Arm in HPC

Contact: 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**

**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

- **2018**
  - Arm Neoverse Announced
  - VMware demonstrated ESXi on 64-bit ARM

- **2019**
  - Neoverse N1 & E1 Platform announced
  - Huawei released Kunpeng 920 CPU and TaiShan server platform
  - Nvidia CUDA stack to Arm Platforms
  - Marvell's ThunderX2 Solution for Microsoft Azure Development

- **2020**
  - AWS announced 2nd Generation Arm-based Graviton2 Server CPU
  - Marvell Announced 96-core ThunderX3 Server SoC
  - Fujitsu Fugaku Riken #1 on Top500
  - Ampere announced industry’s 1st 80-Core Server SoC (128 Altra Max)
  - EPI Zeus License

Nvidia CUDA stack to Arm Platforms
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)**

<table>
<thead>
<tr>
<th>(TOPS)</th>
<th>HPC</th>
<th>AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td>21.6+</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>10.8+</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- A64FX (Fugaku)
- SPARC64 VIII fx (K computer)

Copyright 2019 FUJITSU LIMITED
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\textsuperscript{nd}, handed over Oct 29\textsuperscript{th}
  - Accepted Nov 9\textsuperscript{th}
  - Upgrade to final B2 TX2 silicon, firmware, CPE completed March 15\textsuperscript{th} 2019

http://gw4.ac.uk/isambard/
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
Software Ecosystem
**OS**
- RHEL, SUSE, CentOS, Ubuntu, ...

**Arm Server Ready Platform**
- Standard firmware and RAS

**OEM/ODM’s**
- Cray-HPE, ATOS-Bull, Fujitsu, Gigabyte, ...

**Silicon Suppliers**
- Marvell, Fujitsu, Mellanox, NVIDIA, ...

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

**Containers, Interpreters, etc.**
- Singularity, PodMan, Docker, Python, ...

**Debuggers & Profilers**
- Arm Forge (DDT, MAP), Rogue Wave, HPC Toolkit, Scalasca, Vampir, TAU, ...

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

**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, ...

**Schedulers**
- SLURM, IBM LSF, Altair PBS Pro, ...

**Cluster Management**
- Bright, HPE CMU, XCAT, Warewulf, ...

**Libraries**
- ArmPL, FFTW, OpenBLAS, NumPy, SciPy, Trilinos, PETSc, Hypre, SuperLU, ScALAPACK, ...

**Debuggers & Profilers**
- Arm Forge (DDT, MAP), Rogue Wave, HPC Toolkit, Scalasca, Vampir, TAU, ...

**Filesystems**
- BeeGFS, Lustre, ZFS, HDF5, NetCDF, ...

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

**Container, Interpreters, etc.**
- Singularity, PodMan, Docker, Python, ...

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

**OEM/ODM’s**
- Cray-HPE, ATOS-Bull, Fujitsu, Gigabyte, ...

**Silicon Suppliers**
- Marvell, Fujitsu, Mellanox, NVIDIA, ...

**OS**
- RHEL, SUSE, CentOS, Ubuntu, ...

**Arm Server Ready Platform**
- Standard firmware and RAS
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

Chem/Phys  Weather   CFD  Visualization  Genomics  AI/ML
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

Code Generation
for Arm servers

Performance Engineering
cross platform, scalable

Server & HPC Solution
for Arm servers

arm
COMPILER
FOR LINUX

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

arm FORGE

arm DDT
Debugger

arm MAP
Profiler

arm PERFORMANCE REPORTS
Reporting

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 (DDT)
- Profiler (MAP)
Arm Compiler for Linux
a.k.a Arm Compiler for HPC, a.k.a. Arm Allinea Compiler

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 C/C++/Fortran Compiler

- C/C++ Files (.c/.cpp) → Clang based → C/C++ Frontend → LLVM IR → IR Optimizations → Optimizer → Language agnostic optimization → LLVM IR → Armv8-A code-gen → Armv8-A binary
- Fortran Files (.f/.f90) → PGI Flang based → Fortran Frontend → LLVM IR → Enhanced optimization for Armv8-A and SVE → LLVM IR → SVE code-gen → SVE binary

Language specific frontend

Architecture specific backend

Building on LLVM, Clang and Flang projects.
Optimized BLAS, LAPACK and FFT

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

- 1-d
- complex-to-complex
- double precision
- ThunderX2

Arm Perf Libs better than FFTW (speed-up > 1)

FFTW better than Arm Perf Libs (speed-up < 1)

Performance parity (speed-up = 1)
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 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 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 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

<table>
<thead>
<tr>
<th>PAPI</th>
<th>perf_events</th>
<th>Documentation and Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Support for many aarch64 server-class CPUs:</td>
<td>• Native HPM API is fully supported</td>
<td>• Arm MAP, HPCToolkit, IPM, TAU, ScoreP, etc.</td>
</tr>
<tr>
<td>• e.g. ThunderX2</td>
<td>• User applications may:</td>
<td>• HPM values can be accessed by non-privileged users in a secure manner</td>
</tr>
<tr>
<td>• Marvell planning support for future CPUs e.g. ThunderX4</td>
<td>• Initialize the HPM</td>
<td>• Performance metrics derived from multiple counters:</td>
</tr>
<tr>
<td></td>
<td>• Initiate and reset counters</td>
<td>• Partners provide their own PMU/HPM documentation</td>
</tr>
<tr>
<td></td>
<td>• Read counters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Generate interrupts on counter overflow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Register interrupt handlers from each process and thread independently</td>
<td></td>
</tr>
</tbody>
</table>
Arm Forge Ultimate
A cross-platform toolkit for debugging, profiling and performance analysis

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 petaflopic 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

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

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

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

Qualify the type of workload

Follow guidance advices for your next steps and maximize output