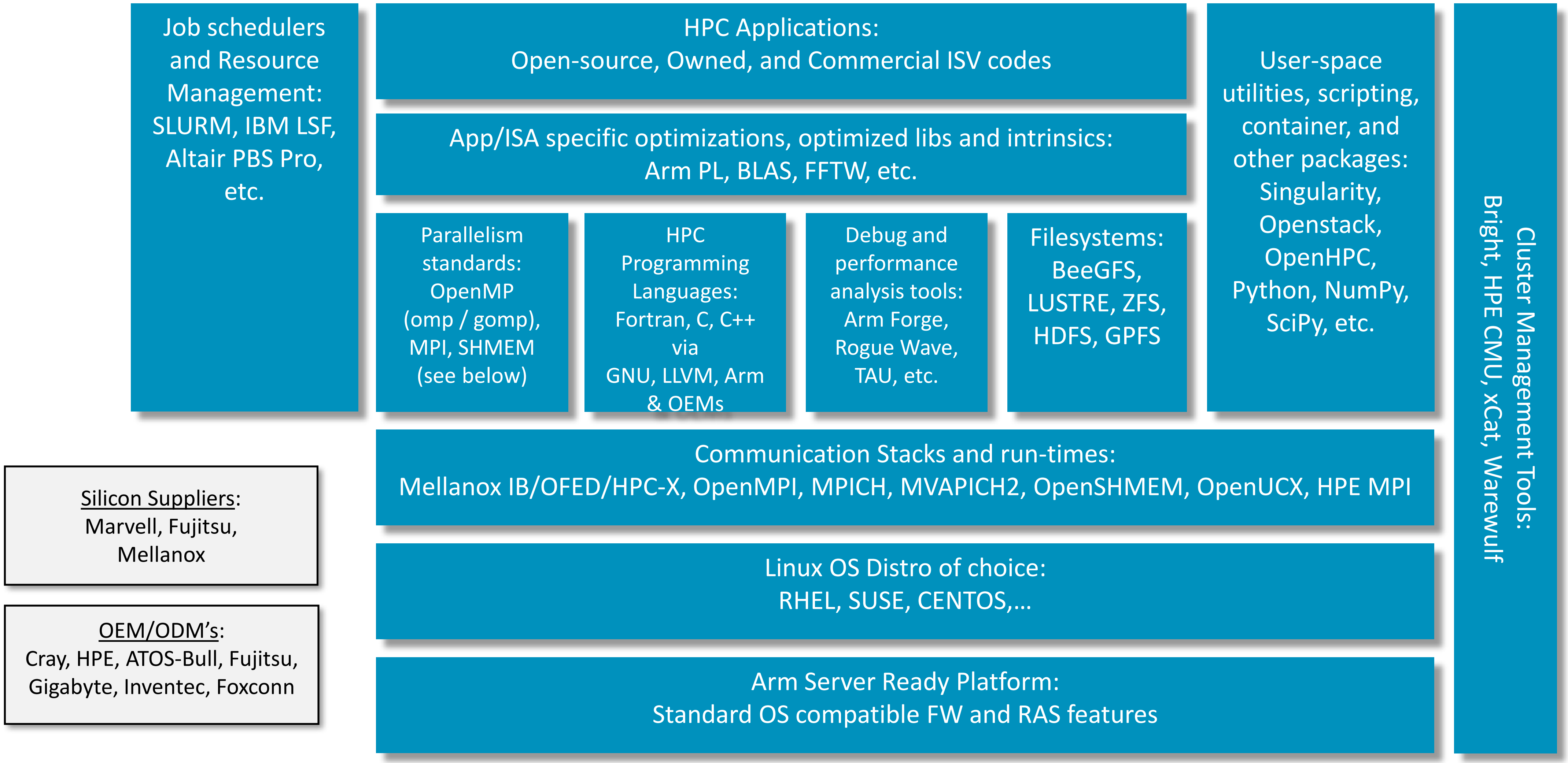


Figure 6. GROMACS scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



Ecosystem

Arm HPC Ecosystem – Overview



Job schedulers and Resource Management: SLURM, IBM LSF, Altair PBS Pro, etc.

HPC Applications: Open-source, Owned, and Commercial ISV codes

App/ISA specific optimizations, optimized libs and intrinsics: Arm PL, BLAS, FFTW, etc.

Parallelism standards: OpenMP (omp / gomp), MPI, SHMEM (see below)

HPC Programming Languages: Fortran, C, C++ via GNU, LLVM, Arm & OEMs

Debug and performance analysis tools: Arm Forge, Rogue Wave, TAU, etc.

Filesystems: BeeGFS, LUSTRE, ZFS, HDFS, GPFS

User-space utilities, scripting, container, and other packages: Singularity, Openstack, OpenHPC, Python, NumPy, SciPy, etc.

Cluster Management Tools: Bright, HPE CMU, xCat, Warewulf

Communication Stacks and run-times: Mellanox IB/OFED/HPC-X, OpenMPI, MPICH, MVAPICH2, OpenSHMEM, OpenUCX, HPE MPI

Linux OS Distro of choice: RHEL, SUSE, CENTOS,…

Arm Server Ready Platform: Standard OS compatible FW and RAS features

Silicon Suppliers: Marvell, Fujitsu, Mellanox

OEM/ODM’s: Cray, HPE, ATOS-Bull, Fujitsu, Gigabyte, Inventec, Foxconn

openHPC : Easy HPC stack deployment on Arm

OpenHPC is a community effort to provide a common, verified set of open source packages for HPC deployments

Arm and partners actively involved:

- Arm is a silver member of OpenHPC
- Linaro is on Technical Steering Committee
- Arm-based machines in the OpenHPC build infrastructure

Status: 1.3.6 release out now

- Packages built on Armv8-A for CentOS and SUSE

Functional Areas	Components include
Base OS	CentOS 7.5, SLES 12 SP3
Administrative Tools	Conman, Ganglia, Lmod, LosF, Nagios, pdsh, pdsh-mod-slurm, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, test-suite
Provisioning	Warewulf
Resource Mgmt.	SLURM, Munge
I/O Services	Lustre client (community version)
Numerical/Scientific Libraries	Boost, GSL, FFTW, Metis, PETSc, Trilinos, Hypre, SuperLU, SuperLU_Dist,Mumps, OpenBLAS, Scalapack, SLEPc, PLASMA, ptScotch
I/O Libraries	HDF5 (pHDF5), NetCDF (including C++ and Fortran interfaces), Adios
Compiler Families	GNU (gcc, g++, gfortran), LLVM
MPI Families	OpenMPI, MPICH
Development Tools	Autotools (autoconf, automake, libtool), Cmake, Valgrind,R, SciPy/NumPy, hwloc
Performance Tools	PAPI, IMB, pdtoolkit, TAU, Scalasca, Score-P, SIONLib

HPC Applications ported to Arm

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 HPC Ecosystem website: www.arm.com/hpc

Starting point for developers and end-users of Arm for HPC

Latest events, news, blogs, and collateral including whitepapers, webinars, and presentations

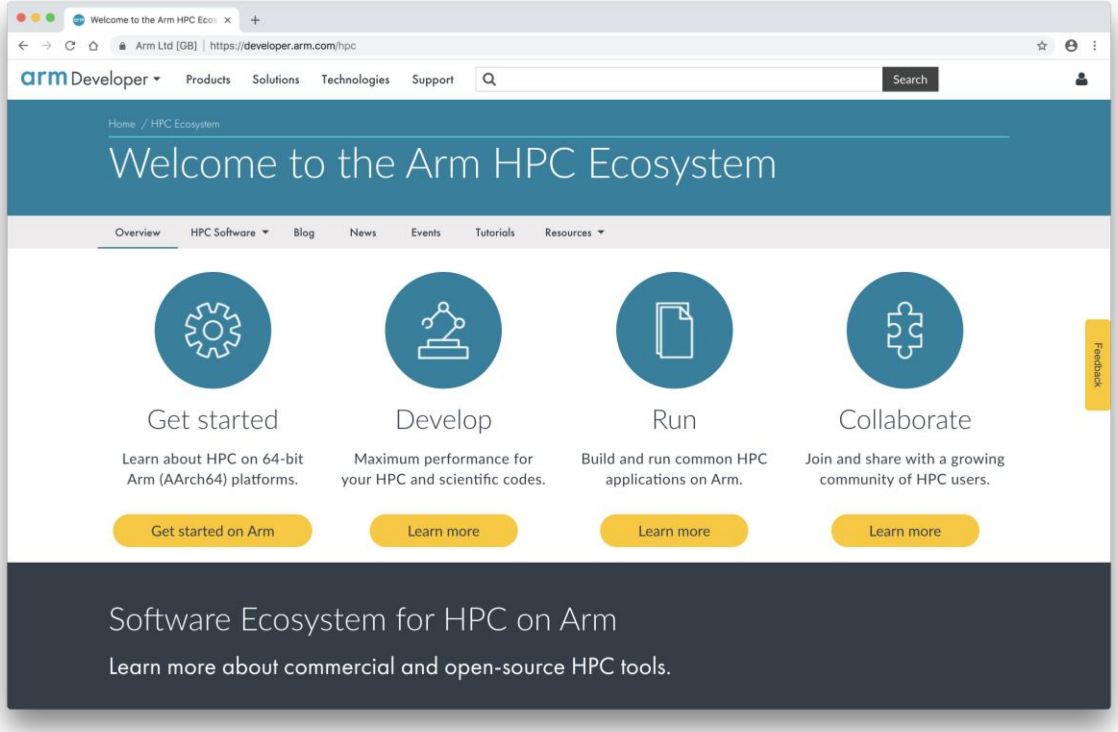
Links to HPC open-source & commercial SW packages

Guides for porting HPC applications

Quick-start guides to Arm tools

Links to community collaboration sites

Curated and moderated by Arm



Welcome to the Arm HPC Ecosystem

Get started

Learn about HPC on 64-bit Arm (AArch64) platforms.

Develop

Maximum performance for your HPC and scientific codes.

Run

Build and run common HPC applications on Arm.

Collaborate

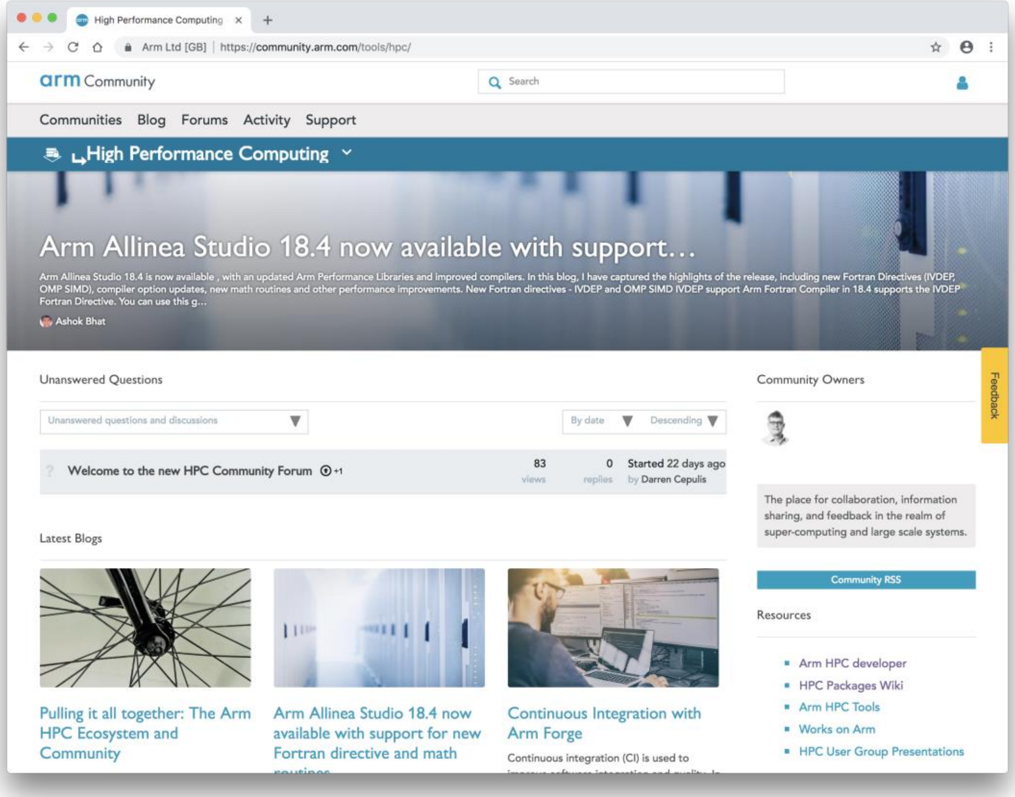
Join and share with a growing community of HPC users.

Software Ecosystem for HPC on Arm

Learn more about commercial and open-source HPC tools.

Arm HPC Community: community.arm.com/tools/hpc/

HPC Community-driven Content



High Performance Computing
Arm Ltd [GB] | https://community.arm.com/tools/hpc/
arm Community
Search
Communities Blog Forums Activity Support
High Performance Computing

Arm Allinea Studio 18.4 now available with support…

Arm Allinea Studio 18.4 is now available , with an updated Arm Performance Libraries and improved compilers. In this blog, I have captured the highlights of the release, including new Fortran Directives (IVDEP, OMP SIMD), compiler option updates, new math routines and other performance improvements. New Fortran directives - IVDEP and OMP SIMD IVDEP support Arm Fortran Compiler in 18.4 supports the IVDEP Fortran Directive. You can use this g…

Ashok Bhat

Unanswered Questions

Unanswered questions and discussions ▼ By date ▼ Descending ▼

Welcome to the new HPC Community Forum +1 | 83 views | 0 replies | Started 22 days ago by Darren Cepulis

Latest Blogs

Pulling it all together: The Arm HPC Ecosystem and Community

Arm Allinea Studio 18.4 now available with support for new Fortran directive and math routines

Continuous Integration with Arm Forge

Continuous integration (CI) is used to

Community Owners

The place for collaboration, information sharing, and feedback in the realm of super-computing and large scale systems.

Community RSS

Resources

- Arm HPC developer
- HPC Packages Wiki
- Arm HPC Tools
- Works on Arm
- HPC User Group Presentations

Blogs by Arm and our HPC community

Calendar of upcoming events such as workshops and webinars

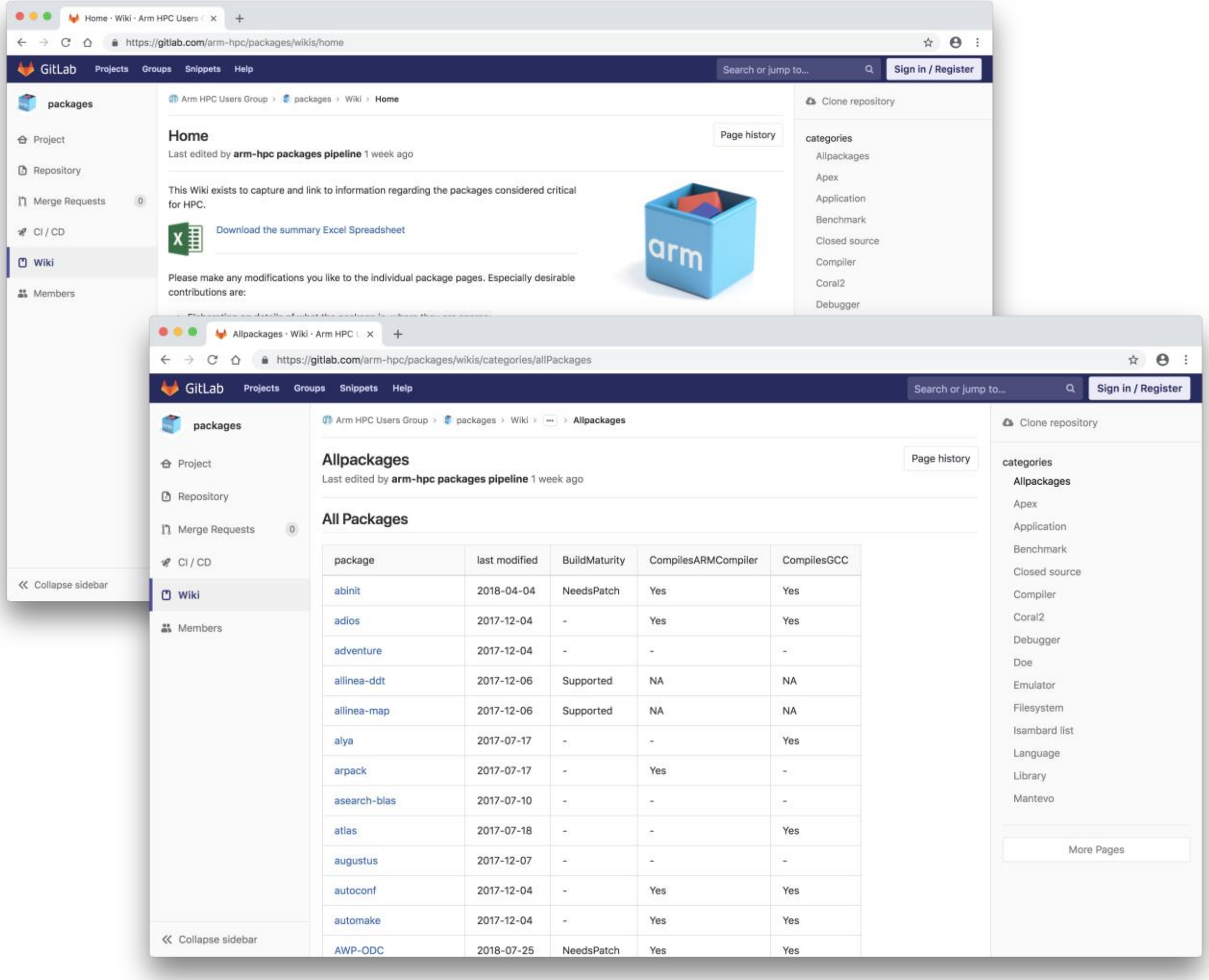
HPC Forum with questions & posts curated and moderated by Arm HPC technical specialists

Ask, answer, share progress and expertise

Arm HPC Packages wiki

www.gitlab.com/arm-hpc/packages/wikis

- Dynamic list of common HPC packages
- Status and porting recipes
- **Community** driven
- **Anyone can join** and contribute
- Provides **focus for porting** progress
- Allows developers to **share** and **learn**



Home

Last edited by **arm-hpc packages pipeline** 1 week ago

This Wiki exists to capture and link to information regarding the packages considered critical for HPC.

Download the summary Excel Spreadsheet

Please make any modifications you like to the individual package pages. Especially desirable contributions are:

Allpackages

Last edited by **arm-hpc packages pipeline** 1 week ago

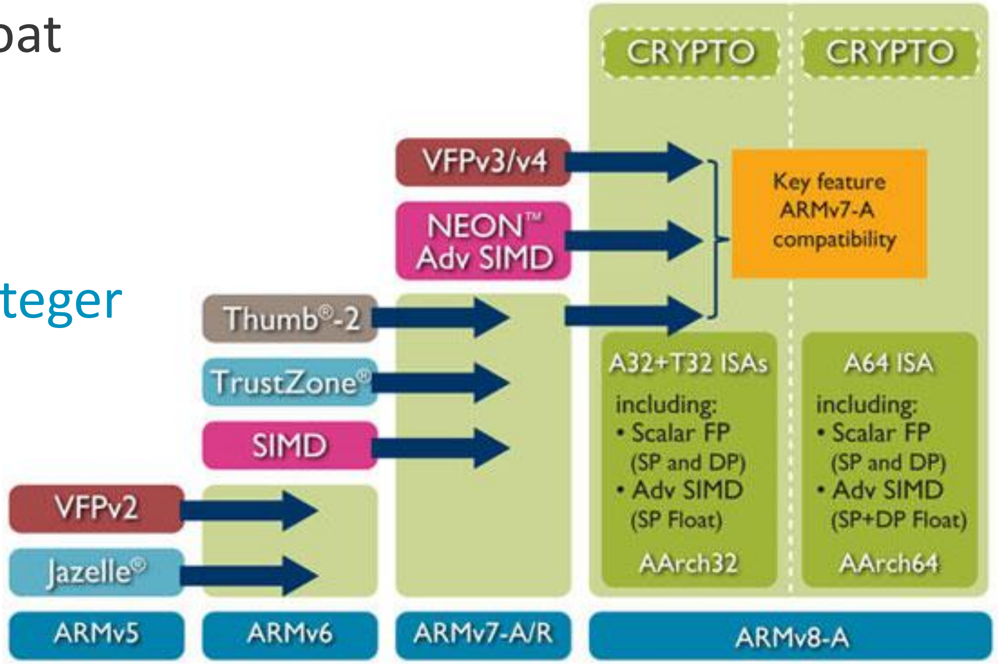
All Packages

package	last modified	BuildMaturity	CompilesARMCompiler	CompilesGCC
abinit	2018-04-04	NeedsPatch	Yes	Yes
adios	2017-12-04	-	Yes	Yes
adventure	2017-12-04	-	-	-
allinea-ddt	2017-12-06	Supported	NA	NA
allinea-map	2017-12-06	Supported	NA	NA
alya	2017-07-17	-	-	Yes
arpack	2017-07-17	-	Yes	-
asearch-blas	2017-07-10	-	-	-
atlas	2017-07-18	-	-	Yes
augustus	2017-12-07	-	-	-
autoconf	2017-12-04	-	Yes	Yes
automake	2017-12-04	-	Yes	Yes
AWP-ODC	2018-07-25	NeedsPatch	Yes	Yes

Looking Forward

Arm Architecture

- Armv7 Advanced SIMD (*aka* Arm NEON instructions) now 12 years old
  - Integer, fixed-point and non-IEEE single-precision float
  - 16 × 128-bit vector registers
- AArch64 Advanced SIMD was an evolution
  - Gained full IEEE double-precision float and 64-bit integer vector ops
  - 32 × 128-bit vector registers

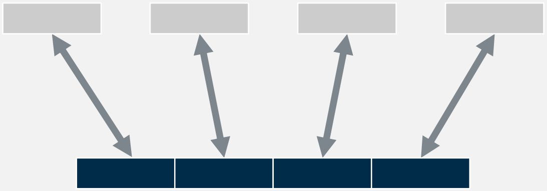

CRYPTO
CRYPTO
VFPv3/v4
NEON™ Adv SIMD
Key feature ARMv7-A compatibility
Thumb®-2
TrustZone®
SIMD
VFPv2
Jazelle®
A32+T32 ISAs including: • Scalar FP (SP and DP) • Adv SIMD (SP Float)
AArch32
A64 ISA including: • Scalar FP (SP and DP) • Adv SIMD (SP+DP Float)
AArch64
ARMv5
ARMv6
ARMv7-A/R
ARMv8-A

What about the Scalable Vector Extension?

- Updated vector extension after Advanced SIMD (aka Arm NEON) with features needed for new markets (e.g., gather load & scatter store, per-lane predication, longer vectors)

- **There is no preferred vector length**
  - The vector length (VL) is a hardware choice, 128-2048b, in increments of 128b
  - A Vector Length Agnostic (VLA) programming adjusts dynamically to the available VL
- **SVE is not an extension of Armv8 Advanced SIMD (*aka* Neon)**
  - SVE is a separate, optional extension with a new set of instruction encodings.
  - Initial focus is HPC and general-purpose server, not media/image processing.
- **SVE begins to tackle traditional barriers to auto-vectorization**
  - Very low overhead vs scalar code to encourage opportunistic vectorization.
  - Software-managed speculative vectorization of uncounted loops (i.e. C break).
  - Extract more data-level parallelism (DLP) from existing C/C++/Fortran source code

SVE is Arm’s next generation SIMD ISA



Gather-load and scatter-store

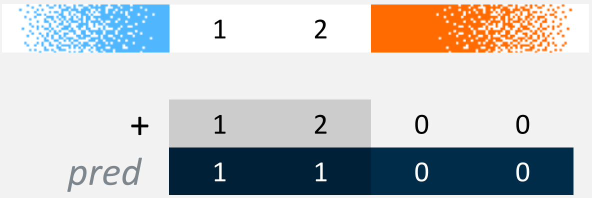
	1	2	3	4
+	5	5	5	5
pred	1	0	1	0
=	6	2	8	4

Per-lane predication

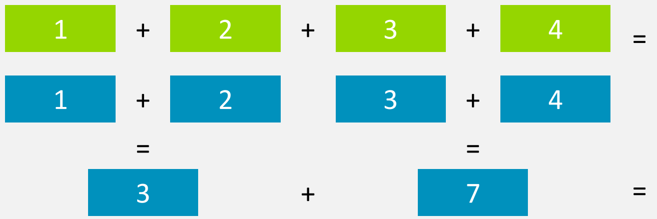
```
for (i = 0; i < n; ++i)
```

INDEX i	n-2	n-1	n	n+1
CMPLT n	1	1	0	0

Predicate-driven loop control and management



Vector partitioning and software-managed speculation



Extended floating-point horizontal reductions

Vectorisation for SVE

- SVE instructions can be generated by autovectorisation
  - With or without directives

- Directives / Pragmas to help with vectorisation:
  - IVDEP
  - VECTOR ALWAYS
  - OMP SIMD

- ACLE (Arm C Language Extensions) for SVE
  - C/C++ (Fortran by bindings) – #include <arm_sve.h>
  - SVE intrinsics and datatypes
  - https://developer.arm.com/docs/100987/latest/arm-c-language-extensions-for-sve

Open source support

- **Arm actively posting SVE open source patches upstream**
  - Beginning with first public announcement of SVE at HotChips 2016.

- **Available upstream**
  - GNU Binutils-2.28: released Feb 2017, includes SVE assembler & disassembler.
  - GCC 8: Full assembly, disassembly and basic auto-vectorization
  - LLVM 7: Full assembly, disassembly
  - QEMU 3: User space SVE emulation
  - GDB 8.2 HPC use cases fully included

- **Under upstream review**
  - LLVM: since Nov 2016, as presented at LLVM conference.
  - Linux kernel: since Mar 2017, LWN article on SVE support.

Compiler support for SVE

Feature	Upstream GCC	Upstream LLVM	Arm Compiler 6 (For bare metal)	Arm HPC Compiler (for Linux user-space)
SVE asm and disasm	Yes	Yes	Yes	Yes
SVE code generation	Yes	No Planned for 2018-19	Yes	Yes
SVE ACLE	No Planned for GCC9 (2019)	No Planned for 2018-19	Yes	Yes
Auto-vectorization	Basic More improvements planned for GCC9	None Planned for 2019-20	Advanced	Advanced

Arm Instruction Emulator

Develop your user-space applications for future hardware today

Develop software for
tomorrow’s hardware today

Runs at close to
native speed

Commercially Supported
by ARM

Start porting and tuning for future architectures early

- Reduce time to market, Save development and debug time with Arm support

Run 64-bit user-space Linux code that uses new hardware features on current Arm hardware

- SVE support available now. Support for 8.x planned.
- Tested with Arm Architecture Verification Suite (AVS)

Near native speed with commercial support

- Emulates only unsupported instructions
- Integrated with other commercial Arm tools including compiler and profiler
- Maintained and supported by Arm for a wide range of Arm-based SoCs

Arm Instruction Emulator

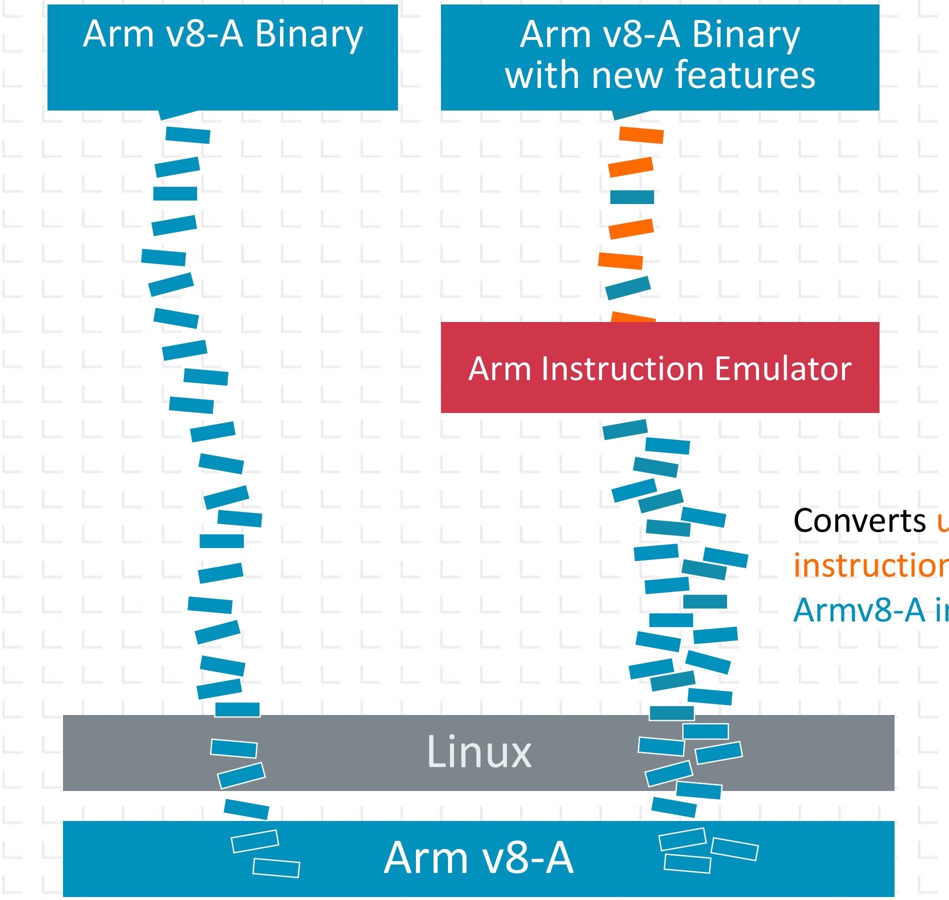
Develop your user-space applications for future hardware today

Run Linux user-space code that uses new hardware features (SVE) on current Arm hardware

-

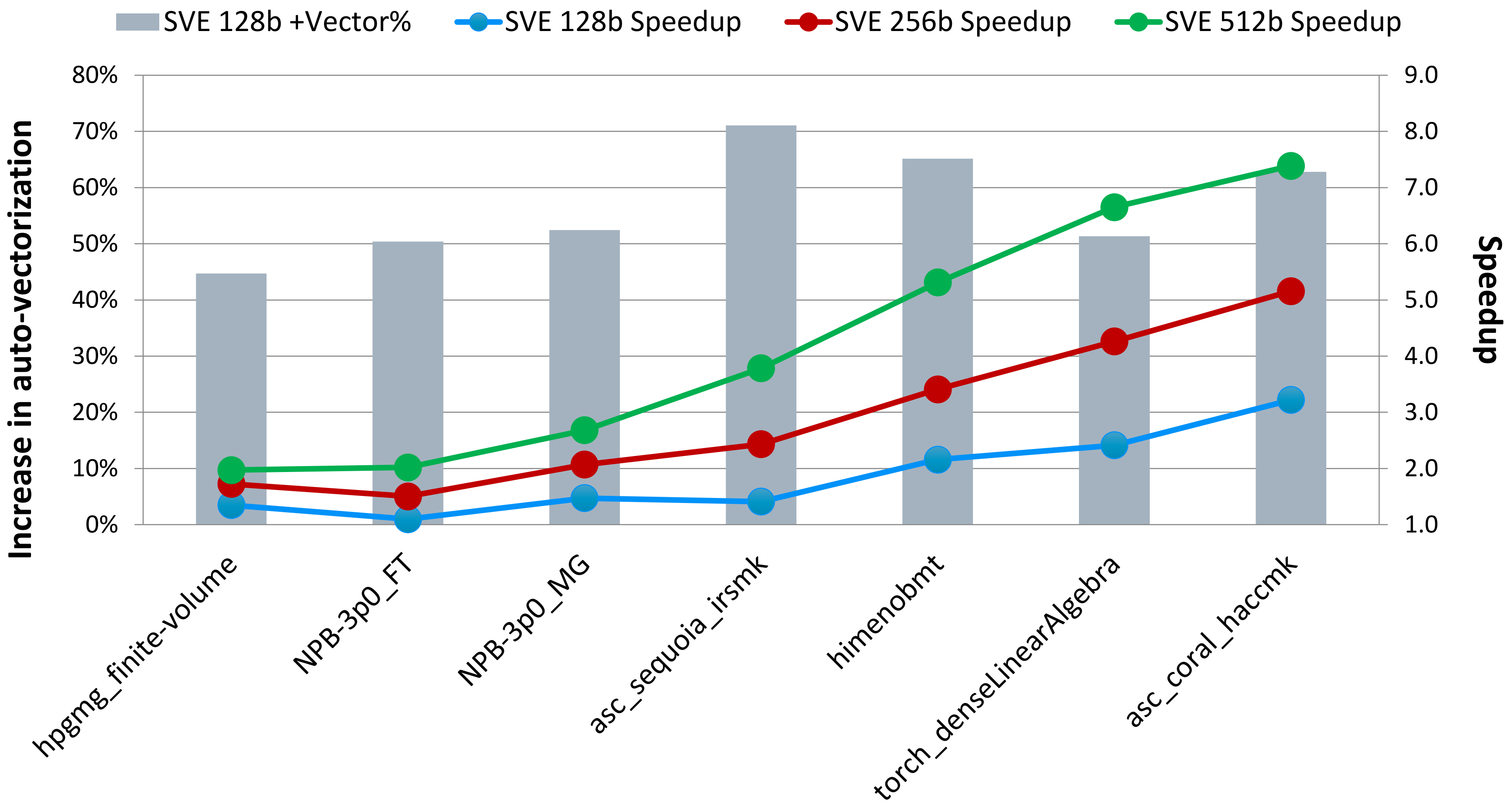
Simple “black box” command line tool

-
```
$ armclang hello.c --march=armv8+sve
$ ./a.out
Illegal instruction
$ armie -a=armv8+sve ./a.out
Hello
```



Converts unsupported instructions to native Armv8-A instructions

HPC benchmarks SVE vs. NEON



arm

Questions?