Applications performance on Arm HPC architecture

International Computing for the Atmospheric Sciences Symposium@Stresa, Italy

Philip Ridley
September 11th, 2019
Arm Technology Already Connects the World

Arm is ubiquitous

Partnership is key

Choice is good

21 billion chips sold by partners in 2017 alone

We design IP, not manufacture chips

One size is not always the best fit for all

Mobile/Embedded/IoT/Automotive/Server/GPUs

Partners build products for their target markets

HPC is a great fit for co-design and collaboration
History of Arm in HPC

2011 Calxada
- 32-bit Armv7-A – Cortex A9

2011-2015 Mont-Blanc 1
- 32-bit Armv7-A
- Cortex A15
- First Arm HPC system

2014 AMD Opteron A1100
- 64-bit Armv8-A
- Cortex A57
- 4-8 Cores

2015 Cavium ThunderX
- 64-bit Armv8-A
- 48 Cores

2017 (Cavium) Marvell ThunderX 2
- 64-bit Armv8-A
- 32 Cores

2019 Fujitsu A64FX
- First Arm chip with SVE vectorisation
- 48 Cores
So Why Bother with Arm?

- Everything is designed to look the same to a user
  - Consistency at the software level

- Arm architecture lets manufacturers differentiate
  - Target different markets with specific hardware (relevant to that market)
  - Not just rely on mass volume - general purpose chips

- In HPC it lets us get the hardware features relevant to our codes
  - High memory bandwidth, vectorisation, fast networks, etc.

- Faster to adapt to changes in market demand
  - AI / ML demands, enhanced instructions
Arm in HPC: Does it Matter?

- For users of HPC as a tool Arm should be boring
  - Everything should just work

- Looks and feels the same as any other HPC system
  - Intel or AMD

- Comprehensive software ecosystem
  - All of the familiar software packages
  - Same user experience for building software

- Lets users focus on the science not the computer

<table>
<thead>
<tr>
<th>Functional Areas</th>
<th>Components include</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base OS</td>
<td>RHEL, Centos, SLES, Ubuntu</td>
</tr>
<tr>
<td>Administrative Tools</td>
<td>Conman, Ganglia, Lmod, LosF, Nagios, pdsh, pdsh-mod-slurm, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, test-suite</td>
</tr>
<tr>
<td>Scheduler</td>
<td>SLURM, PBS Pro, Torque</td>
</tr>
<tr>
<td>I/O Services</td>
<td>NFS, Lustre, BeeGFS, CEPH</td>
</tr>
<tr>
<td>I/O Libraries</td>
<td>HDF5 (pHDF5), NetCDF (including C++ and Fortran interfaces), Adios</td>
</tr>
<tr>
<td>Compiler Families</td>
<td>GNU, LLVM, Cray, Arm, Fujitsu</td>
</tr>
<tr>
<td>MPI Families</td>
<td>OpenMPI, MPICH, MVAPICH2, HPE, Cray</td>
</tr>
<tr>
<td>Development Tools</td>
<td>Autotools (autoconf, automake, libtool), Cmake, Valgrind, R, Python, SciPy/NumPy, hwloc</td>
</tr>
<tr>
<td>Performance Tools</td>
<td>PAPI, IMB, pdtoolkit, TAU, Scalasca, Score-P, SIONLib, Arm Forge</td>
</tr>
</tbody>
</table>
Sandia: Vanguard Astra by HPE (Top 500 system)

Arm HPC in the US

- 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
CEA: The French Alternative Energies and Atomic Energy Commission

Arm HPC in France

- 276 BullSequanaX1310 compute nodes
- 17,664 cores, 2.3 PFLOPs (peak)
- **Marvell ThunderX2** ARM SoC, 32 core, 2.2 GHz
- 256GB of memory / node
- Mellanox EDR InfiniBand
- Part of the Mont-Blanc 3 project
Catalyst
Arm HPC in the UK

- Three Catalyst systems, each with
  - 64 HPE Apollo 70 nodes
  - Dual 32-core Marvell ThunderX2 processors
  - 4096 cores
  - 256GB of memory / node
  - Mellanox EDR InfiniBand
- OS: SUSE Linux Enterprise Server for HPC
Isambard @ GW4
Arm HPC in the UK

• **10,752** Arm 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, Cray MPI, Cray LibSci, CrayPat, ...
  • OS: CLE 7.0 (Cray Linux Environment)
•Phase 2 (the Arm part):
  • Accepted Nov 9th 2018
• UK Tier-2 HPC resource
Marvell ThunderX2 CN99XX

- Marvell’s current generation Arm HPC processor
- 32 cores @ 2.5 GHz (other SKUs available)
  - 4 Way SMT (up to 256 threads / node)
  - Fully out of order execution
  - 8 DDR4 Memory channels (~250 GB/s Dual socket)
    - Vs 6 on Skylake
  - L1 is 32KB and L2 is 256KB per core with 32MB distributed L3
- Available in dual SoC configurations
  - CCPI2 interconnect
  - 180-200w / socket
- No SVE vectorisation
  - 128-bit NEON vectorisation
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
Arm HPC Ecosystem website: developer.arm.com/solutions/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
### HPC Compilers for Arm

Options for compiling on Arm

<table>
<thead>
<tr>
<th>GCC Toolchain</th>
<th>Arm HPC Compiler</th>
<th>Vendor Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="GCC Logo" /></td>
<td><img src="image2.png" alt="Arm HPC Compiler Logo" /></td>
<td><img src="image3.png" alt="Vendor Specific Logos" /></td>
</tr>
</tbody>
</table>
Arm Performance Libraries
Highly optimized BLAS, LAPACK, FFT and other routines

• Commercial 64-bit Armv8-A math libraries
  • Commonly used low-level math routines – BLAS(42), LAPACK(1700) and FFT(44)
    – LAPACK 3.8.0
  • Provides FFTW compatible interface
    – Matches that used by AMD’s ACML library
    – Additional compatibility layer to allow users to call using the FFTW3 interface
    – Support for Basic, Advanced, Guru and MPI interfaces

• Optimized for best serial and parallel performance
  • Generic optimizations for Arm HPC architectures
  • Tuning for specific Arm HPC platforms
  • Better than FFTW in many cases

• Validated and supported by Arm
  • Available for a wide range of Arm HPC platforms
  • Validated with NAG’s test suite, a de facto standard
  • Available for Arm compiler or GNU
Debugging and Profiling
Arm DDT

- Dedicated HPC debugger
  - Fortran, C, C++ and Python
- Designed for massively parallel applications
  - Designed for MPI applications
  - Support for OpenMP
- Highly scalable
  - Shown to debug at hundreds of thousands of cores
  - Fast reduction algorithms
- Memory debugging
  - Variable comparison
  - Distributed arrays
- GPU support
  - For NVIDIA CUDA 10
Arm MAP

- MAP is a sampling based scalable profiler
  - Built on same framework as DDT
  - Parallel support for MPI, OpenMP
  - Designed for C/C++/Fortran (partial Python)
- Designed for simple ‘hot-spot’ analysis
  - Stack traces
  - Augmented with performance metrics
- Lossy sampler
  - Throws data away – 1,000 samples / process
  - Low overhead, scalable and small file size
Libamath – increased performance for math.h functions

ELEFUNT run on ThunderX2: cases gfortran(+libamath 19.2), armflang+libamath 19.0-19.2
arm

NEMO 4.0 on TX2
NEMO 4.0

Background

• Nucleus for European Modelling of the Ocean (NEMO) is a state-of-the-art modelling framework for research activities in ocean and climate sciences
• Fortran 90 + MPI – no OpenMP
• Historically NEMO was written with Vector processing in mind
• Version 4.0 was released at beginning of 2019

svn co https://forge.ipsl.jussieu.fr/nemo/svn/NEMO/releases/release-4.0
NEMO 4.0: Building and running on Arm

Gotchas

• NEMO 4.0 + Arm compiler
  • USE IEEE_IS_NAN is required in src/OCE/stpctl.F90
  • Arm compiler 19.3 needed for building XIOS 2.0+

• Try Open MPI 4.0.0 or MPICH 3.3
  • With MPICH only use -map-by core -bind-to core (binding to cores helps)
  • Performance should be very similar with both MPIs

• Open MPI and the timing.output file
  • A bug related to MPI_WTIME incorrectly shows *’s in the formatted output
  • Add -fdefault-double-8 to the compile flags (e.g. in arch/arch-aarch64_gnu.fcm)

• Open MPI must use non-default transport protocols (when using >1 node)
  • E.g. mpirun --mca pml ucx -x UCX_TLS=mm,ud -np 1024 ./nemo

• Make sure you are using lamath (20+% speedup)
  • Link against libamath in arch/arch-aarch64_gnu.fcm
NEMO 4.0: Test case

BENCH

• TX2 specs
  • 32 core dual socket running at 2.2GHz (rev B1) with SMT=1
  • SLES 12 sp3

• BENCH
  • [https://github.com/NEMO-ocean/NEMO-examples/tree/master/BENCH](https://github.com/NEMO-ocean/NEMO-examples/tree/master/BENCH)
  • Single node runs with 64 MPI tasks
  • Build with GCC 8.2.0 + Armpl 19.3 + Open MPI 4.0.0 or MPICH 3.3
  • Processor grid: \( j_p n_i = j_p n_j = 8 \), Global domain: \( n_n_j_s i z e = 362 \), \( n_n_j_s i z e = 332 \), \( n_n_k_s i z e = 75 \)
  • 1 degree resolution – BENCH-1 with 1000 time steps
BENCH-1 Results Summary

- Without libamath: 57 minutes, of this around 10.3% is MPI (9.4% MPI_Recv)
- With libamath: 46.5 minutes, of this around 9.6% is MPI (8.3% MPI_Recv)
- Results similar for Open MPI or MPICH
- Assigning MPI tasks in the same task to core id order as on x86 also makes little difference (i.e. socket 1 0,2,4, ... socket 2 1,3,5, ... )
- Spreading the MPI tasks out over more than just a single node does not help
- Running with SMT=1, 2, or 4 seems to make very little difference
- Could try using different transport protocols
NEMO 4.0: Intra-node performance on TX2

MPI communication time is higher on TX2 than on x86
NEMO 4.0: Inter-node performance GYRE_PISCES

[Graph showing scaling efficiency (%)]

- Cray XC50 (Isambard) GCC 8.2.0 + MPT 7.7.6
- Cray XC50 (Isambard) CCE 8.7.9 + MPT 7.7.6
- Cray XC50 (Isambard) Arm 19.0 + MPT 7.7.6
Links to Online Technical Documentation

Help and tutorials

Porting and tuning guides for various packages

Arm Allinea Studio (C/C++/Fortran Compilers, Performance Libraries and Forge)
The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks