Arm HPC Solution

A partner enabled ecosystem
HPC Strategy

Mission:
Enable the world’s first Arm supercomputer(s)

Strategy:
Enablement + Co-Design + Partnership

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
Software Tools
But to engage with customers, we need a reason...

<table>
<thead>
<tr>
<th>Arm Allinea Studio</th>
<th>Develop and run on today’s hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arm Compiler for HPC</td>
<td>Linux user space compiler for HPC applications</td>
</tr>
<tr>
<td>Arm Performance Libraries</td>
<td>BLAS, LAPACK and FFT</td>
</tr>
<tr>
<td>Arm Forge Professional</td>
<td>Multi-node interoperable profiler and debugger</td>
</tr>
<tr>
<td>Arm Performance Reports</td>
<td>Interoperable application performance insight</td>
</tr>
</tbody>
</table>

**Key benefits:**
- Get the best performance out of Arm hardware
- Reduce the time to develop codes
- Improve productivity by speeding up applications

**and, importantly...**
- Migrate from x86 to aarch64 faster with portable tools
- Forge/PR licences are a great justification to stay close to our end-users
Arm Compiler – Building on LLVM, Clang and Flang projects

Arm C/C++/Fortran Compiler

- **C/C++ Files (.c/.cpp)**
  - C/C++ Frontend
  - Clang based
  - LLVM IR

- **Fortran Files (.f/.f90)**
  - Fortran Frontend
  - PGI Flang based

- **Optimizer**
  - IR Optimizations
    - Auto-vectorization
    - Enhanced optimization for ARMv8-A and SVE
  - LLVM IR
  - Language agnostic optimization

- **Armv8-A Backend**
  - LLVM based
  - Armv8-A binary

- **SVE Backend**
  - LLVM based
  - SVE binary

Language specific frontend
Language agnostic optimization
Architecture specific backend
Open Architecture implies other software tool options.

• GNU-gcc
  • including internal Arm developers

• LLVM-clang
  • including internal Arm developers
  • All enhancements to LLVM for the Arm HPC compilers are pushed upstream
    – Using PGI-flang as base for armflang

• Cray
  • Complete, HPC-optimized software stack including the Cray Linux® Environment and Cray Programming Environment. Cray’s CCE compiler and programming environment are enhanced to achieve improved performance from the Cavium ThunderX2 processors.
  • I assume craypat is a functioning profiler on aarch64

• Fujitsu
Hardware options
Cavium CN99XX - 1st member of

- 24/28/32 Custom ARMv8 cores
- Fully Out-Of-Order (OOO) Execution
- 1S and 2S Configuration
- Up to 8 DDR4 Memory Controllers
- Up to 16 DIMMs per Socket
- Server Class RAS features
- Server class virtualization
- Integrated IOs
- Extensive Power Management

2nd gen ARM server SoC
Delivers 2-3X higher performance
Fujitsu Post-K specs

**Architecture Features**
- Armv8.2-A (AArch64 only)
- SVE 512-bit wide SIMD
- 48 computing cores + 4 assistant cores*  
  *All the cores are identical
- HBM2 32GiB
- Tofu 6D Mesh/Torus  
  28Gbps x 2 lanes x 10 ports
- PCIe Gen3 16 lanes

**7nm FinFET**
- 8,786M transistors
- 594 package signal pins

**Peak Performance (Efficiency)**
- >2.7TFLOPS (>90%@DGEMM)
- Memory B/W 1024GB/s (>80%@Stream Triad)
**Fujitsu Post-K attributes for HPC applications.**

<table>
<thead>
<tr>
<th>Features</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-lane predication</td>
<td>High vectorization rate</td>
</tr>
<tr>
<td>Fault-tolerant speculative vectorization</td>
<td>Wider SIMD (512-bit wide for Post-K)</td>
</tr>
<tr>
<td>Gather-load and scatter-store</td>
<td>Efficient utilization of vector</td>
</tr>
<tr>
<td>Horizontal and serialized vector operations</td>
<td>e.g. Gather/Scatter for packed 32-bit FP,</td>
</tr>
<tr>
<td></td>
<td>Packed SIMD for 1-, 2-, 4- and 8-byte integer</td>
</tr>
<tr>
<td>HPC-focused instructions</td>
<td>Highly optimized executables</td>
</tr>
<tr>
<td>e.g. Reciprocal inst., Math. acceleration inst., etc.</td>
<td>Binary portability between different vector-length CPUs</td>
</tr>
</tbody>
</table>
ArmV8 and HPC
Scalable Vector Extension

• Not an extension to NEON
• advanced loads & stores
• white-paper: https://developer.arm.com/-/media/developer/developers/hpc/white-papers/a-sneak-peek-into-sve-and-vla-programming.pdf?revision=c702475b-6325-41a2-b3d3-d9f244028841
  • Francesco Petrogalli
• Vector Length Agnostic (VLA) programming because of predication
  • can vectorize loops with control flow in the loop body
• Run SVE code on non–sve platforms with ARMIE: https://developer.arm.com/products/software-development-tools/hpc/arm-instruction-emulator
• SC’18 workshop
Applications on Aarch64
Bristol via GW4 is an active co-designer.

Cavium ThunderX2, a seriously beefy CPU

- 32 cores at up to 2.5GHz
- Each core is 4-way superscalar, Out-of-Order
- 32KB L1, 256KB L2 per core
- Shared 32MB L3
- Dual 128-bit wide NEON vectors
  - Compared to Skylake’s 512-bit vectors, and Broadwell’s 256-bit vectors
- 8 channels of 2666MHz DDR4
  - Compared to 6 channels on Skylake, 4 channels on Broadwell
- AMD’s EPYC also has 8 channels
The orange is different from the apples.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Cores</th>
<th>Clock speed GHz</th>
<th>FP64 TFLOP/s</th>
<th>Bandwidth GB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadwell</td>
<td>2 × 22</td>
<td>2.2</td>
<td>1.55</td>
<td>154</td>
</tr>
<tr>
<td>Skylake (Gold)</td>
<td>2 × 20</td>
<td>2.4</td>
<td>3.07</td>
<td>256</td>
</tr>
<tr>
<td>Skylake (Platinum)</td>
<td>2 × 28</td>
<td>2.1</td>
<td>3.76</td>
<td>256</td>
</tr>
<tr>
<td>Knights Landing</td>
<td>64</td>
<td>1.3</td>
<td>2.66</td>
<td>~490</td>
</tr>
<tr>
<td>ThunderX2</td>
<td>2 × 32</td>
<td>2.2</td>
<td>1.13</td>
<td>320</td>
</tr>
</tbody>
</table>

**BDW 22c** Intel Broadwell E5-2699 v4, **$4,115** each (near top-bin)

**SKL 20c** Intel Skylake Gold 6148, **$3,078** each

**SKL 28c** Intel Skylake Platinum 8176, **$8,719** each (near top-bin)

**TX2 32c** Cavium ThunderX2, **$1,795** each (near top-bin)

http://gw4.ac.uk/sambard/
Memory bound apps do perform better on TX2.

Comparative Benchmarking of the First Generation of HPC-Optimised Arm Processors on Isambard

S. McIntosh-Smith, J. Price, T. Deakin and A. Poenaru, CUG 2018, Stockholm
What is the cost for your performance?

This is highly subjective to the market and procurement deals.
No single compiler is the best.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>ThunderX2</th>
<th>Broadwell</th>
<th>Skylake</th>
<th>Xeon Phi</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAM</td>
<td>GCC 7</td>
<td>Intel 18</td>
<td>Intel 18</td>
<td>Intel 18</td>
</tr>
<tr>
<td>CloverLeaf</td>
<td>Arm 18.2</td>
<td>Intel 18</td>
<td>Intel 18</td>
<td>Intel 18</td>
</tr>
<tr>
<td>TeaLeaf</td>
<td>GCC 7</td>
<td>Intel 18</td>
<td>Intel 18</td>
<td>Intel 18</td>
</tr>
<tr>
<td>SNAP</td>
<td>CCE 8.6</td>
<td>Intel 18</td>
<td>Intel 18</td>
<td>Intel 18</td>
</tr>
<tr>
<td>Neutral</td>
<td>GCC 7</td>
<td>Intel 18</td>
<td>Intel 18</td>
<td>Intel 18</td>
</tr>
<tr>
<td>CP2K</td>
<td>GCC 7</td>
<td>GCC 7</td>
<td>GCC 7</td>
<td>—</td>
</tr>
<tr>
<td>GROMACS</td>
<td>GCC 7</td>
<td>GCC 7</td>
<td>GCC 7</td>
<td>—</td>
</tr>
<tr>
<td>NAMD</td>
<td>Arm 18.2</td>
<td>GCC 7</td>
<td>Intel 18</td>
<td>—</td>
</tr>
<tr>
<td>NEMO</td>
<td>CCE 8.7</td>
<td>CCE 8.7</td>
<td>CCE 8.7</td>
<td>—</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>GCC 7</td>
<td>GCC 7</td>
<td>GCC 7</td>
<td>—</td>
</tr>
<tr>
<td>OpenSBLI</td>
<td>CCE 8.7</td>
<td>Intel 18</td>
<td>Intel 18</td>
<td>—</td>
</tr>
<tr>
<td>UM</td>
<td>CCE 8.6</td>
<td>CCE 8.5</td>
<td>CCE 8.6</td>
<td>—</td>
</tr>
<tr>
<td>VASP</td>
<td>CCE 8.7</td>
<td>CCE 8.6</td>
<td>CCE 8.6</td>
<td>—</td>
</tr>
</tbody>
</table>
The HPC community has options to influence Arm

- Arm has made the effort to help port many applications:
  - [https://gitlab.com/arm-hpc/packages/wikis/categories/allPackages](https://gitlab.com/arm-hpc/packages/wikis/categories/allPackages)
- Please contribute and enhance ported packages during your campaigns
- [support-hpc-sw@arm.com](mailto:support-hpc-sw@arm.com) with concerns
- [https://spack.io](https://spack.io)
  - HPC deployment package manager
  - patch in for IP review for armhpc compiler
  - ?standardization of benchmarks?
- SVE (starting with 512 bit vector units) will bring more performance to the table
  - compiler implementations will always be advancing
Arm help
HPC Infrastructure Tools Group
Worldwide HPC knowledge experts in cross-platform enablement and performance optimization

Teams:

• Professional Services
• Commercial and open-source SW engineering
• HPC Tools Development, training and support

Offerings:

HPC application modernization, porting, and optimization:
• Identify and resolve scalability bottlenecks.
• Optimize application data movement and vectorization.

Performance analysis tools and techniques:
• Deploy and support software tools from Arm and the Arm community.
• Documentation, training, tutorials, and workshops.

Knowledge transfer and Arm adoption:
• Publications, presentations, and community participation.
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद