arm

HPC Software Stack and Tools

The 3rd Isambard hackathon - "Full Steam Ahead!"

Fabrice Dupros Fabrice.Dupros@arm.com

Conrad Hillairet Conrad.Hillairet@arm.com

Phil Ridley Phil.Ridley@arm.com

23rd March 2021

Agenda

- HPC Software Ecosystem
  - Overview
  - Community Building and Support

- Compilers, Libraries and Tools
  - Open-source and Commercial
    - Compilers
    - Maths Libraries
  - Profiling and Debugging

HPC Ecosystem

Applications
Open-source, owned, commercial ISV codes, …

Containers, Interpreters, etc.
Singularity, PodMan, Docker, Python, …

Performance Engineering
Arm Forge (DDT, MAP), Rogue Wave, HPC Toolkit, Scalasca, Vampir, TAU, …

Schedulers
SLURM, IBM LSF, Altair PBS Pro, …

Cluster Management
Bright, HPE CMU, xCat, Warewulf, …

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, GPFS, …

Silicon Suppliers
Marvell, Fujitsu, Mellanox, NVIDIA, …

OS
RHEL, SUSE, CentOS, Ubuntu, …

Arm Server Ready Platform
Standard firmware and RAS

Software Ecosystem – Open Source Enablement

Arm actively develops both open source and commercial tools

Open-source compilers

GCC and LLVM toolchains have active development work from Arm

- Our commercial compiler feeds into both LLVM and Flang
- Optimizations are targeted at:
  - Improving functionality
  - Increasing performance across Arm partner cores
  - Specific micro-architectural tuning work

Supporting libraries and tools

Packages such as OpenBLAS and FFTW have contributions from our silicon partners and OEMs

We work with many communities to get Arm platforms better supported, such as the various MPI implementations, and are on the OpenMP steering committee

Applications in the Community

wiki.arm-hpc.org

GitLab Projects Groups Snippets Help

Search or jump to... Sign in / Register

packages

- Project overview
- Repository
- Labels
- Merge Requests 0
- Requirements
- CI/CD
- Operations
- Analytics
- Wiki
- Members

Collapse sidebar

Arm HPC Users Group › packages › Wiki › Home

Last edited by jlinford 23 hours ago

Page history

Home

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

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

- Elaborating on details of what the package is, where they are sparse;
- Mentioning yourself if you are actively working on it or have some expertise that you'd like to share;
- Adding labels to the relevant pages if you (for example) know that something includes NEON optimisations or is known to compile on Arm (with either GCC or the Arm Compiler suite);
- Sharing instructions, gotchas, recipes, results and anything else that you think could help those who want to evaluate a particular package on Arm.

All Packages

A list of all packages can be found here here

Categories

allPackages, APEX, application, benchmark, Catalyst, closed-source, compiler, CORAL2, debugger, DoD, DOE, emulator, filesystem, Isambard-list, language, LANL-Crossroads, library, Mantevo, mini app, ML, mpi-runtime, network, open-source, OpenHPC, ORNL, profiler, RIKEN-list, sles-hpc-module, system-service, tool, Tri-lab CTS-2, visualisation

Most recently modified packages

package	last modified	BuildMaturity	CompilesARMCompiler	CompilesGCC	EasyBuild	SpackSupport
scalapack	2021-03-18	Upstream	Yes	Yes	-	-
nemo	2021-03-15	NeedsPatch	Yes	Yes	-	-
XIOS	2021-03-14	NeedsPatch	Yes	Yes	-	-
netcdf	2021-03-14	NeedsPatch	Yes	Yes	-	-
hdf5	2021-03-14	NeedsPatch	Yes	Yes	-	-

Clone repository

categories

- allPackages
- apex
- application
- benchmark
- catalyst
- closed source
- compiler
- coral2
- debugger
- dod
- doe
- emulator
- filesystem
- isambard list
- language

View All Pages

Community Groups

https://arm-hpc.groups.io/g/ahug

Arm HPC User Group (AHUG) ahug@arm-hpc.groups.io

The Arm HPC User Group collaboration and info sharing site for end-users and ecosystem partners. Please do NOT post any restricted or confidential information on this site.

This site offerings the following:

- A focal point for HPC leadership to guide and influence the direction of the collaboration in support of the Arm HPC ecosystem and its user group.
- Curated/moderated main and sub-groups aligned with Arm HPC ecosystem initiatives, deployments, projects.
- Email reflectors (arm-hpc@groups.io) reaching all members as well as separate email aliases for each sub-group.
- Topic discussions for the main AHUG arm-hpc group and each sub-group.
- User directories and points of contact.

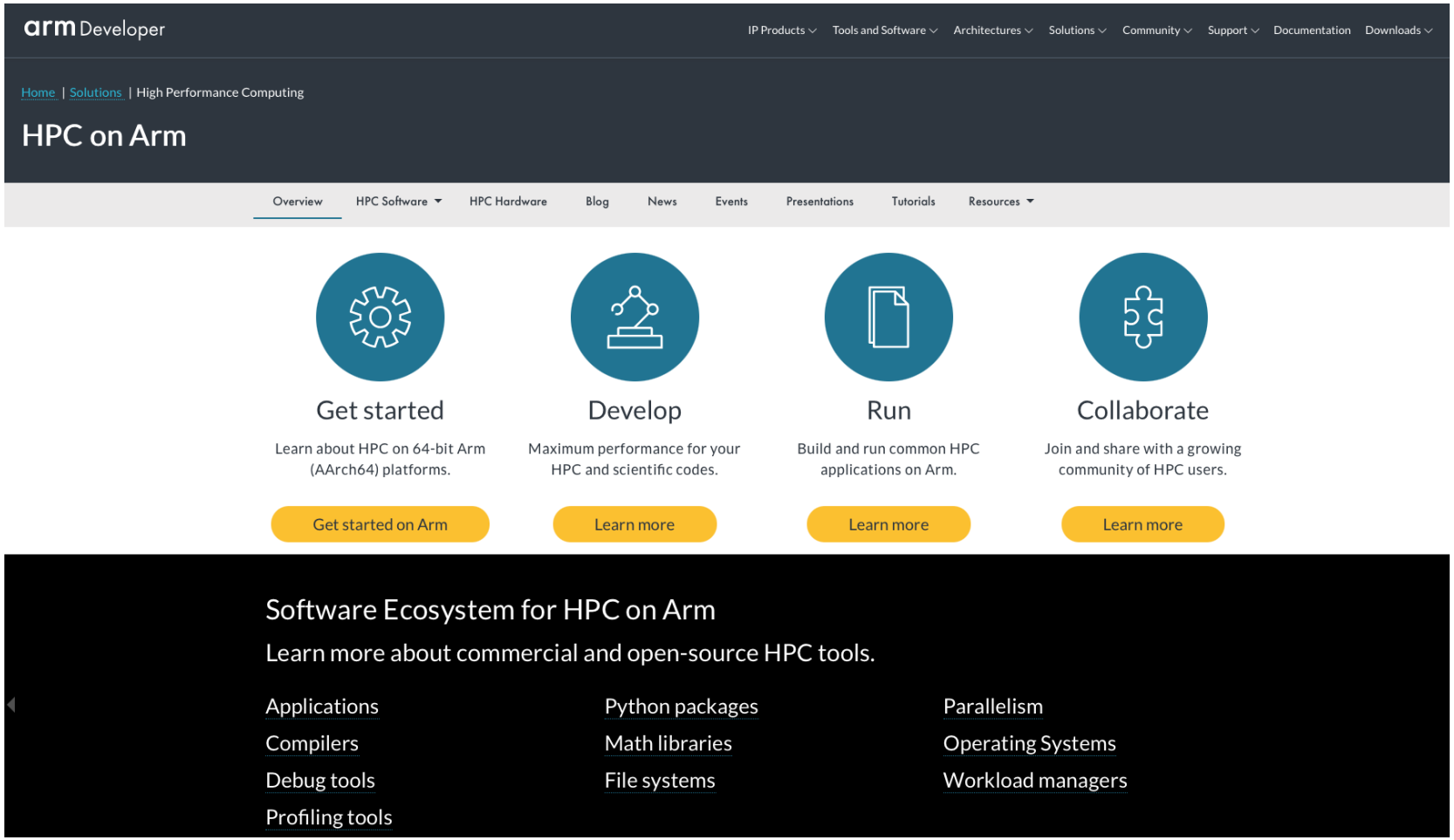
This site is NOT intended to duplicate or replace existing Arm HPC content portals and resources such as:

- https://developer.arm.com/hpc (Arm's hosted HPC ecosystem portal covering ecosystem news, blogs, presentations, training, software, tools, etc)
- https://gitlab.com/arm-hpc/packages/wikis/home (Arm HPC SW Packages Wiki)
- https://community.arm.com/b/hpc (Arm Community HPC Blog landing page)

Discussion in this forum is **vastly preferred**, in order to reach the widest range of recipients and promote the most vibrant discussion possible.
AHUG Topics of discussion: https://arm-hpc.groups.io/g/ahug/topics

Arm Developer

https://developer.arm.com/solutions/hpc



Events and Hackathons

https://gitlab.com/arm-hpc/training/arm-sve-tools

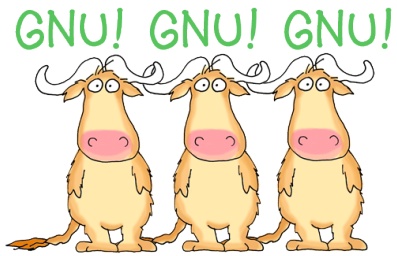
Join us for another Arm Scalable Vector Extension Hackathon! This hands-on, non-NDA event will introduce vector length agnostic programming and jumpstart developers targeting the the first CPU to implement SVE, the Fujitsu A64FX. Hands-on exercises will introduce SVE compilers, libraries, and tools from Fujitsu, Cray, Arm, and GNU, and show how popular HPC codes can take advantage of SVE. Bring your own code or use our prepared hands-on exercises!

This event is generously supported by Fujitsu and the University of Bristol. Fujitsu will provide remote access to a Fujitsu PRIMEHPC FX700 system. The University of Bristol will provide access to the HPE Apollo 80 partition of Isambard 2, the largest Arm-based supercomputer in Europe. Don't miss out! Remote access details will be provided to all registered attendees.

Compilers, Libraries and Tools

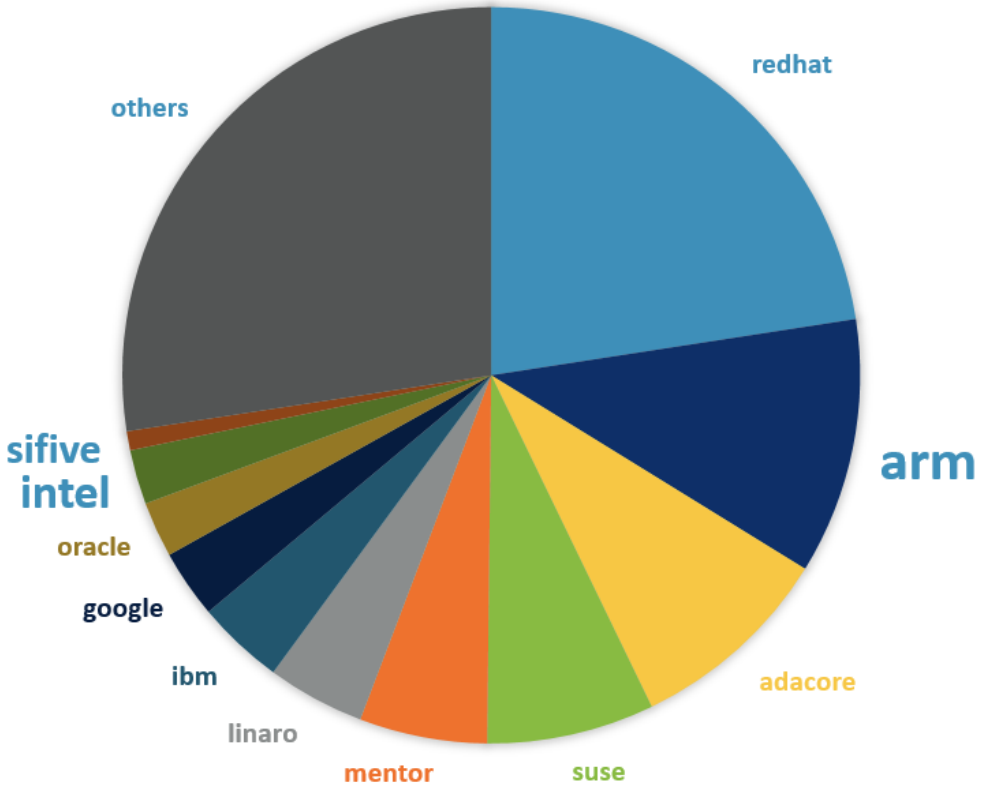
GNU compilers are a solid option

With Arm being significant contributor to upstream GNU projects



- GNU compilers are first class Arm compilers
  - Arm is one of the largest contributors to GCC
  - Focus on enablement and performance
  - Key for Arm to succeed in Cloud/Data center segment

- GNU toolchain ships with Arm Allinea Studio
  - Best effort support
  - Bug fixes and performance improvements in upcoming GNU releases



GCC

GCC 10.2

- GCC 10.2
  - Scalable Vector Extension (SVE) ACLE types and intrinsics now supported(arm_sve.h)
  - Improved vectorizer for SVE (gather loads, scatter stores and cost model)
  - ACLE intrinsics enables support for
    - The Transactional Memory Extension
    - The Matrix Multiply extension
    - Armv8.6-A features, e.g. the bfloat16 extension -march=armv8.6-a
    - SVE2 with -march=armv8.5-a+sve2
- GCC 11.0
  - Support for the Fujitsu A64FX -mcpu=a64fx

Open Source Maths Libraries

https://developer.arm.com/solutions/hpc/hpc-software/categories/math-libraries

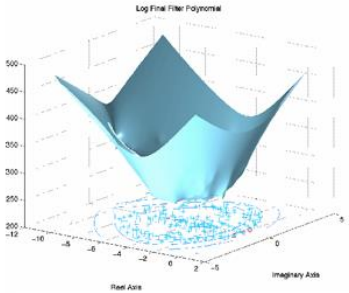
NSIMD

GNU Scientific Library

FEAST

El

Elemental



ARPACK

sundials

Eigen

hypre high performance preconditioners

Trilinos

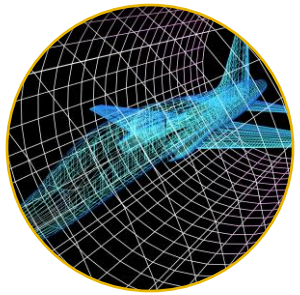
SLEEF

SIMD Library for Evaluating Elementary Functions

Scalable Library for Eigenvalue Problem Computations SLEPc

And OpenBLAS, FFTW, BLIS, PETSc, ATLAS, PBLAS, ScaLAPACK

arm ALLINEA STUDIO

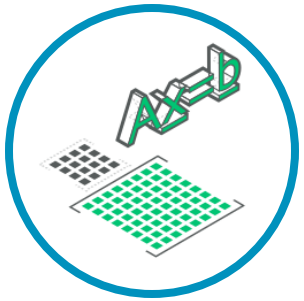


Fortran Compiler

- Fortran 2003 support
- Partial Fortran 2008 support
- OpenMP 3.1
- Directives to support explicit vectorization control
- SVE

C/C++ Compiler

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



Performance Libraries

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

Forge

- 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 server-class Arm-based platforms

Server & HPC Development Solutions from Arm

Best in class commercially supported tools for Linux and high-performance computing

Code Generation

for Arm servers

arm COMPILER FOR LINUX

- arm C/C++ Compiler
- arm Fortran Compiler
- arm Performance Libraries

+

Performance Engineering

*for **any architecture**, at **any scale***

arm FORGE

arm DDT	*Debugger*
arm MAP	*Profiler*
arm PERFORMANCE REPORTS	*Reporting*

=

Server & HPC Solution

for Arm servers

arm ALLINEA STUDIO

Commercially Supported Toolkit for applications development on Linux

- C/C++ Compiler for Linux
- Fortran Compiler for Linux
- Performance Libraries
- Performance Reports
- Debugger
- Profiler

Arm Compiler for Linux (ACfL): Front-end

Clang and Flang

C/C++

- Clang front-end
  - C11 including GNU11 extensions and C++14
  - Arm's 10-year roadmap for Clang is routinely reviewed and updated to respond to customers
- C11 with GNU11 extensions and C++14
- **Auto-vectorization for SVE and Neon**
- OpenMP 4.5

Fortran

- Flang front-end
  - Extended to support gfortran flags
- Fortran 2003 with some 2008
- **Auto-vectorization for SVE and Neon**
- OpenMP 3.1
- Transition to flang "F18" in progress
  - Extensible front-end written in C++17
  - Complete Fortran 2008 support
  - OpenMP 4.5 support

arm PERFORMANCE LIBRARIES

Best-in-class performance

Commercially Supported by Arm

Validated with NAG test suite

- Commercial 64-bit ArmV8-A math Libraries
  - Commonly used low-level maths routines – BLAS, LAPACK and FFT
  - Optimised maths intrinsics
  - Validated with NAG's test suite, a de facto standard
- Best-in-class performance with commercial support
  - Tuned by Arm for specific cores – like Arm Neoverse-N1
  - Maintained and supported by Arm for wide range of Arm-based SoCs
- Silicon partners can provide tuned micro kernels for their SoCs
  - Partners can contribute directly through open source routes
  - Parallel tuning within our library increases overall application performance

ACfL + Arm PLs 21.0

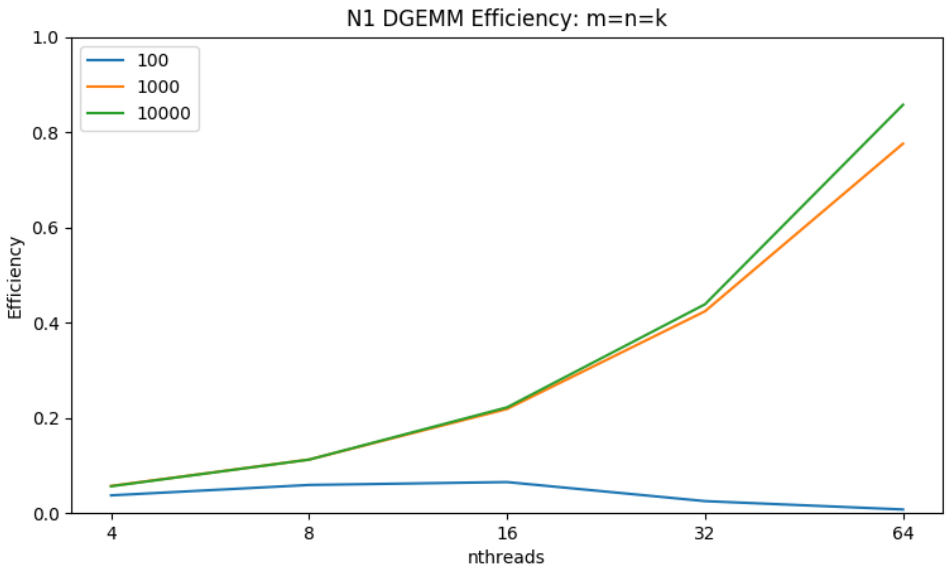
Latest Release

- Compilers
  - LLVM 11
  - SVE improvements
- Libraries
  - Just two varieties provided
    - Neon and SVE, with all uArch performance handled internally
  - New features
    - Real-to-real DFT
    - GEMMT
    - Batched BLAS/LAPACK interfaces
    - Performance improvements, including for small matrix sizes
- Packaging
  - New and improved modules
  - GCC 10.2.0

Optimised Maths Libraries (Arm Performance Libraries)

- Arm produce a set of accelerated maths routines
  - Microarchitecture tuned for each Arm core
  - BLAS, LAPACK, FFT (Standard interface)
  - Tuned math calls
    - Transcendentals (libm) + string functions
  - Sparse operations
    - SpMV / SpMM
  - Available for GCC and Arm compiler

- Other vendor maths libraries also available
  - HPE/Cray (LibSci), Fujitsu (SSL2), NVIDIA HPC SDK math libraries



DGEMM Performance on the Neoverse N1 for different matrix sizes

Max efficiency 85.7%

Compile and link your application with ACfL

Building your application

- Modify the Makefile/installation scripts to ensure compilation for aarch64 happens
- Compile the code with **ACfL**
- Link the code with the **Arm Performance Libraries (Arm PLs)**

- Examples:
- $> armclang -mcpu=native –Ofast -c –I/path/armpl/include example.c -o example.o
- $> armclang example.o -armpl -o example.exe -lm

Arm Compiler for Linux	GNU Compiler
armclang	gcc
armclang++	g++
armflang	gfortran

Linking your application with the Arm PLs

Building your application

- Modify the Makefile/installation scripts to ensure compilation for aarch64 happens
- Compile the code with **GCC**
- **Load the correct module for the Arm Performance Libraries (Arm PLs)**
- Link the code with the **Arm PLs**

- Example:
- $> module load tools/arm-compiler-a64fx/gcc-10.2.0
- $> module use /software/aarch64/tools/arm-compiler/21.0/modulefiles
- $> module load armpl-AArch64/21.0.0 or module load armpl-AArch64-SVE/21.0.0
- $> gcc -c –I$ARMPL_DIR/include example.c -o example.o
- $> gcc example.o -L$ARMPL_DIR/lib -larmpl_mp -L$ARMPL_DIR/lib -lamath -o example.exe -lm

A64FX Features and Fujitsu Compiler

- A64FX CPU Inherits features of K computer and PRIMEHPC FX100
  - Usability including options and programming models are inherited
- Compiler targeting 512-bit wide vectorization to promotes optimization, such as constant folding, by fixing vector length
  - Vectorization as VLA(vector-length-agnostic) and Neon (Advanced SIMD) is also supported

	Functions & Architecture	Fugaku	FX100	K computer
Processor	Base ISA + SIMD Extensions	ARMv8-A+SVE	SPARC V9 +HPAC-ACE2	SPARC V9 +HPC-ACE
	SIMD width [bits]	512	256	128
	Float Packed SIMD	✔ Enhanced	✔	-
	FMA	✔	✔	✔
	Reciprocal approx. inst. Math. acceleration inst.	✔	✔	✔
	Inter-core hardware barrier	✔	✔	✔
	Sector cache	✔ Enhanced	✔	✔
	Hardware "prefetch" assist	✔ Enhanced	✔	✔

Fujitsu Compiler: Language Standard Support

Languages	Specification	Support Level
C	C11 (ISO/IEC 9899:2011)	fully supported
C++	C++14 (ISO/IEC 14882:2014) C++17C++17 (ISO/IEC 14882:2017)	fully supported partially supported
Fortran	Fortran 2008 (ISO/IEC 1539-1:2010) Fortran 2018 (ISO/IEC 1539-1:2018)	fully supported partially supported
OpenMP	OpenMP 4.0 (released in July 2013) OpenMP 4.5 (released in Nov. 2015) OpenMP 5.0 (released in Nov. 2018)	fully supported partially supported partially supported

Promotes object-oriented programming and accelerates high performance by supporting latest language standards

Fujitsu Compiler Commands

- Cross-compiler, which works on x86 server

Tips: cross compiler commands have **px** suffix, which means cross-platform.

Language	Command
Fortran	`frtpx [option list] [file list]`
C	`fccpx [options list] [file list]`
C++	`FCCpx [options list] [file list]`

- Own-compiler, which works on Arm server

Language	Command
Fortran	`frt [option list] [file list]`
C	`fcc [options list] [file list]`
C++	`FCC [options list] [file list]`

Note: own-compiler is also called native-compiler or self-compiler.

Cross-compiler and own-compiler have the same ability.
Differences are their command names and where they work.

Recommended Options

- -Kfast option is recommend for higher performance
  - Turns on the following options internally
  - Some Options cause side-effects in execution result such as precision

Option	Feature
-O3	Compile at highest optimization level 3.
-Kdalign	Assume alignment on a double-word boundary.
-Keval	Apply optimization to change the method of mathematical evaluation
-Kfp_contract	Optimize by using FMA arithmetic instructions.
-Kfp_relaxed	Execute reciprocal approximation operations.
-Kfz	**[New for Armv8]** Enable flush-to-zero mode to treat denormal numbers as zero
-Kilfunc	Inline expand intrinsic math functions into approximate instructions
-Kmfunc	Apply multi-operation functionalization to promote vectorizing
-Komitfp	Omit the frame pointer register for a procedure call.
-Ksimd_packed_promotion	**[New for SVE]** Promote packed simd for SVE instructions to optimize address calculation
-Klib	[C/C++ only] Optimize with recognizing C standard libraries functions as built-in
-Krdconv	[C/C++ only] Optimize with assuming that 4-byte int loop variants does not overflow
-x-	[C/C++ only] Inline expand user-defined functions

Other Notable Options for Performance

- Options to promote or control optimizations

Option	Feature
-Kpreex	Evaluate codes which is invariant through loops before entering loops. This may cause a execution time error such as segmentation fault.
-Ksimd=2	Vectorizes loops which contains IF-constructs with predication.
-Kocl	Enable Optimization Control Lines (OCL), FUJITSU-specific directives to control optimization.

- Parallelization options

Option	Feature
-Kparallel	Apply automatic parallelization.
-Kopenmp	Enable OpenMP directives
-Kopenmp_simd	Enable OpenMP simd directives

under development

- Useful options to know applied optimizations

Option	Feature
-Nlst=t	Output Compilation Optimization information in list file (*.lst)
-Koptmsg=2	Output optimization messages

A lot of flags to play with

Flag	Description
-Ndang / -Nnodang	
-KA64FX / -KGENERIC_CPU	
-Kassume=notime_saving_compilation	Priority on optimization for application rather than compilation time
-SSL2 / -SSL2BLAMP	Link against optimized (serial or threaded) math and BLAS/LAPACK routines
-Kassume=noshortloop	Assumes iteration count of innermost loop is small
-Kassume=(no)memory_bandwidth	Innermost loop assumed memory bandwidth bounded and will impact SIMD, unrolling and ZFILL generation
-Kloop_fission / -Kloop_nofission	
-KSVE (default) / -KNOSVE	
-Kunroll / -Knounroll	
-Ksimd_reg_size={128,256,512,agnostic}	Agnostic to encourage VLA programming
-Koptmsg=guide	
-Kswp / -Knoswp / -Kswp_weak	Enable / Disable software pipelining Relaxing software pipelining : when memory bandwidth bound : less benefits of using software pipelining
-Nlibomp / -Nfjomplib	LLVM OpenMP library vs Fujitsu OpenMP library
-Kzfill	Using DC ZVA instructions

OCL

Stands for Optimization Control Line

Directive	Description
!OCL ASSUME ((NO)SHORTLOOP | (NO)MEMORY_BANDWIDTH | (NO)TIME_SAVING_COMPILATION)	Controls optimization based on the loop caracteristics
!OCL CACHE_SECTOR_SIZE(l2_n1, l2_n2) … !OCL END_CACHE_SECTOR_SIZE	Maximum number of ways for sector 0 and sector 1 of cache L2 Data reused in sector 1
!OCL CACHE_SUBSECTOR_ASSIGN(array1[,array2...]) … !OCL END_CACHE_SUBSECTOR	Assigns an array to sector 1 of cache.
!OCL FISSION_POINT[(n)] Also available : !OCL LOOP_(NO)FISSION	Ask to consider fission of the loop
!OCL ITERATIONS(max=n1) also min and avg	Ask to optimize considering max, avg or min iteration count
!OCL (NO)SIMD	Allow or not SIMD instructions
!OCL (NO)SWP	Allow or not software pipelining
!OCL (NO)UNROLL[(n/'full')]	Loop unrolling
!OCL (NO)UNROLL_AND_JAM(_FORCE)[(n)]	Performs unroll and jam
!OCL (NO)ZFILL	Allows or not ZFILL optimization

Runtime Options

Memory Cache Usage and Paging

```
export FLIB_SCCR_CNTL=TRUE
export FLIB_HPCFUNC=TRUE
export FLIB_HPCFUNC_INFO=TRUE
export FLIB_FASTOMP=TRUE
export FLIB_BARRIER=HARD
export XOS_MMM_L_HPAGE_TYPE=hugetlbfs
export XOS_MMM_L_PAGING_POLICY=demand:demand:demand
export XOS_MMM_L_PRINT_ENV=on
```

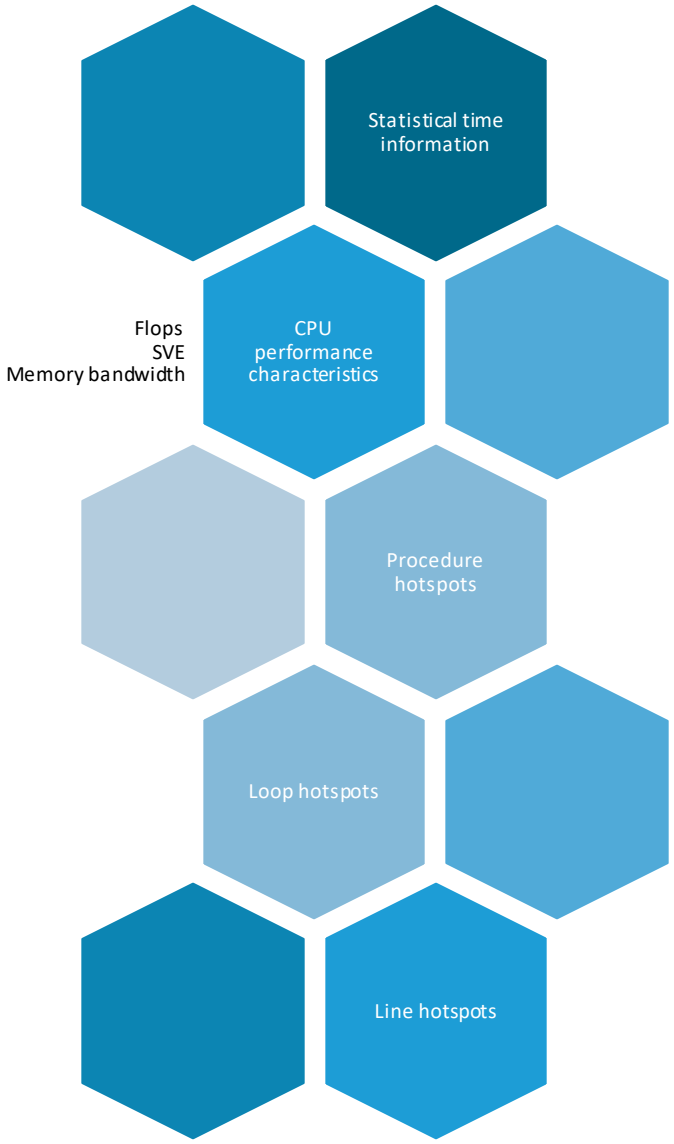
Use sector cache

Large memory pages

Paging policy for NUMA

fipp & fapp

Fujitsu Instant Performance Profiler




```
*****************************************************************************
    Process      35 - loops
*****************************************************************************

          Cost          %  Operation (s)        Barrier          %    Nest    Kind     Exec    Start      End
-----------------------------------------------------------------------------------------------------------------
           608   100.0000        60.8038             --         --      --      --       --       --       --   Process      35
-----------------------------------------------------------------------------------------------------------------
           360    59.2105        36.0023             --         --       3   DO      SERIAL      880     1108   rhs4th3fortsgstr_
            96    15.7895         9.6006             --         --       4   DO      SERIAL       88      135   addsgd4_
            25     4.1118         2.5002             --         --       3   DO      SERIAL     1139     1454   rhs4th3fortsgstr_
            14     2.3026         1.4001             --         --       4   DO      SERIAL     1190     1209   rhs4th3fortsgstr_
            13     2.1382         1.3001             --         --       4   DO      SERIAL     1930     1933   corrfort_
            13     2.1382         1.3001             --         --       4   DO      SERIAL     1900     1903   predfort_
            11     1.8092         1.1001             --         --       4   DO      SERIAL     1958     1961   dpdmtfort_
            11     1.8092         1.1001             --         --       1   FOR     SERIAL      427      428   Sarray::set_to_zero()
             9     1.4803         0.9001             --         --       3   DO      SERIAL       86      136   addsgd4_
             5     0.8224         0.5000             --         --       4   DO      SERIAL     1374     1379   rhs4th3fortsgstr_

           MPI          %  Communication (s)    Nest    Kind     Exec    Start      End
-----------------------------------------------------------------------------------------
             1     0.1645             0.1000      --      --       --       --       --   Process      35
-----------------------------------------------------------------------------------------

Lines profile (Total thread cost basis)

*****************************************************************************
    Application - lines
      Application and Process outputs the total value of the cost of each thread.
      Procedure outputs the total value of the line cost of each thread.
*****************************************************************************

          Cost          %  Operation (s)        Barrier          %     Line
-----------------------------------------------------------------------------
         29067   100.0000      2906.8723             --         --       --   Application
-----------------------------------------------------------------------------
          4508    15.5090       450.8268             --         --       89   addsgd4_
          4390    15.1030       439.0266             --         --     1022   rhs4th3fortsgstr_
          3607    12.4093       360.7214             --         --      980   rhs4th3fortsgstr_
          2506     8.6215       250.6144             --         --     1063   rhs4th3fortsgstr_
          1809     6.2236       180.9113             --         --      954   rhs4th3fortsgstr_
          1650     5.6765       165.0096             --         --      910   rhs4th3fortsgstr_
          1355     4.6616       135.5079             --         --      932   rhs4th3fortsgstr_
           711     2.4461        71.1041             --         --      428   Sarray::set_to_zero()
           617     2.1227        61.7037             --         --     1931   corrfort_
           563     1.9369        56.3033             --         --     1901   predfort_

           MPI          %  Communication (s)     Line
-------------------------------------------------------
            27     0.0929             2.7001       --   Application
-------------------------------------------------------
            10     0.0344             1.0000       44   main
            16     0.0550             1.6000       70   main
             1     0.0034             0.1000       75   main

*****************************************************************************
    Process      14 - lines
*****************************************************************************
```

fipp & fapp

Fujitsu Advanced Performance Profiler & CPU Performance Analysis Report

Process no.	0
CMG no.	1
Measured region	all, 0

Vector length (bit)	512
CPU frequency (GHz)	2.184

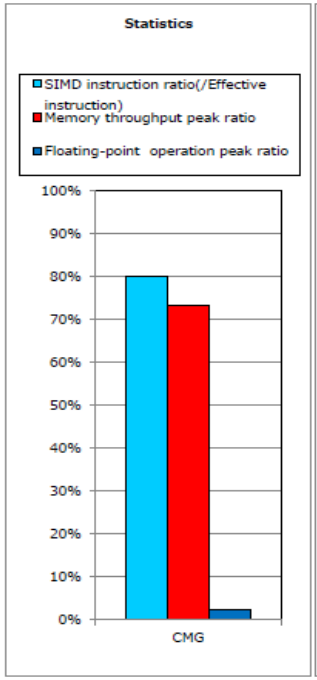
ry put :)	Memory throughput peak ratio (%)	Effective instruction	Floating-point operation	SIMD instruction rate (%) (/Effective instruction)	SVE operation rate (%)	Floating-point pipeline Active element rate (%)	IPC	GIPS
0.60	73.46%	1.54E+09	2.99E+09	74.31%	100.00%	67.72%	0.22	0.48
6.34		1.24E+09	2.67E+09	80.80%	100.00%	66.74%	0.34	0.75
6.34		1.24E+09	2.67E+09	80.81%	100.00%	66.78%	0.34	0.75
6.40		1.24E+09	2.67E+09	80.80%	100.00%	66.77%	0.34	0.76
6.53		1.24E+09	2.67E+09	80.80%	100.00%	66.74%	0.34	0.75
6.75		1.24E+09	2.67E+09	80.81%	100.00%	66.79%	0.34	0.75
6.70		1.24E+09	2.67E+09	80.80%	100.00%	66.79%	0.34	0.75
6.90		1.24E+09	2.67E+09	80.79%	100.00%	66.74%	0.34	0.75
6.58		1.24E+09	2.67E+09	80.79%	100.00%	66.67%	0.34	0.75
6.68		1.24E+09	2.67E+09	80.80%	100.00%	66.78%	0.34	0.75
6.76		1.24E+09	2.67E+09	80.81%	100.00%	66.74%	0.34	0.75
6.34		1.24E+09	2.67E+09	80.80%	100.00%	66.77%	0.34	0.76
8.05	73.46%	1.52E+10	3.23E+10	80.14%	100.00%	66.86%	0.33	8.52

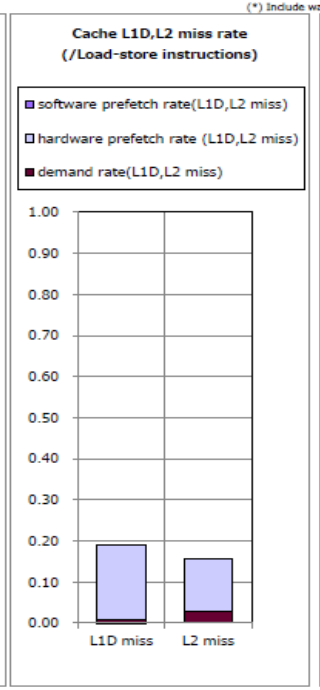
Cycle Accounting		Prefetch port busy wait		Memory access wait & Cache access wait						Operation wait		Other wait						Other instruction commit				
Process	Thread	Prefetch port busy wait by hardware prefetch	Prefetch port busy wait by software prefetch	Integer load memory access wait	Floating-point load memory access wait	Integer load L2 cache access wait	Integer load L1D cache access wait	Floating-point load L2 cache access wait	Floating-point load L1D cache access wait (*)	Integer operation wait	Floating-point operation wait	Branch instruction wait	Other wait	Store port busy wait	Instruction fetch wait	Barrier synchronization wait	1 instruction commit	2 instruction commit	3 instruction commit	4 instruction commit	Other instruction commit	Total
0	0	1.59E-01	0.00E+00	1.21E-02	7.04E-01	6.81E-03	2.58E-02	2.50E-01	1.74E-01	1.38E-03	1.39E+00	1.51E-03	1.60E-03	0.00E+00	7.04E-02	2.12E-01	2.57E-02	3.72E-02	2.19E-02	1.33E-01	0.00E+00	3.23E+0
0	1	1.11E-01	0.00E+00	9.86E-03	7.59E-01	3.62E-03	2.70E-03	2.49E-01	5.35E-02	7.92E-05	3.30E-02	1.56E-04	3.77E-04	0.00E+00	1.75E-02	2.41E-01	1.60E-02	2.43E-02	1.32E-02	1.17E-01	6.95E-05	1.65E+0
0	2	1.19E-01	0.00E+00	1.04E-02	7.47E-01	3.56E-03	2.90E-03	2.62E-01	5.04E-02	3.75E-04	3.25E-02	1.59E-04	3.86E-04	0.00E+00	1.78E-02	2.27E-01	1.50E-02	2.42E-02	1.22E-02	1.18E-01	3.95E-05	1.64E+0
0	3	1.18E-01	0.00E+00	9.85E-03	7.62E-01	4.39E-03	2.90E-03	2.61E-01	4.77E-02	3.94E-04	3.25E-02	1.61E-04	3.89E-04	0.00E+00	1.78E-02	2.14E-01	1.50E-02	2.43E-02	1.21E-02	1.17E-01	4.48E-05	1.64E+0
0	4	9.10E-02	0.00E+00	9.17E-03	8.03E-01	4.02E-03	3.10E-03	2.56E-01	5.26E-02	1.41E-04	3.36E-02	1.58E-04	4.04E-04	0.00E+00	1.71E-02	2.06E-01	1.60E-02	2.45E-02	1.33E-02	1.17E-01	3.12E-05	1.65E+0
0	5	8.85E-02	0.00E+00	8.92E-03	8.17E-01	3.87E-03	2.61E-03	2.56E-01	5.16E-02	4.83E-04	3.34E-02	1.63E-04	4.14E-04	0.00E+00	1.78E-02	1.90E-01	1.49E-02	2.42E-02	1.21E-02	1.18E-01	1.97E-06	1.64E+0
0	6	8.43E-02	0.00E+00	8.96E-03	8.12E-01	3.82E-03	2.84E-03	2.62E-01	5.03E-02	4.79E-04	3.34E-02	1.59E-04	4.20E-04	0.00E+00	1.77E-02	1.98E-01	1.47E-02	2.41E-02	1.19E-02	1.18E-01	6.32E-05	1.64E+0
0	7	8.15E-02	0.00E+00	7.41E-03	8.72E-01	3.86E-03	2.89E-03	2.63E-01	5.07E-02	1.04E-04	3.41E-02	1.66E-04	4.11E-04	0.00E+00	1.64E-02	1.42E-01	1.60E-02	2.45E-02	1.32E-02	1.17E-01	0.00E+00	1.65E+0
0	8	1.37E-01	0.00E+00	8.56E-03	7.72E-01	4.10E-03	3.17E-03	2.73E-01	4.33E-02	1.80E-04	2.11E-02	1.61E-04	1.69E-04	0.00E+00	1.51E-02	2.05E-01	1.52E-02	2.50E-02	1.25E-02	1.17E-01	0.00E+00	1.65E+0
0	9	8.53E-02	0.00E+00	8.13E-03	8.30E-01	3.85E-03	2.71E-03	2.67E-01	5.11E-02	4.43E-04	3.39E-02	1.59E-04	3.97E-04	0.00E+00	1.74E-02	1.79E-01	1.48E-02	2.43E-02	1.20E-02	1.19E-01	1.70E-05	1.65E+0
0	10	8.15E-02	0.00E+00	7.85E-03	8.41E-01	3.81E-03	2.79E-03	2.62E-01	5.32E-02	1.26E-04	3.42E-02	1.55E-04	3.87E-04	0.00E+00	1.66E-02	1.80E-01	1.59E-02	2.43E-02	1.32E-02	1.18E-01	4.19E-05	1.65E+0
0	11	9.39E-02	0.00E+00	9.61E-03	7.51E-01	4.49E-03	2.97E-03	2.72E-01	5.11E-02	3.51E-04	3.33E-02	1.58E-04	3.50E-04	0.00E+00	1.79E-02	2.32E-01	1.46E-02	2.40E-02	1.19E-02	1.18E-01	1.66E-05	1.64E+0
CMG 1 total		1.04E-01	0.00E+00	9.23E-03	7.89E-01	4.18E-03	4.78E-03	2.61E-01	6.08E-02	3.78E-04	1.46E-01	2.72E-04	4.75E-04	0.00E+00	2.16E-02	2.02E-01	1.61E-02	2.54E-02	1.33E-02	1.19E-01	2.71E-05	1.78E+0

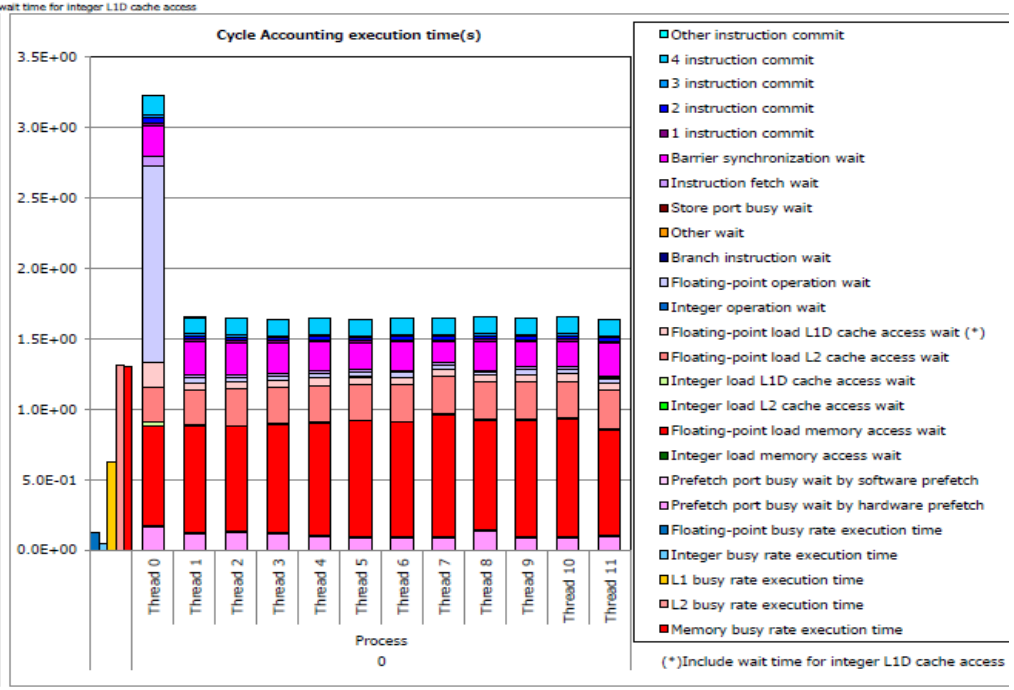
(*) Include wait time for integer L1D cache access

er ion B ate	L1 busy rate (%)	L2 busy rate (%)	Memory busy rate (%)	Address calculation operation pipeline A busy rate (%)	Address calculation operation pipeline B busy rate (%)	Floating-point pipeline A Active element rate (%)	Floating-point pipeline B Active element rate (%)	L1 pipeline 0 Active element rate (%)	L1 pipeline 1 Active element rate (%)	SFI(Store Fetch Interlock) rate
48%	22.00%	73.69%	73.46%	5.56%	5.66%	47.69%	100.00%	99.99%	99.99%	0.00
12%	37.34%			9.56%	9.96%	43.40%	100.00%	99.99%	99.99%	0.00
11%	38.12%			9.49%	9.97%	43.44%	100.00%	99.99%	99.99%	0.00
13%	37.96%			9.54%	10.02%	43.47%	100.00%	99.99%	99.99%	0.00
14%	37.35%			9.62%	10.01%	43.39%	100.00%	99.99%	99.99%	0.00
13%	37.85%			9.56%	10.03%	43.41%	100.00%	99.99%	99.99%	0.00
16%	37.92%			9.63%	10.09%	43.40%	100.00%	99.99%	99.99%	0.00
14%	37.86%			9.60%	9.97%	43.38%	100.00%	99.99%	99.99%	0.00
07%	37.77%			9.37%	9.87%	43.54%	100.00%	99.99%	99.99%	0.00
12%	37.67%			9.53%	9.99%	43.40%	100.00%	99.99%	99.99%	0.00
14%	37.38%			9.60%	9.98%	43.35%	100.00%	99.99%	99.99%	0.00
11%	37.74%			9.50%	9.97%	43.39%	100.00%	99.99%	99.99%	0.00
03%	35.33%	73.69%	73.46%	8.94%	9.33%	43.88%	100.00%	99.99%	99.99%	0.00

iss oad- ion)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)	L1D TLB miss rate (/Load-store instruction)	L2D TLB miss rate (/Load-store instruction)
0.19	2.95%	97.15%	-0.10%	1.11E+08	0.15	14.17%	86.13%	0.00%	0.00109	0.00007
0.19	3.67%	96.37%	-0.04%	1.03E+08	0.15	16.84%	83.46%	0.00%	0.00013	0.00000
0.19	4.31%	95.77%	-0.08%	1.03E+08	0.15	17.42%	82.99%	0.00%	0.00014	0.00000
0.19	3.95%	96.09%	-0.04%	1.03E+08	0.15	17.69%	82.71%	0.00%	0.00015	0.00000
0.19	3.37%	96.73%	-0.10%	1.05E+08	0.16	18.12%	82.16%	0.00%	0.00014	0.00000
0.19	3.46%	96.72%	-0.17%	1.05E+08	0.16	18.69%	81.66%	0.00%	0.00014	0.00000
0.19	3.47%	96.66%	-0.13%	1.05E+08	0.16	18.99%	81.31%	0.00%	0.00015	0.00000
0.19	3.43%	96.72%	-0.15%	1.06E+08	0.16	20.36%	80.03%	0.00%	0.00014	0.00000
0.19	3.12%	96.95%	-0.07%	1.05E+08	0.16	18.27%	82.11%	0.00%	0.00014	0.00000
0.19	3.41%	96.70%	-0.11%	1.05E+08	0.16	18.90%	81.49%	0.00%	0.00014	0.00000
0.19	3.37%	96.74%	-0.11%	1.06E+08	0.16	18.59%	81.75%	0.00%	0.00015	0.00000
0.19	3.50%	96.67%	-0.17%	1.02E+08	0.15	16.01%	84.29%	0.00%	0.00014	0.00000
0.19	3.50%	96.61%	-0.11%	1.26E+09	0.16	17.83%	82.51%	0.00%	0.00023	0.00001







Load-store instruction											Prefetch instruction			DCZVA instruction	Floating-point instruction			Floating-point move and conversion instruction		Integer instruction	Branch instruction	Predicate instruction	Crypto-graphic instruction	Other instruction	Total
d instruction					Store instruction																				
				Non-SIMD	SIMD					Non-SIMD															
ast ion	Floating-point register fill instruction	Predicate register fill instruction	First-fault load instruction	Non-SIMD load instruction	Single vector contiguous store instruction	Multiple vector contiguous structure store instruction	Non-contiguous scatter store instruction	Floating-point register spill instruction	Predicate register spill instruction	Non-SIMD store instruction	Contiguous prefetch instruction	Gathering prefetch instruction	Scalar prefetch instruction		Floating-point instruction except FMA and reciprocal	FMA instruction	Floating-point reciprocal instruction	Floating-point conversion instruction	Floating-point move instruction						

Power Consumption (W)		Power consumption used by core	Power consumption used by L2 cache	Power consumption used by memory
Process	Thread			

Forge

Debug, profile and analyse at scale

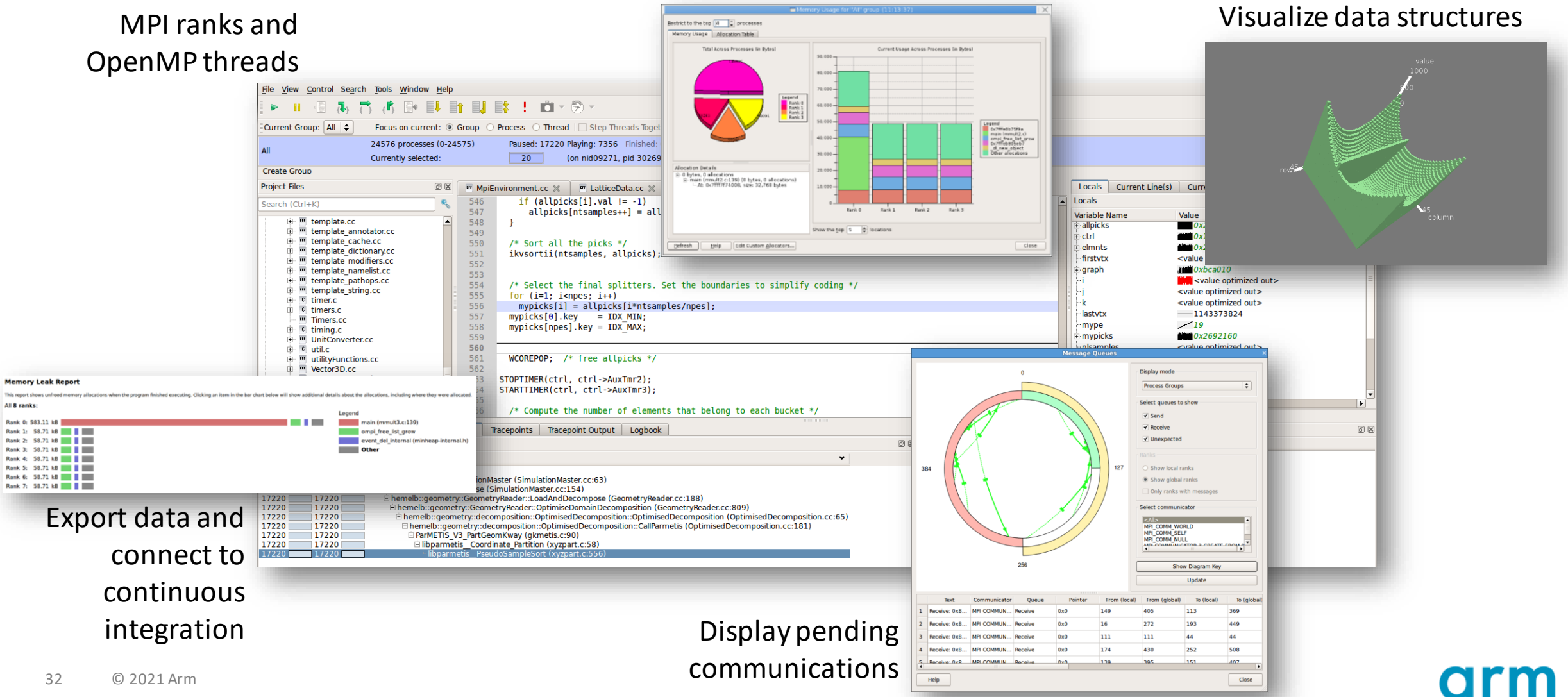
- CUDA GDB is now shipped on AArch64
- Assembly debugging mode in DDT
- RHEL 8 support
- GDB 8.2 (default)
- Python debugging in DDT
- Perf Metrics support for A76 and Neoverse-N1 CPUs
- Configurable perf metric support
- Statistical Profiling Extension (SPE) in MAP
- Release 21.0

Arm Forge – DDT Parallel Debugger

Switch between MPI ranks and OpenMP threads

Analyze memory usage

Visualize data structures



Export data and connect to continuous integration

Display pending communications

Arm Forge – MAP Multi-node Low-overhead Profiler

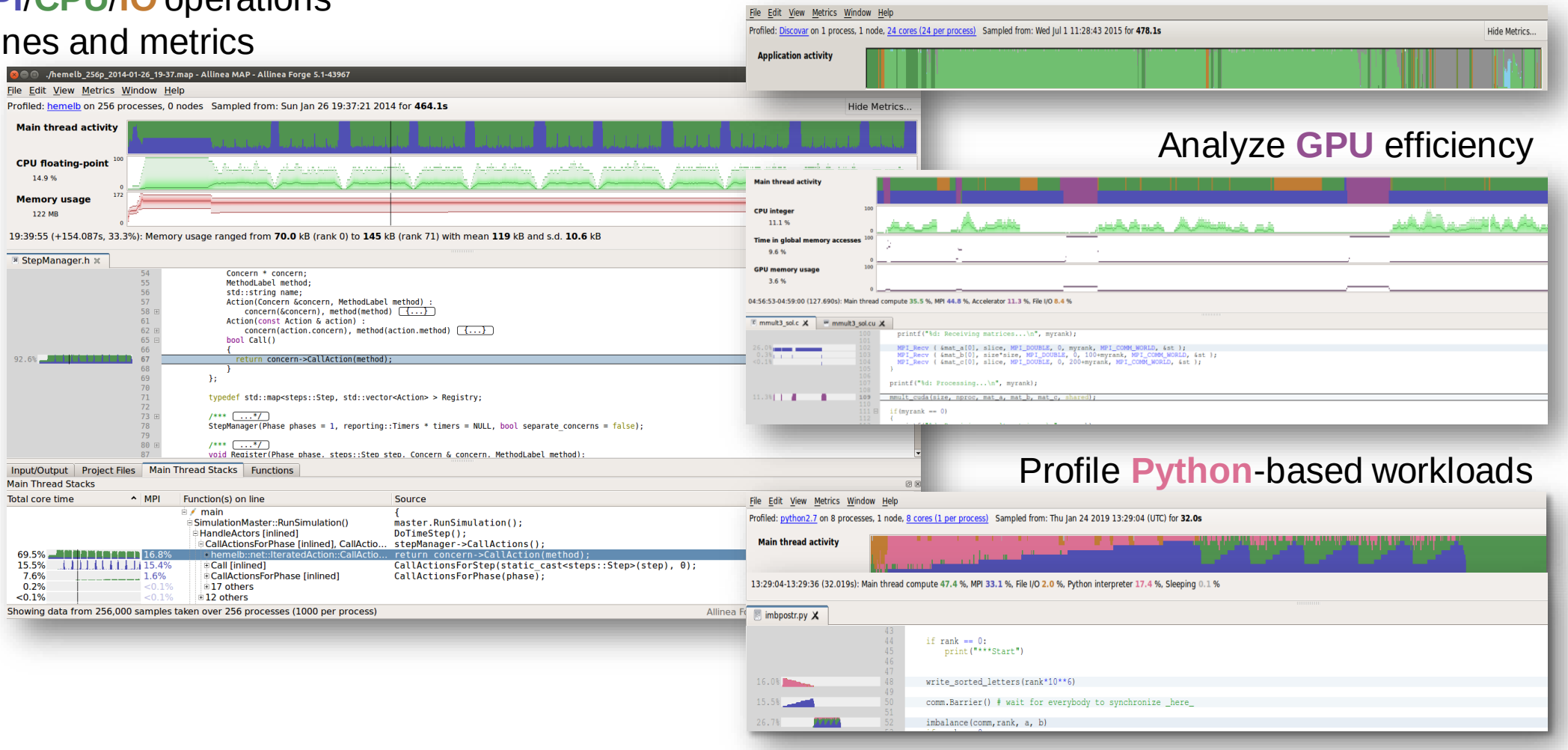
Understand **MPI**/**CPU**/**IO** operations thanks to timelines and metrics

Inspect **OpenMP** activity

Analyze **GPU** efficiency

Investigate annotated source code and stack view

Profile **Python**-based workloads



Guides, Examples, and References

http://developer.arm.com/hpc

- **Arm Reference Guides**
  - https://developer.arm.com/tools-and-software/server-and-hpc/help/help-and-tutorials
  - Compilers, Math Libraries, Debugging and Profiling
- **Porting, Quick Reference and Optimizing Guides**
  - https://developer.arm.com/tools-and-software/server-and-hpc/help/porting-and-tuning

  and
  - https://developer.arm.com/documentation/101725/0200/Steps-and-Tools-to-Port-your-Application

arm

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה