

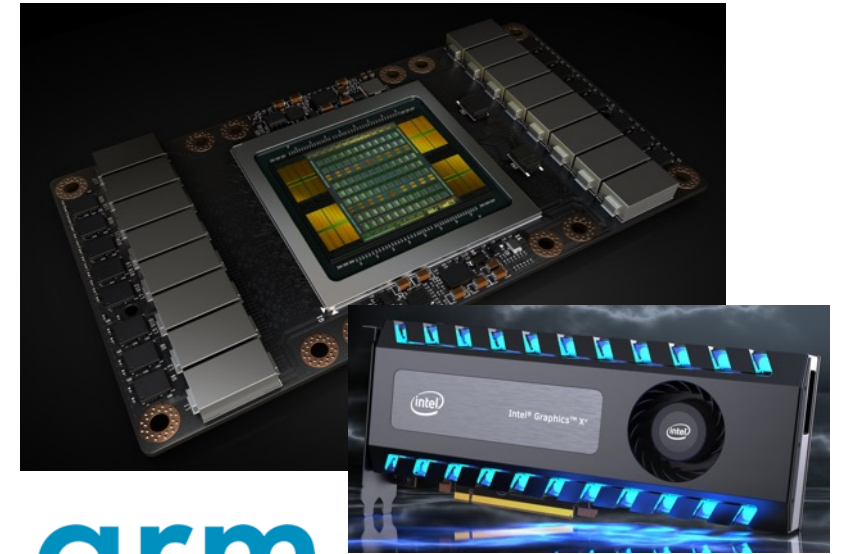
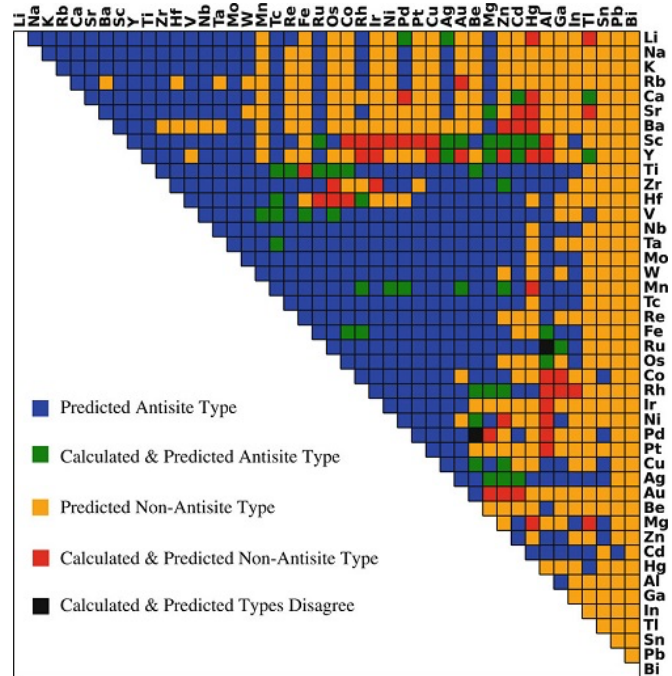
NNSA Explorations: ARM for Supercomputing

- Simon Hammond – Sandia National Laboratories (sdhammo@sandia.gov)
- Howard Pritchard – Los Alamos National Laboratory (howardp@lanl.gov)

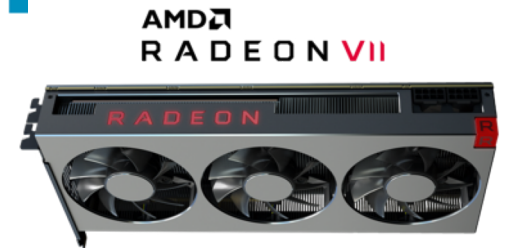
Exciting Time to be in HPC...



EUROPEAN TECHNOLOGY
PLATFORM FOR HIGH
PERFORMANCE COMPUTING



arm



Exascale Computing

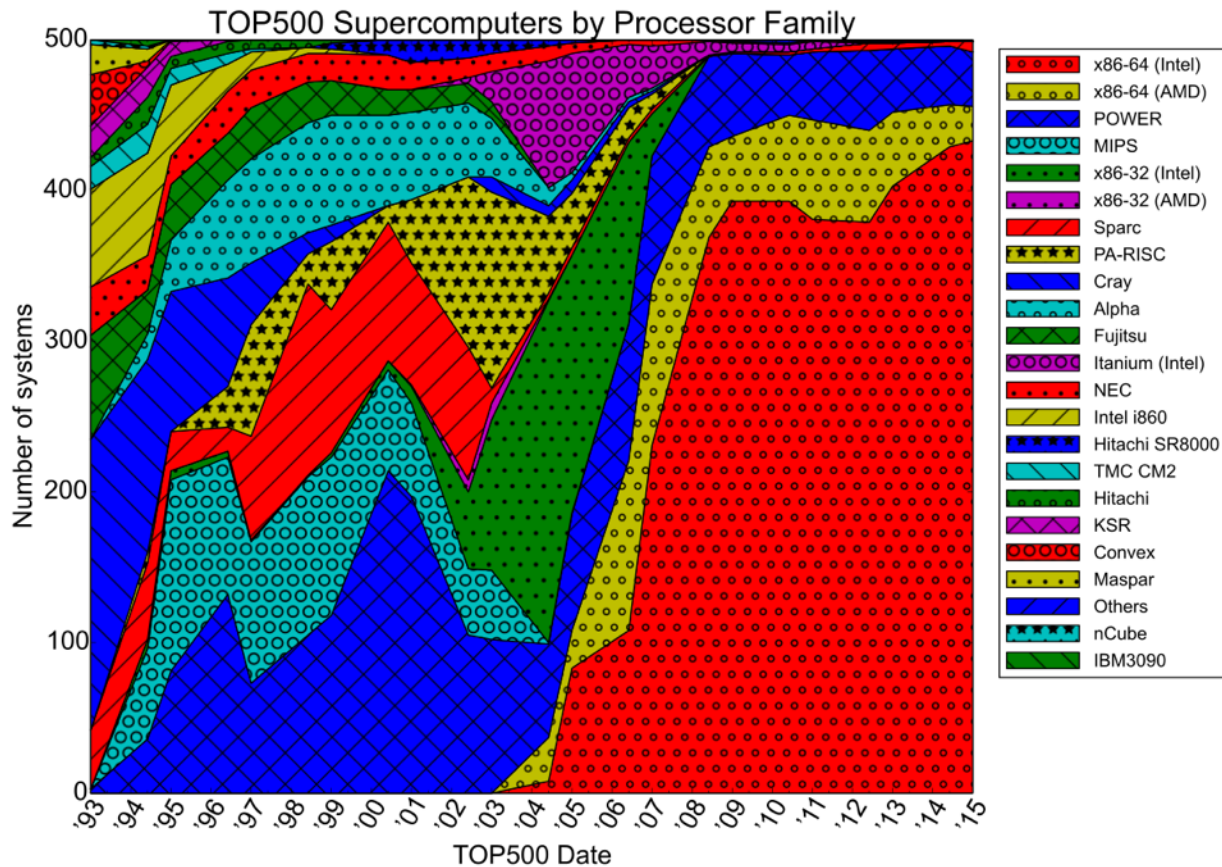
Adoption of ML/AI for HPC

New Hardware/Software

What we'll cover today

- Why Arm – what's so interesting about it?
- Marvell Thunder TX2 overview and comparison with x86_64
- Astra/Vanguard Program
- ASC mini-app and application performance
- Porting to ARM

What's soooooo Interesting About Arm?



- In many ways not much...
 - Its *just* an instruction set
 - As long as it can run Fortran, C and C++ we are good right?
- In others ways quite a lot is interesting
 - Different business model, consortium of implementations
 - Open for partners to suggest new instructions
 - Broad range of intellectual property opportunities
 - Broad(er) range of implementations than say X86, POWER, SPARC *etc*

What's soooooo Interesting About Arm?

- **DOE invests more than \$100M in the hardware of a typical supercomputer (often substantially more than this when the final bill comes in)**

- Competition helps to drive down prices and increase innovation
- We want to optimize price/perf for our machines – get the absolute best workload performance we can for the best price we can buy hardware

- **The future is interesting – Arm is an IP company, not an implementation**

- What if we could blend existing Arm IP blocks with our own DOE inspired accelerators?
- Build workload optimized processors and computers that benefit DOE scientists?
 - e.g. a machine just for designing new materials but one which is 100X faster than today?
- Arm is an opportunity to engage with a broad range of suppliers and an ecosystem
 - Not the only way to do this, can partner with traditional vendors like Intel, IBM, AMD etc

Arm is Growing in HPC...



HPC wire


Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

- Home
- Technologies
- Sectors
- AI/ML/DL
- Exascale

France's CEA and Japan's RIKEN to Partner on ARM and Exascale

By Nishi Katsuya and John Russell

January 19, 2017



Los Alamos National Laboratory

Delivering science and technology to protect our nation and promote world stability

SCIENCE & INNOVATION | COLLABORATION | CAREERS | COMMUNITY

Our Stories » News Releases » News Releases - 2018 » November » Los Alamos pursues efficient computing

Los Alamos pursues efficient computing with Cray, Marvell and Arm

The collaboration with Cray Inc. integrates the Marvell ThunderX2 processors with Cray's proven networking and software ecosystem in the Lab's secure computing environment.



FUJITSU

Services | Products

Home > About Fujitsu > Resource Center > News > Press releases > 2019 > Fujitsu Begins Production of Post-K

Press releases

- > 2019
- > 2018
- > 2017
- > 2016
- > 2015
- > 2014
- > 2013
- > 2012
- > 2011

Fujitsu Begins Production of Post-K

Also advances productization of commercial units based on the supercomputer technology

Fujitsu Limited

Tokyo, April 15, 2019

Fujitsu Limited today announced that it has become the successor to the K supercomputer technology (MEXT) is aimed at now concluded an official agreement. In addition, Fujitsu will provide support for the K development process.

The company's efforts in this regard were held on May 17 at the Tokyo



Sandia National Laboratories

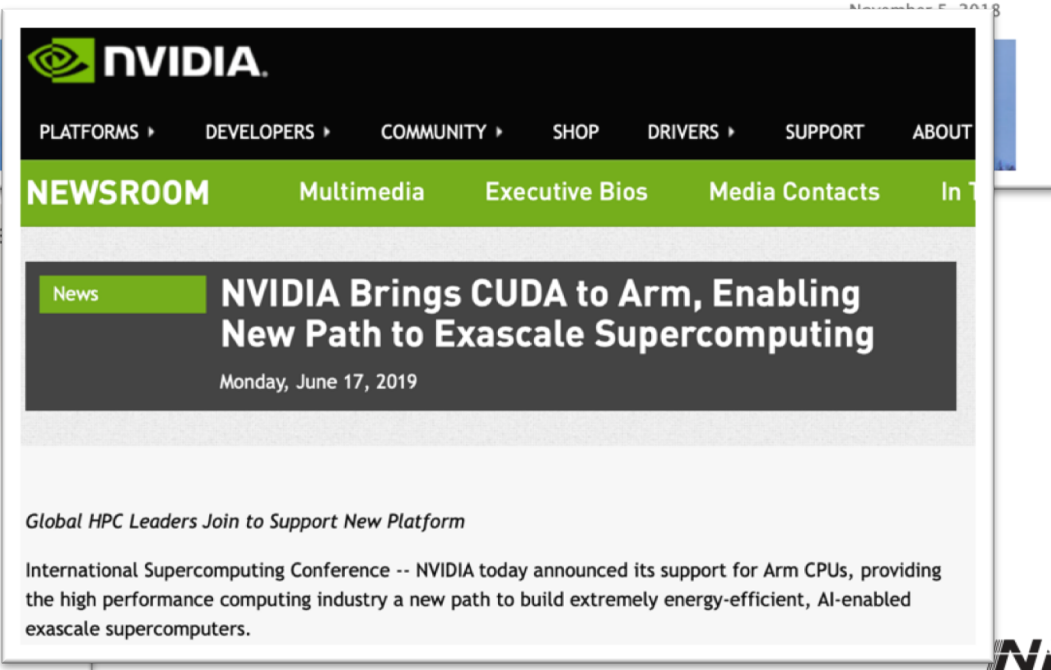
ABOUT | PROGRAMS | RESEARCH

Sandia Labs News Releases

June 18, 2018

Arm-based supercomputer prototype to be deployed at Sandia National Laboratories by DOE

ALBUQUERQUE, N.M. — Microprocessors designed by Arm are ubiquitous in automobile electronics, cellphones and other embedded applications, but until recently they have not provided the performance necessary to make them practical for high-performance



NVIDIA

PLATFORMS | DEVELOPERS | COMMUNITY | SHOP | DRIVERS | SUPPORT | ABOUT

NEWSROOM

Multimedia | Executive Bios | Media Contacts | In This Section

NVIDIA Brings CUDA to Arm, Enabling New Path to Exascale Supercomputing

Monday, June 17, 2019

Global HPC Leaders Join to Support New Platform

International Supercomputing Conference -- NVIDIA today announced its support for Arm CPUs, providing the high performance computing industry a new path to build extremely energy-efficient, AI-enabled exascale supercomputers.

NNSA/ASC Vanguard Program

A proving ground for next-generation HPC technologies in support of the
NNSA mission

<http://vanguard.sandia.gov>

Astra – the First Petscale Arm based Supercomputer

HPE Apollo 70 Chassis: 4 nodes



HPE Apollo 70 Rack



18 chassis/rack

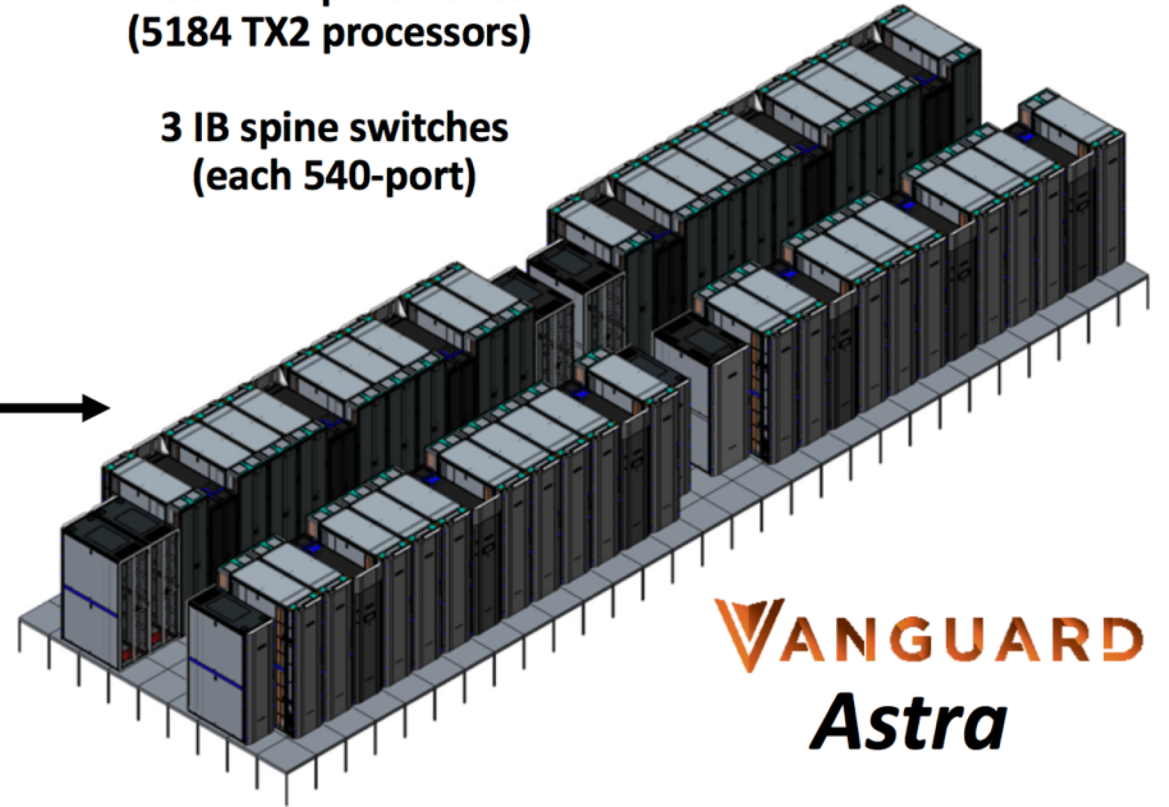
72 nodes/rack

**3 IB switches/rack
(one 36-port switch
per 6 chassis)**

**36 compute racks
(9 scalable units, each 4 racks)**

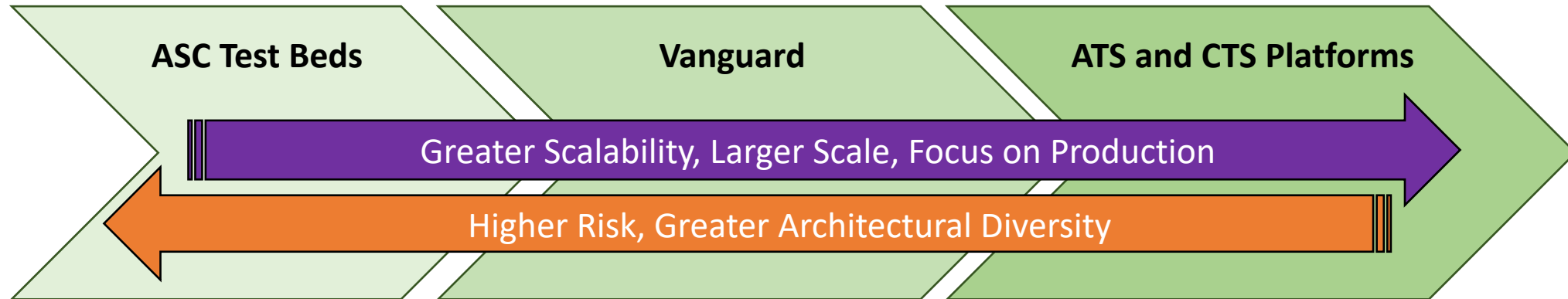
**2592 compute nodes
(5184 TX2 processors)**

**3 IB spine switches
(each 540-port)**



**VANGUARD
Astra**

Where Vanguard Fits in our Program Strategy



Test Beds

- Small testbeds (~10-100 nodes)
- Breadth of architectures Key
- Brave users

Vanguard

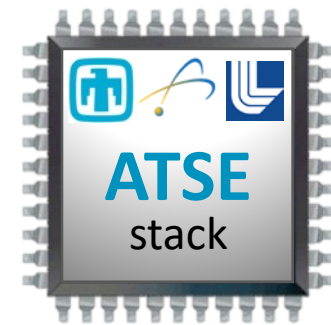
- Larger-scale experimental systems
- Focused efforts to mature new technologies
- Broader user-base
- Not Production
- **Tri-lab resource but not for ATCC runs**

ATS/CTS Platforms

- Leadership-class systems (Petascale, Exascale, ...)
- Advanced technologies, sometimes first-of-kind
- Broad user-base
- Production Use

NNSA/ASC Advanced Trilab Software Environment (ATSE) Project

- Advanced Tri-lab Software Environment
 - Sandia leading development with input from Tri-lab Arm team
 - Will be the user programming environment for Vanguard-Astra
 - Partnership across the NNSA/ASC Labs and with HPE
- Lasting value
 - Documented specification of:
 - Software components needed for HPC production applications
 - How they are configured (i.e., what features and capabilities are enabled) and interact
 - User interfaces and conventions
 - Reference implementation:
 - Deployable on multiple ASC systems and architectures with common look and feel
 - Tested against real ASC workloads
 - Community inspired, focused and supported



ATSE is an integrated software environment for ASC workloads

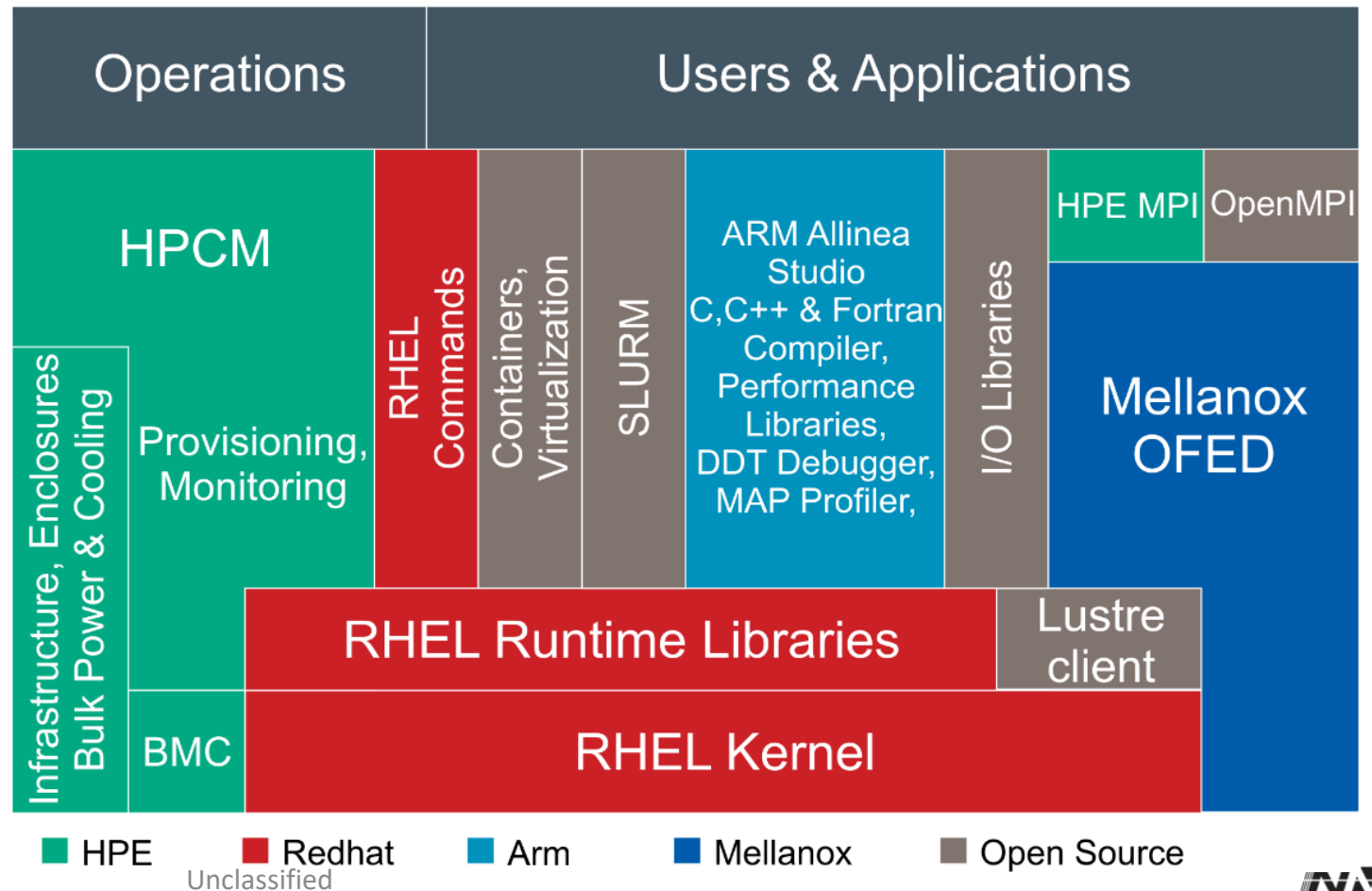
ATSE Collaboration with HPE's HPC Software Stack

HPE's HPC Software Stack

HPE:

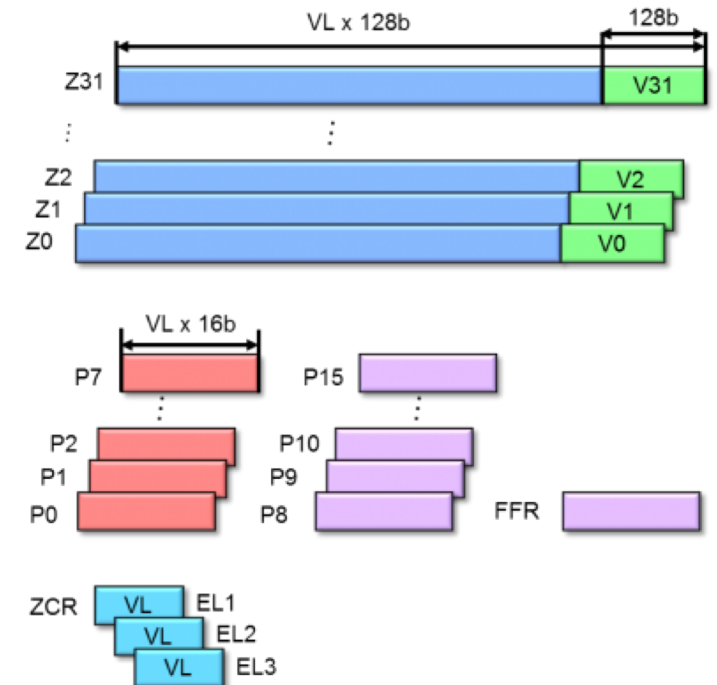
- HPE MPI (+ XPMEM)
- HPE Cluster Manager
- Arm:
 - Arm HPC Compilers
 - Arm Math Libraries
 - Alinea Tools
- Mellanox-OFED & HPC-X
- RedHat 7.x for aarch64


**Hewlett Packard
Enterprise**



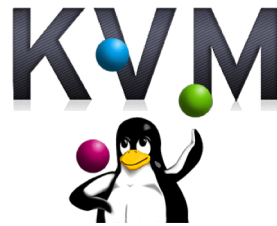
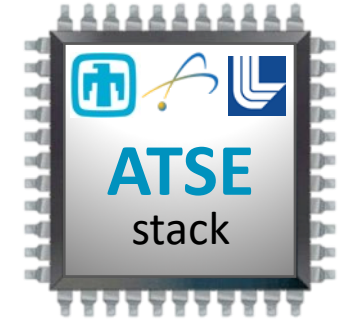
SVE Enablement – Next Generation of SIMD/Vector Instructions

- SVE work is underway
 - SVE = Scalable Vector Extensions
 - Length agnostic vector instructions at an ISA level
 - Using ArmIE (fast emulation) and RIKEN GEM5 Simulator
 - GCC and Arm toolchains
- Collaboration with RIKEN
 - Visited Sandia (participants from SNL, LANL, LLNL, RIKEN)
 - Discussion of performance and simulation techniques
 - Deep-dive on SVE (GEM5)
- Short term plan
 - Use of SVE intrinsics for Kokkos-Kernels SIMD C++/data parallel types
 - Underpins number of key performance routines for Trilinos libraries
 - Seen large (6X) speedups for AVX512 on KNL and Skylake
 - Expect to see similar gains for SVE vector units
 - Critical performance enablement for Sandia production codes



ATSE R&D Efforts – Developing Next-Generation NNSA Workflows

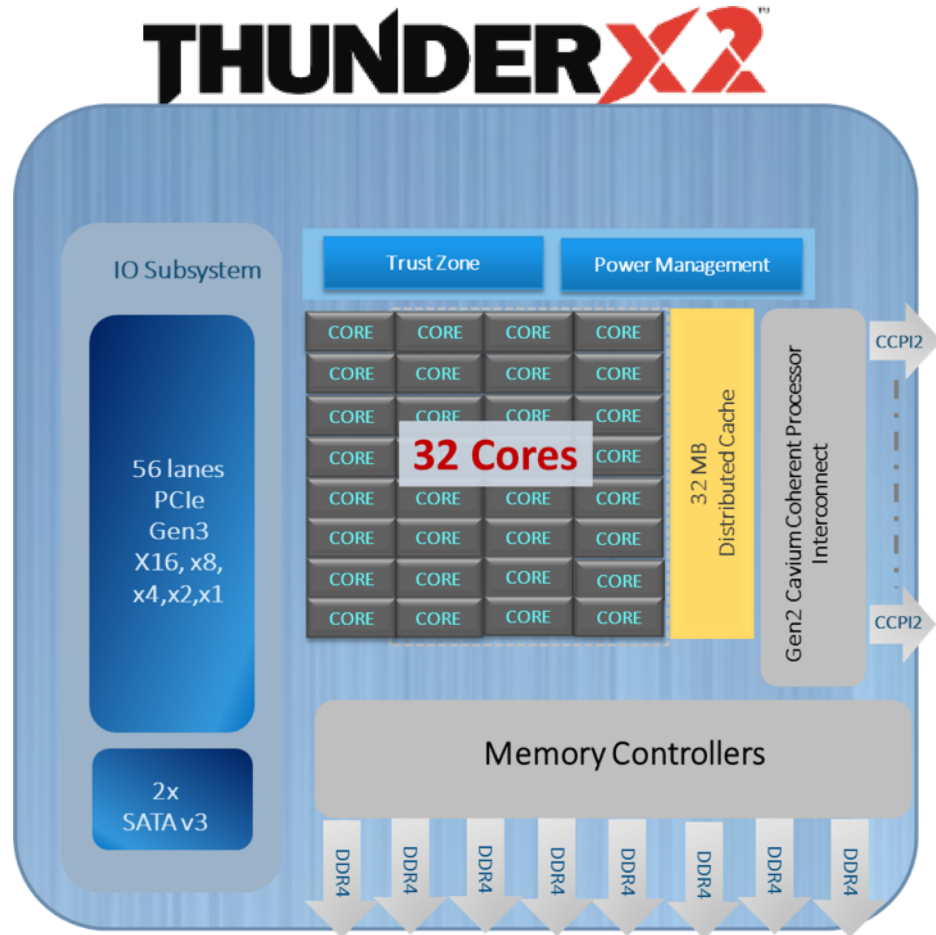
- Workflows leveraging containers and virtual machines
 - Support for machine learning frameworks
 - ARMv8.1 includes new virtualization extensions, SR-IOV
- Evaluating parallel filesystems + I/O systems @ scale
 - GlusterFS, Ceph, BeeGFS, Sandia Data Warehouse, ...
- Resilience studies over Astra lifetime
- Improved MPI thread support, matching acceleration
- OS optimizations for HPC @ scale
 - Exploring spectrum from stock distro Linux kernel to HPC-tuned Linux kernels to non-Linux lightweight kernels and multi-kernels
 - Arm-specific optimizations





Marvell Thunder X2

ThunderX2 - Second Generation High-End Armv8-A Server SoC



Up to 32 custom Armv8.1 cores, up to 2.5GHz

Full OoO, 1, 2, 4 threads per core

1S and 2S Configuration

Up to 8 DDR4-2667 Memory Controllers, 1 & 2 DPC

Up to 56 lanes of PCIe, 14 PCIe controllers

Full SoC: Integrated SATAv3 USB3 and GPIOs

Server class RAS & Virtualization

Extensive Power Management

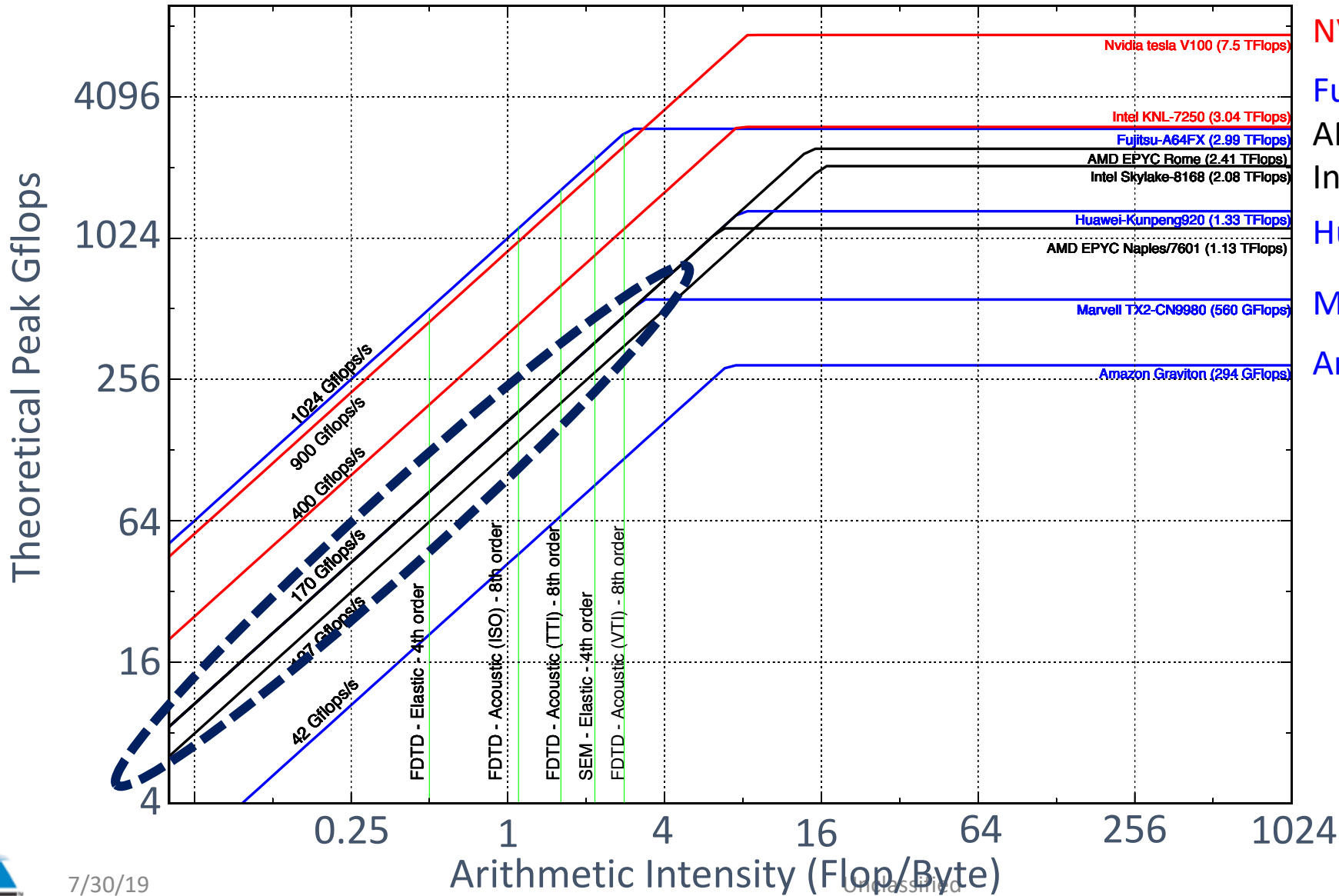
LGA and BGA for most flexibility

40+ SKUs (75W – 180W)

ThunderX2 Comparison with Xeon Processors

	Marvell ThunderX2	Haswell E5-2698 v3	Broadwell E5-2695	Skylake Gold 6152
Cores/Socket	32 (max 4 HT)	16 (2 HT)	22 (2 HT)	22 (2 HT)
L1 Cache/Core	32KB I/D (8-way)	32KB I/D (8-way)	32KB I/D (8-way)	32KB I/D (8-way)
L2 Cache/Core	256KB (8-way)	256 KB (8-way)	256 KB (8-way)	1 MB (16-way)
L3 Cache/Socket	32 MB	40 MB	33 MB	30.25 MB
#Memory Channels/Socket	8 DDR4	4 DDR4	4 DDR4	6 DDR4
Base Clock Rate	2.2 GHz	2.3 GHz	2.2 GHz	2.1 GHz
Vector/SIMD Length	128b (NEON)	256b (AVX2)	256b (AVX2)	512b (AVX512)

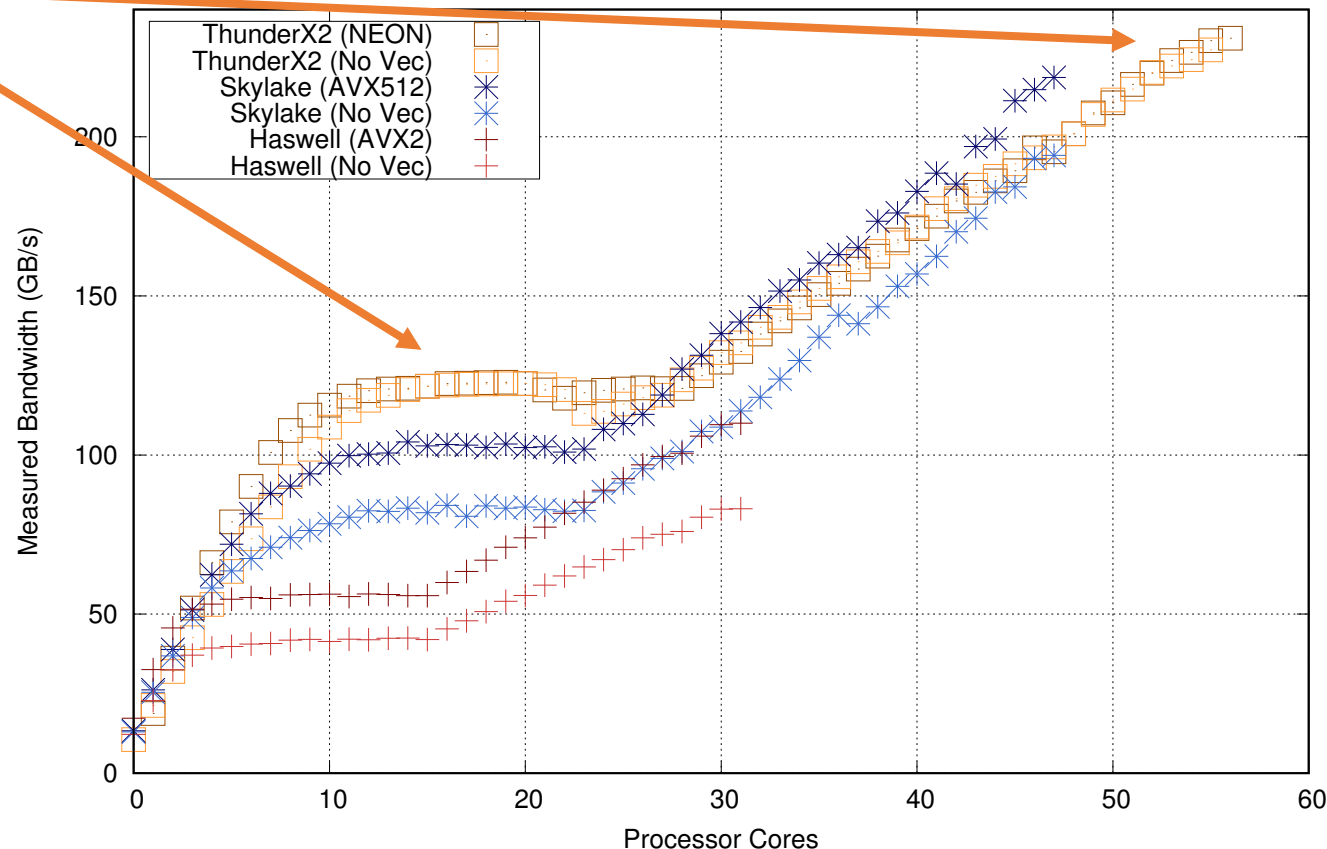
Roofline Comparison



- NVIDIA Tesla V100 (7.5 TFlops)
- Fujitsu A64FX (2.99 TFlops)
- AMD EPYC Rome (2.41 TFlops)
- Intel Skylake 8168 (2.08 TFlops)
- Huawei Kunpeng920 (1.13 TFlops)
- Marvell ThunderX2 (0.56 TFlops)
- Amazon Graviton (0.294 TFlops)

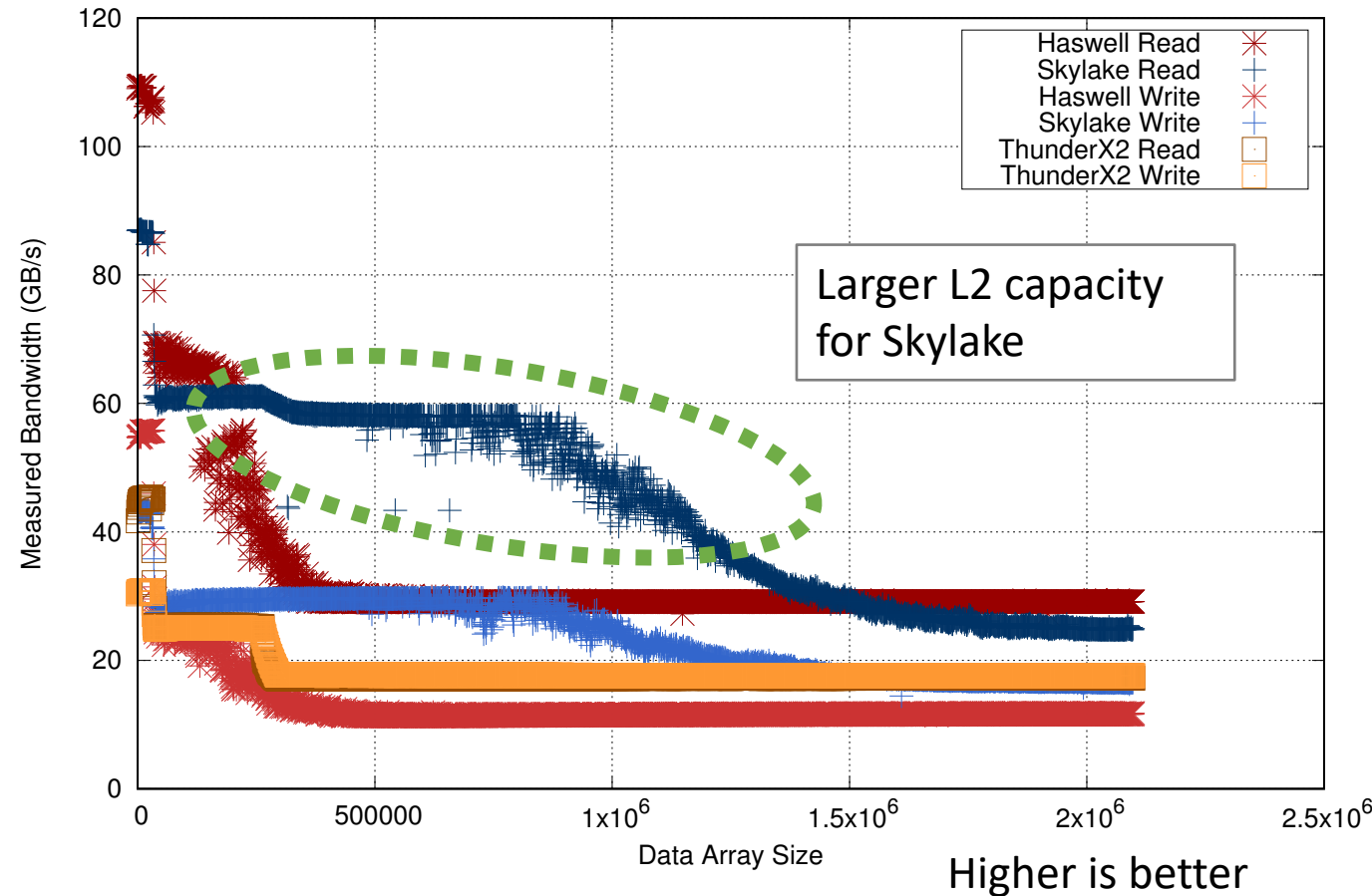
STREAM Triad Bandwidth

- ThunderX2 provides highest bandwidth of all processors
- Vectorization makes no discernable difference to performance at large core counts
 - Around 10% higher with NEON at smaller core counts (5 – 14)
- Significant number of kernels in HPC are bound by the rate at which they can load/store to memory (“memory bandwidth bound”)
 - Makes high memory bandwidth desirable
 - Ideally want to get to these bandwidths without needing to vectorize



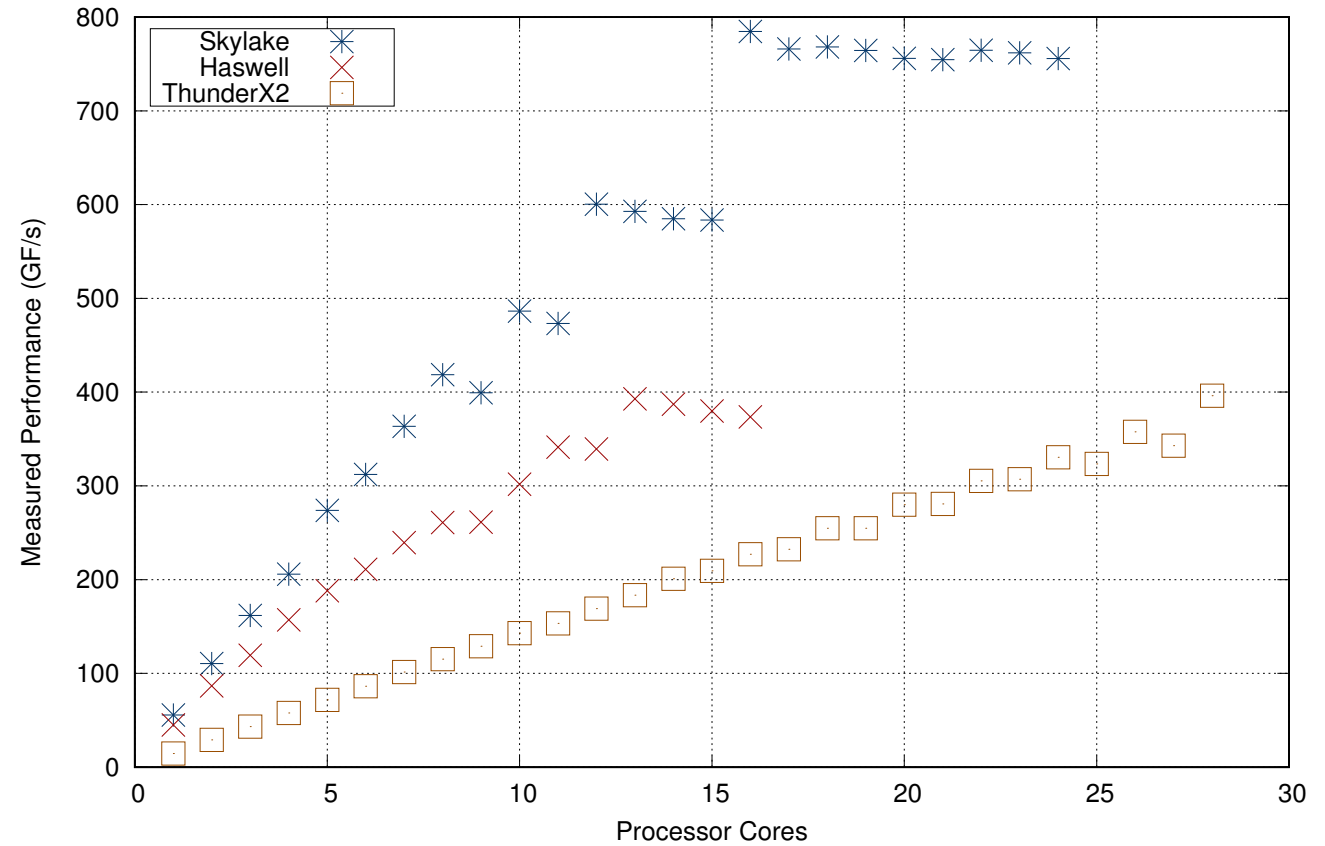
Cache Performance

- Haswell has highest per-core bandwidth (read and write) at L1, slower at L2.
- Skylake redesigned cache sizes (larger L2, smaller L3) shows up in graph
 - Higher performance for certain work-set sizes (typical for unstructured codes)
- TX2 more uniform bandwidth at larger scale (see less asymmetry between read/write)



DGEMM Compute Performance

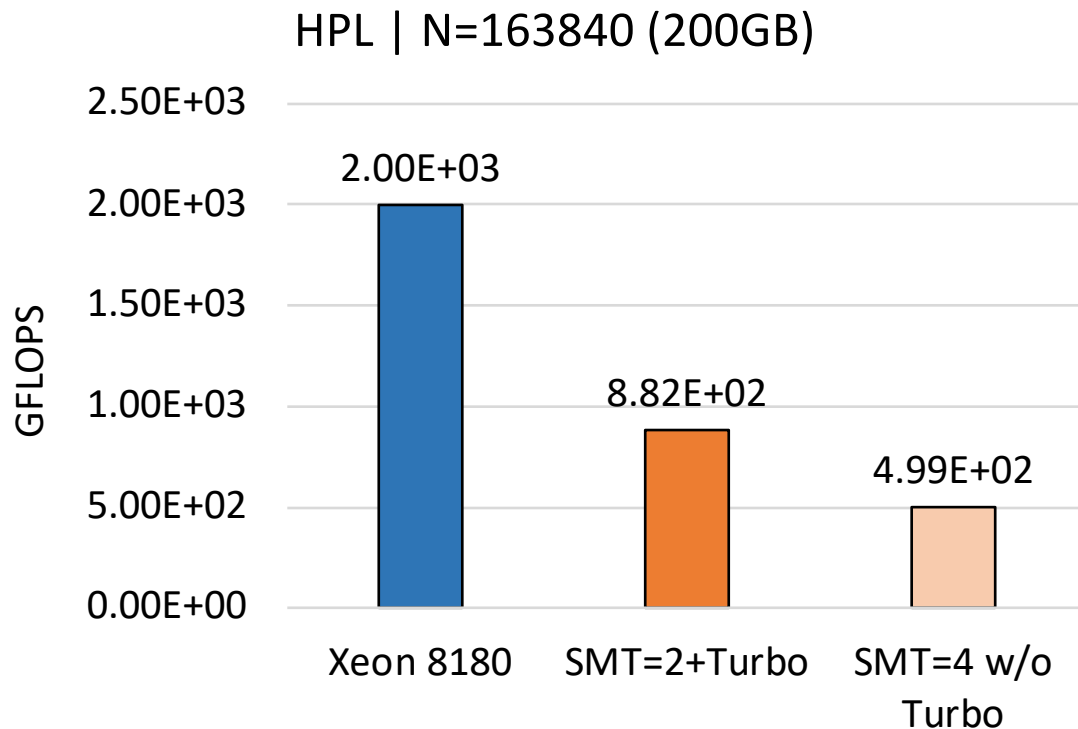
- ThunderX2 has similar performance at scale to Haswell
 - Roughly twice as many cores (TX2)
 - Half the vector width (TX2 vs. HSW)
- See strata in Intel MKL results, usually a result of matrix-size kernel optimization
 - ARM PL provides smoother performance results (essentially linear growth)



Higher is better

Floating Point Performance Sanity Check: HPL

- ThunderX2 has about half the floating point capacity of comparable Xeon CPUs
- Xeon 8180 vs. ThunderX2



- HPL.dat

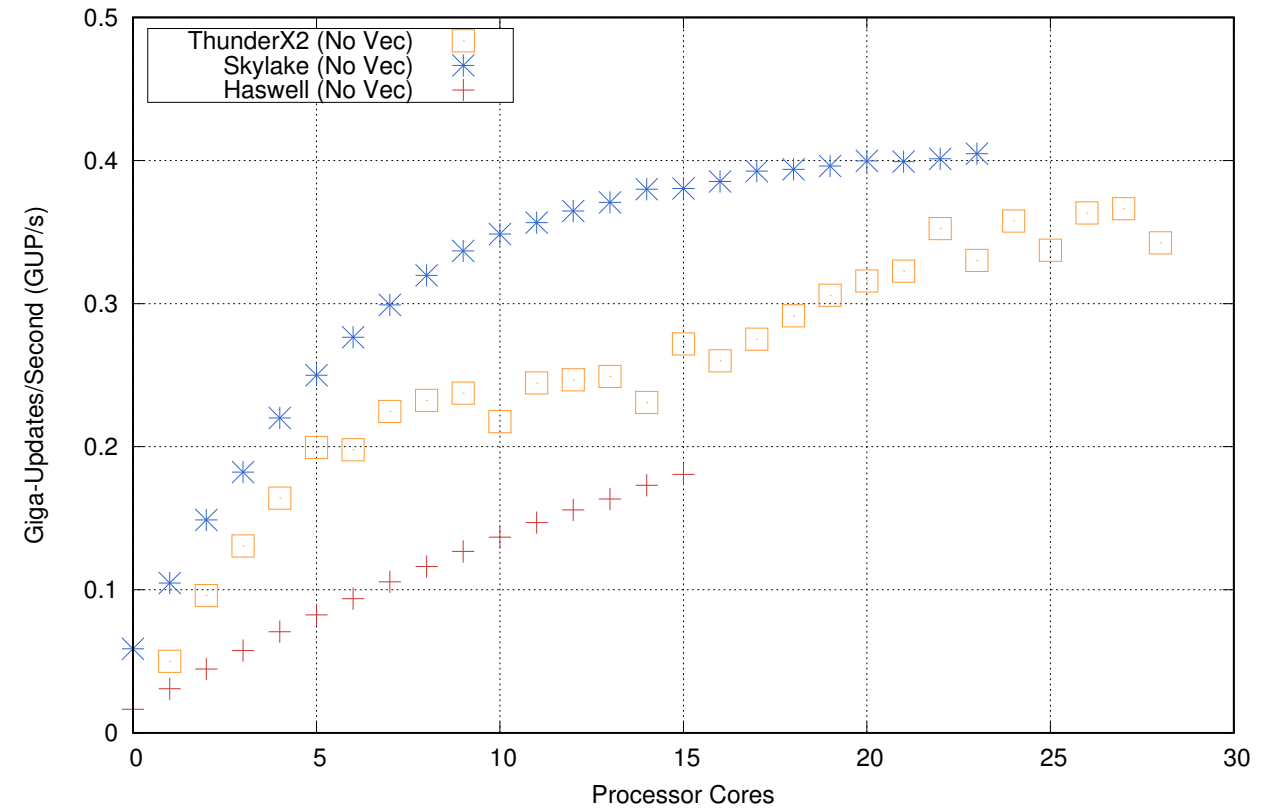
```
163840      Ns
256         NBs
0           PMAP process mapping (0=Row-,1=Column-
major)
7           Ps
8           Qs
1           PFACTs (0=left, 1=Crout, 2=Right)
2           RFACTs (0=left, 1=Crout, 2=Right)
0           BCASTs
(0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
0           DEPTHS (>=0)
2           SWAP (0=bin-exch,1=long,2=mix)
64          swapping threshold
```


Applications

Results from using Astra and other TX2 Platforms

GUPS Random Access

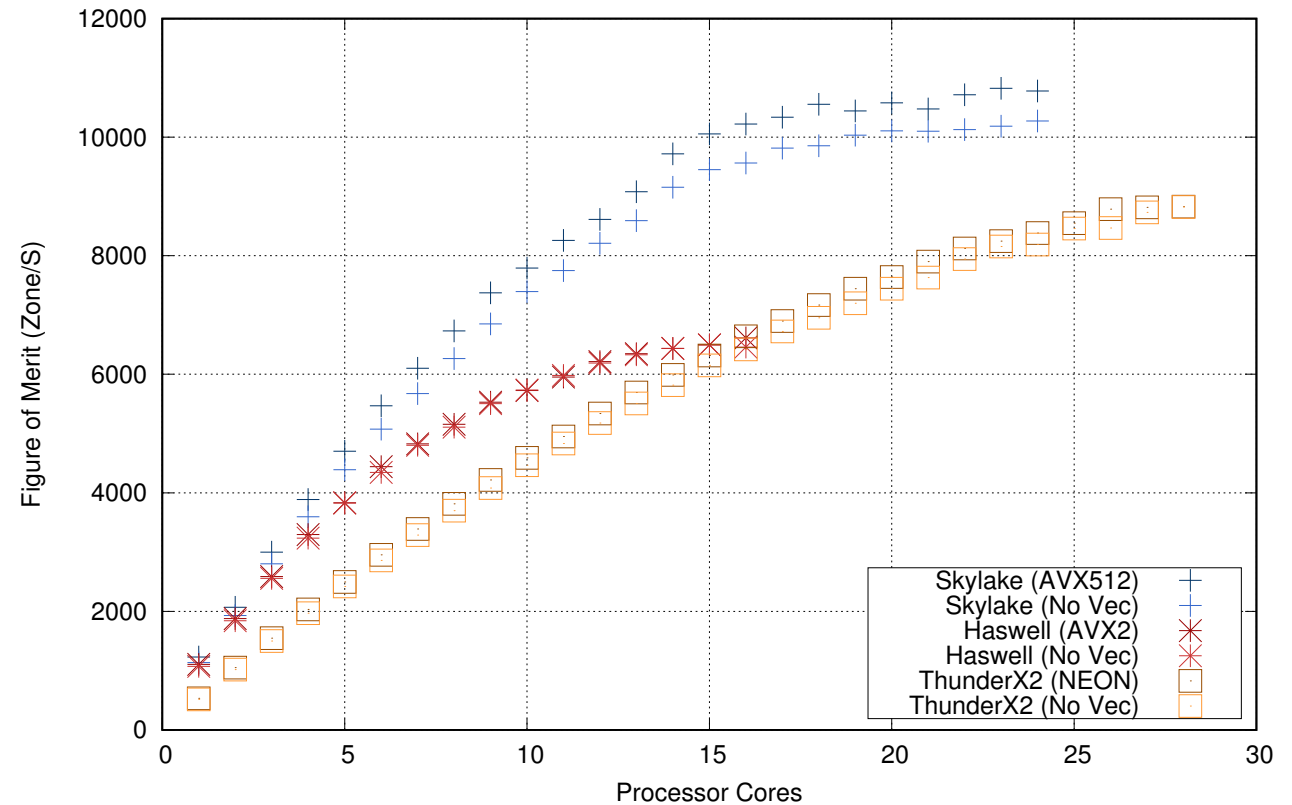
- Running all processors in SMT-1 mode, SMT(>1) is usually better performance
 - Expect SMT2/4 on TX2 to give better numbers
- Usually more cores gives higher performance (more load/store units driving requests).
 - Typical for TLB performance to be a limiter
 - Need to consider larger pages for future runs



Higher is better

LULESH Hydrodynamics Mini-App

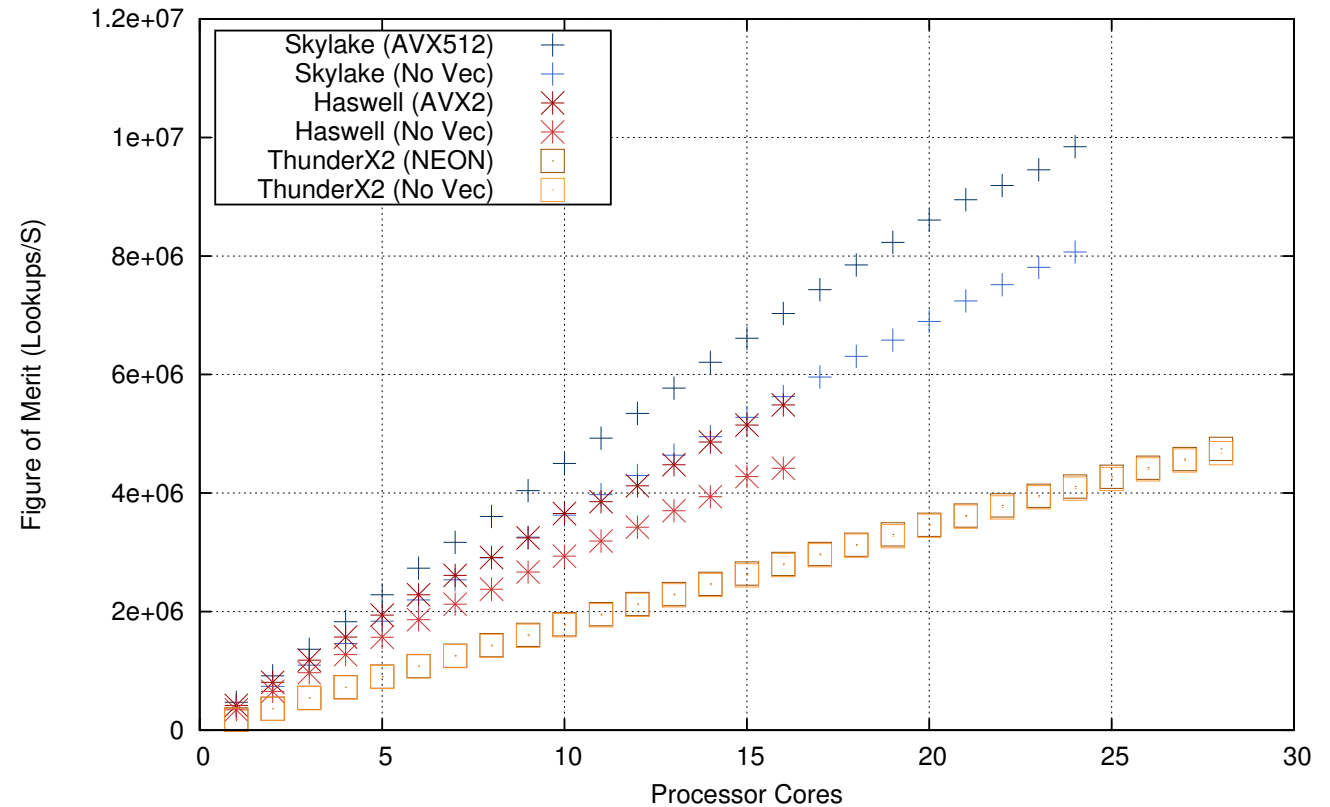
- Typically fairly intensive L2 accesses for unstructured mesh (although LULESH is regular structure in unstructured format)
- Expect slightly higher performance with SMT(>1) modes for all processors



Higher is better

XSbench Cross-Section Lookup Mini-App

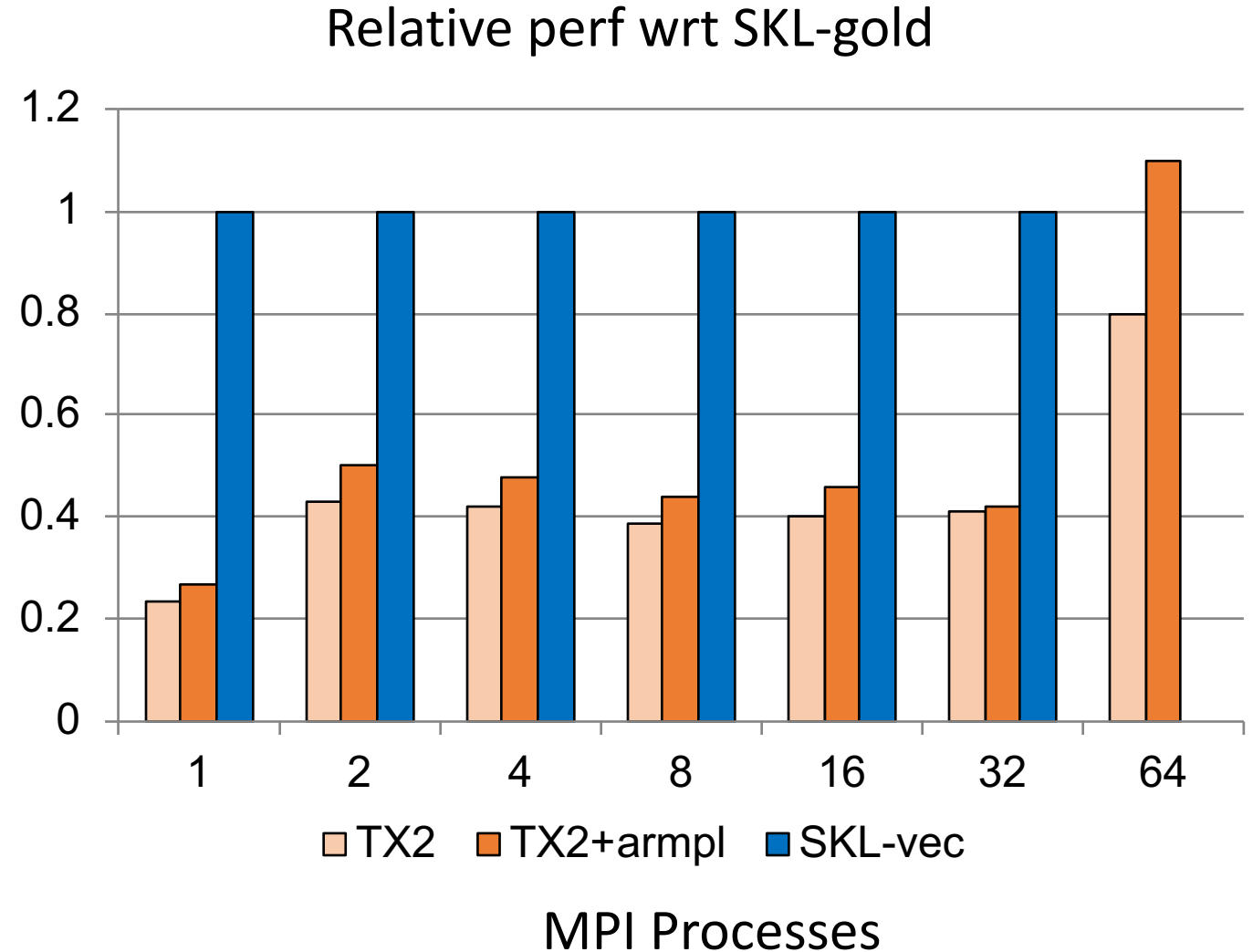
- Two level random-like access into memory, look-up in first table and then use indirection to reach second lookup
 - Means random access but is more like search so vectors can help
- See gain on Haswell and Skylake which both have vector-gather support
 - No support for gather in NEON
 - XSbench is mostly read-only (gather)



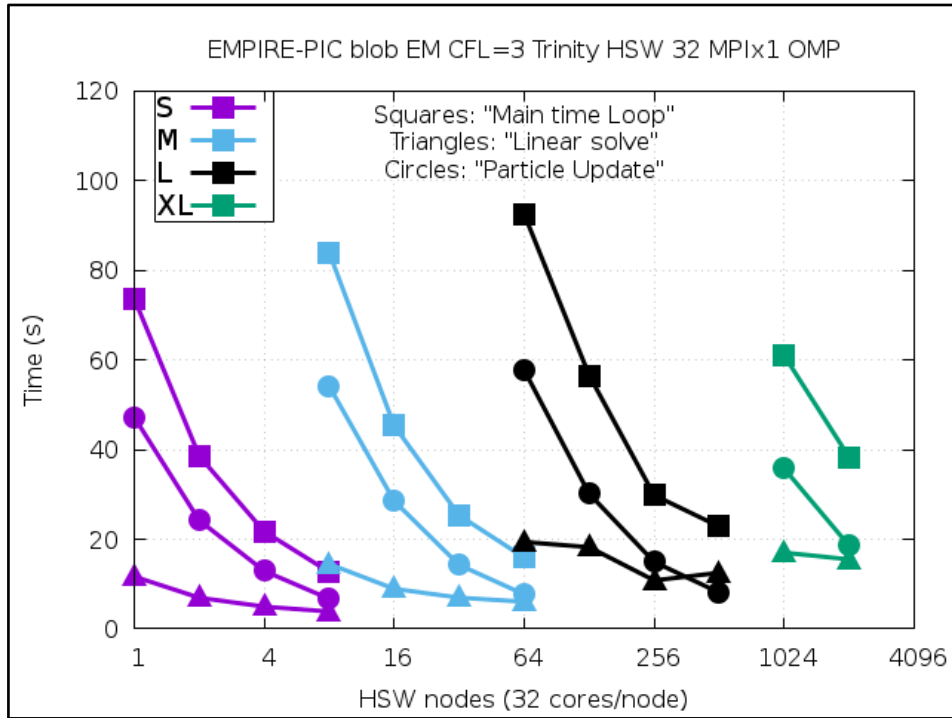
Higher is better

Branson Mini-App and Benchmark

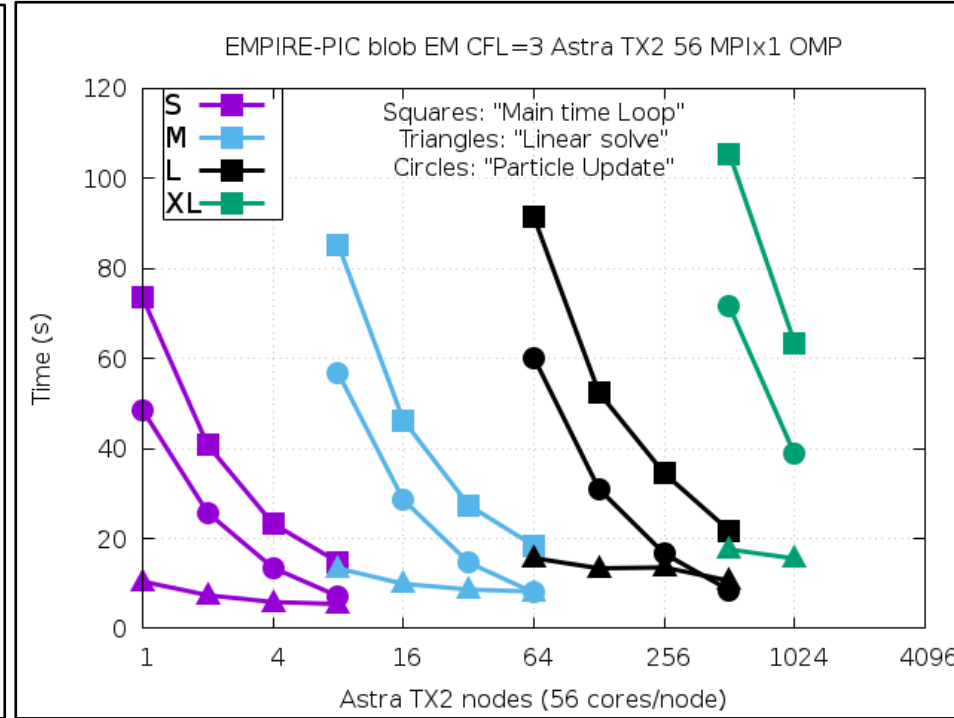
- Monte Carlo based Radiation transport mini-app
- Lots of time spent in math intrinsics (exp, log, sin, cos). Benefits from ARM optimized math intrinsics
- Poor memory locality, benefits some from large pages
- Doesn't vectorize
- Random number generator not yet optimized for ARM
- On a per node basis, TX2 is on par with SKL-gold
- Need to improve vectorizability



EMPIRE on Astra



Trinity HSW 32 MPI x 1 OMP



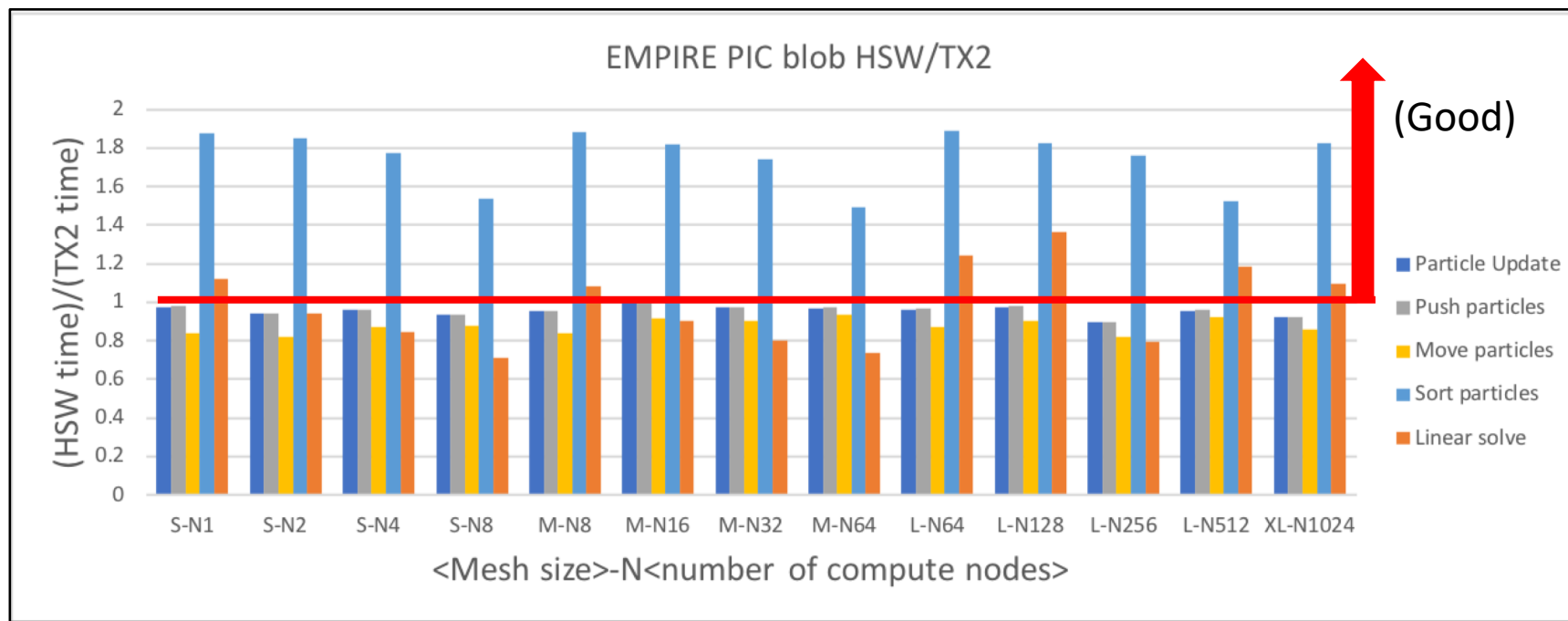
Astra TX2 56 MPI x 1 OMP

Strong and weak scaling studies for EMPIRE-PIC for awesome blob test case

Missing Trinity XL mesh 512 and 4096 node results because of MueLu FPE

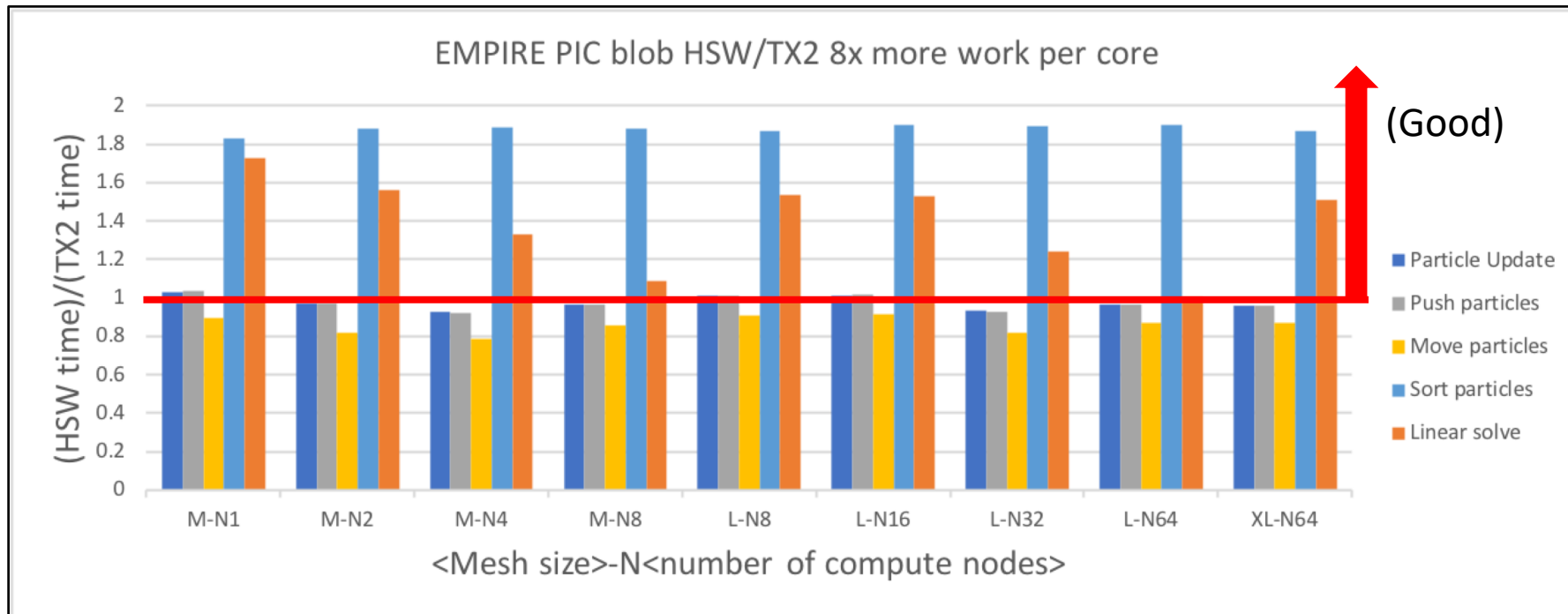
Missing Astra XL mesh 2048 node results because of MueLu FPE

EMPIRE on Astra



- TX2 node has ~2x memory bandwidth and 1.75x cores (56 vs. 32) of Trinity HSW node
- $(\text{HSW time})/(\text{TX2 time}) > 1$ means TX2 is faster
- Strong scaling for awesome blob small mesh (1-8 nodes), strong scaling for medium mesh (8-64 nodes), strong scaling for large mesh (64-512)
- $(\text{HSW time})/(\text{TX2 time})$ for linear solve not great, low computation/communication regime

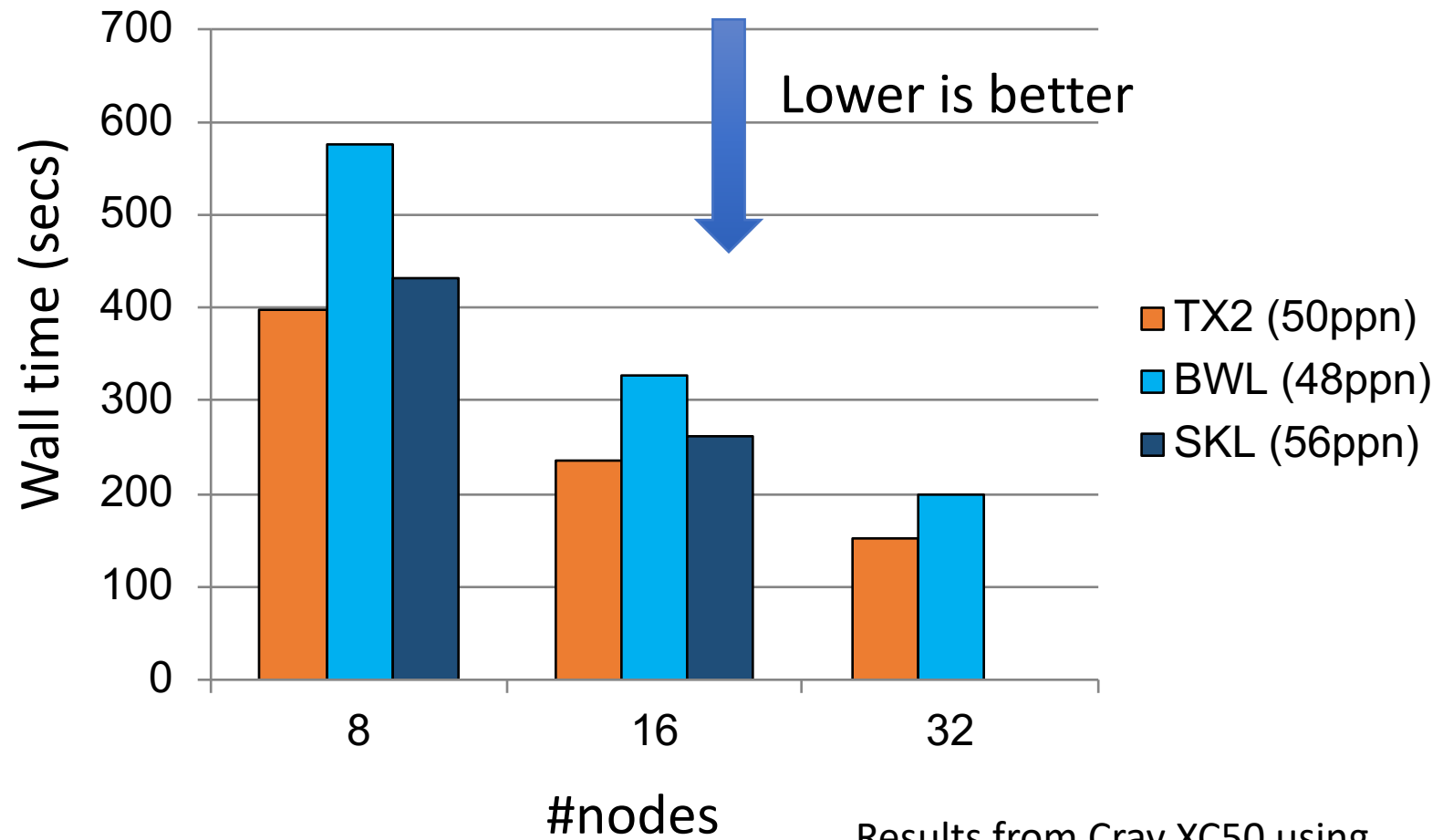
EMPIRE on Astra



- TX2 node has ~2x memory bandwidth and 1.75x cores (56 vs. 32) of Trinity HSW node
- $(\text{HSW time})/(\text{TX2 time}) > 1$ means TX2 is faster
- Strong scaling for awesome blob medium mesh (1-8 nodes), strong scaling for large mesh (8-64 nodes)
- $(\text{HSW time})/(\text{TX2 time})$ for linear solve definite better than previous slide, due to increased computation/communication

xRAGE

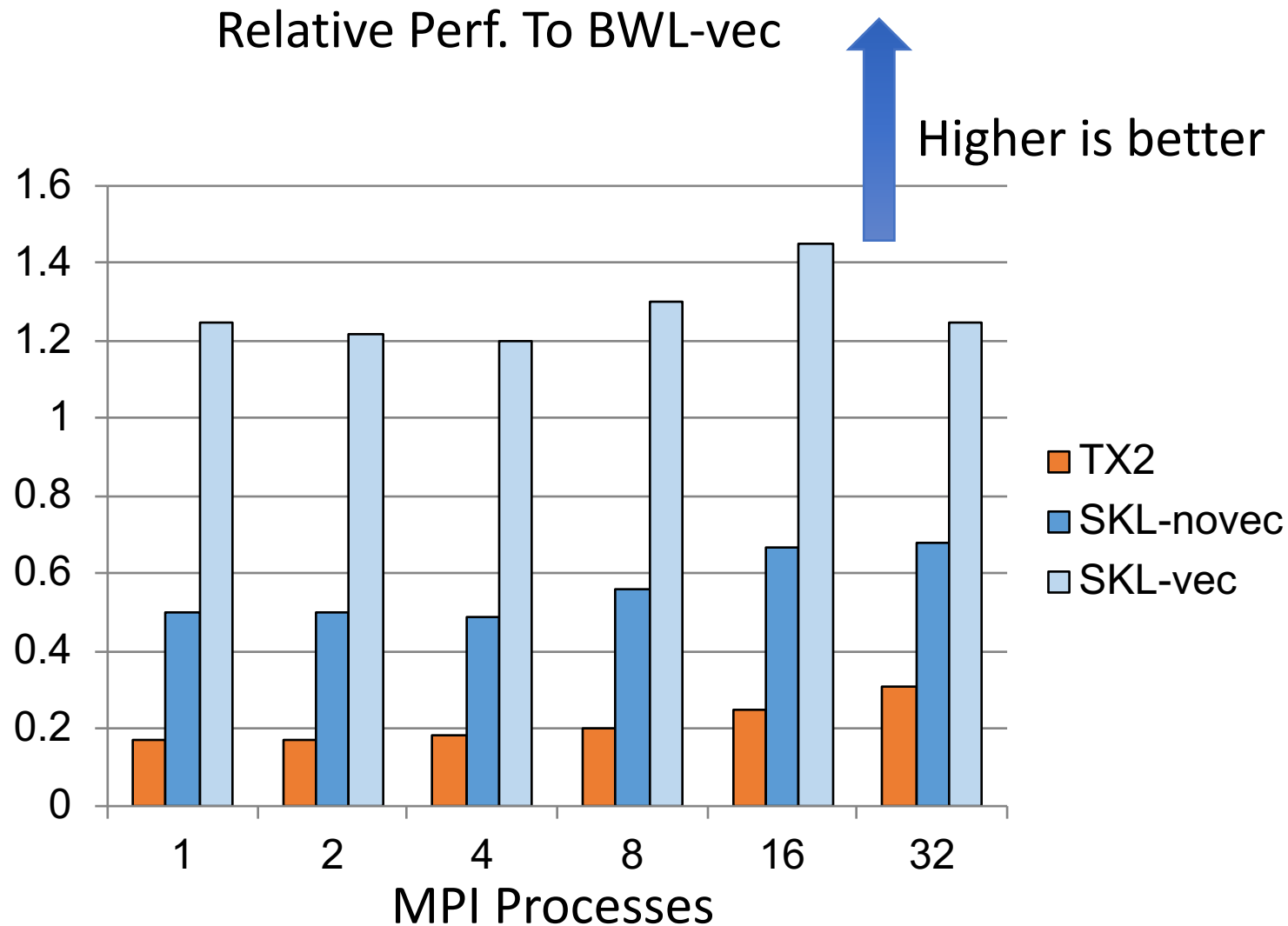
- Eulerian-based hydrodynamics/radiation transport application
- Uses adaptive mesh refinement
- Significant amount of gather/scatter
- Does not currently benefit from AVX2/512 vectorization
- Memory bound



Results from Cray XC50 using
Cray CCE9 Compiler

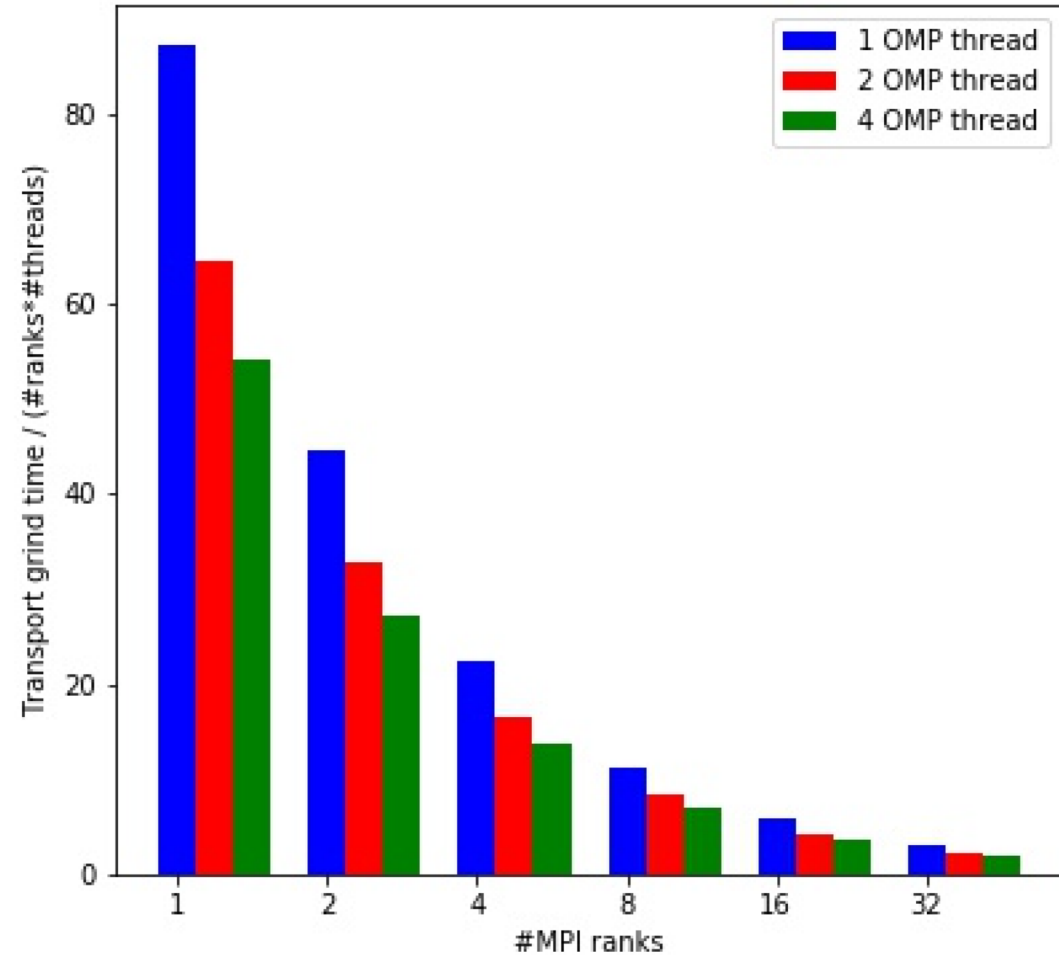
PARTISN

- Neutron transport code – deterministic SN method
- Sensitive to cache performance, not typically memory bound
- Vectorizes well for avx512, NEON
- Can be run mixed MPI/OpenMP
- Limited by cache BW on TX2 and front end stalls



PARTISN can benefit from 4 SMTs/core

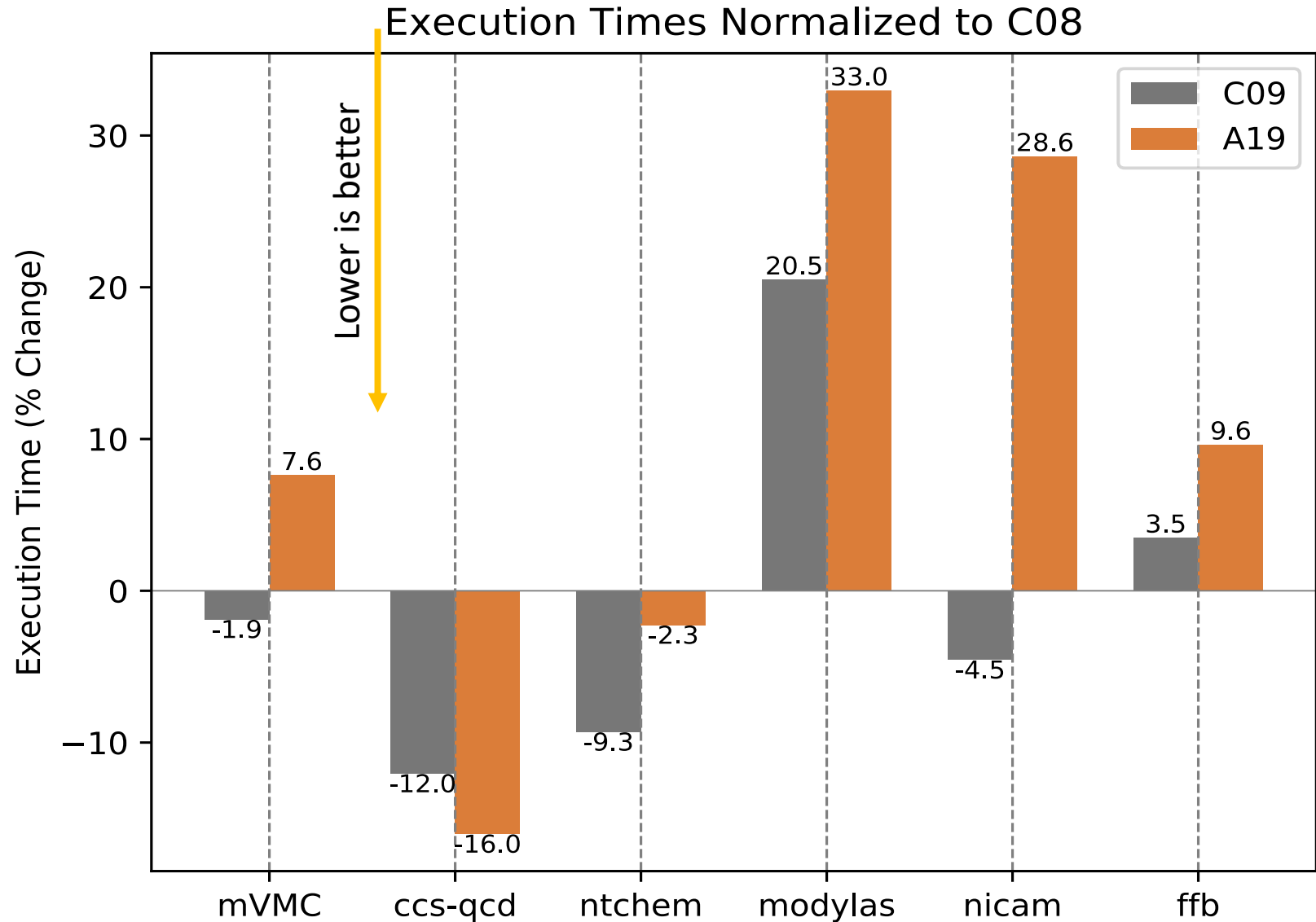
- Example of code with significant front end stalls
- Taucommander indicates high rate of branch misprediction in the sweep kernel



Cray XC50 - CCE 9.0 compiler

RIKEN Fiber Benchmarks – Compiler Performance Comparison

- Comparison of Cray 8/9 compilers against Allinea19 using Riken Fiber benchmarks
- Results are mixed, no clear winner in terms of compilers
- Takeaway is to try to build your app with several compilers

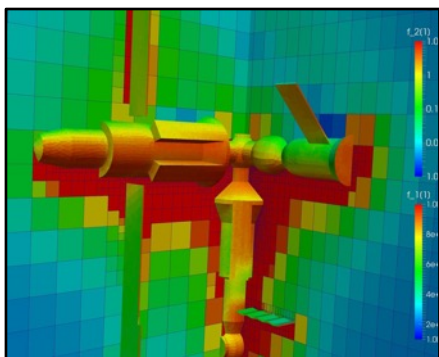


Unclassified

Early Results from Astra

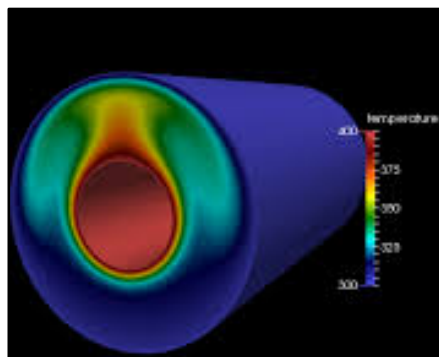
System has been online for around two weeks , incredible team working round the clock, already running full application ports and many of our key frameworks

Baseline: Trinity ASC Platform (Current Production), dual-socket Haswell



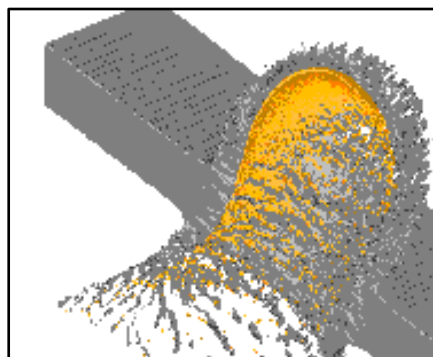
Monte Carlo

1.60X



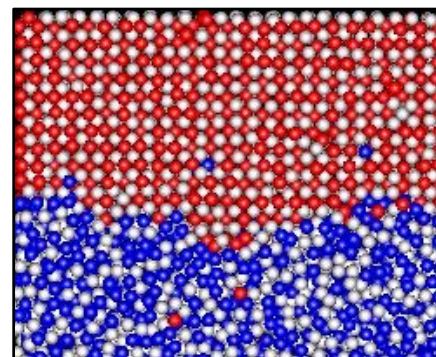
CFD Models

1.45X



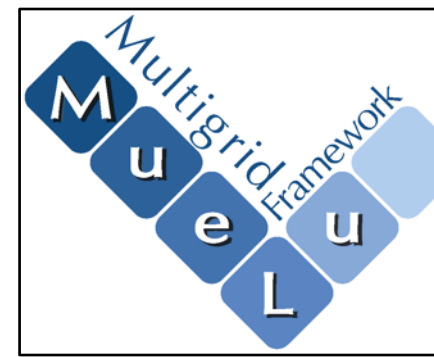
Hydrodynamics

1.30X



Molecular Dynamics

1.42X



Linear Solvers

1.87X



Porting to ARM

Sanity Checks

- See if your software has already been ported to aarch64:
 - www.gitlab.com/arm-hpc/packages/wikis
 - See if its available via Spack <https://github.com/spack/spack>
- Don't use old compilers:
 - GCC 8.2 or newer, 9.1 better
 - Allinea armflang/armclang 19.0 or newer
 - If you're package relies on some system packages in performance critical areas, may want to build your own versions. Libraries that come with base release are not optimized for Thunderx2
- If your application has lots of dependencies, this may be a good time to learn how to use Spack
- Checkout training material at <https://gitlab.com/arm-hpc/training>

Porting Cheat Sheet

Ensure all dependencies have been ported.

- Arm HPC Packages Wiki: <https://gitlab.com/arm-hpc/packages/wikis/categories/allPackages>

Update or patch autotools and libtool as needed

- `wget 'http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD' -O config.guess`
- `wget 'http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD' -O config.sub`
- `sed -i -e 's#wl=""#wl="-Wl,"#g' libtool`
- `sed -i -e 's#pic_flag=""#pic_flag=" -fPIC -DPIC"#g' libtool`

Update build system to use the right compiler and architecture

- Check `#ifdef` in Makefiles. Use other architectures as a template.

Use the right compiler flags

- Start with `-mcpu=native -Ofast`

Avoid non-standard compiler extensions and language features

- Arm compiler team is actively adding new “unique” features, but it’s best to stick to the standard.

Update hard-wired intrinsics for other architectures

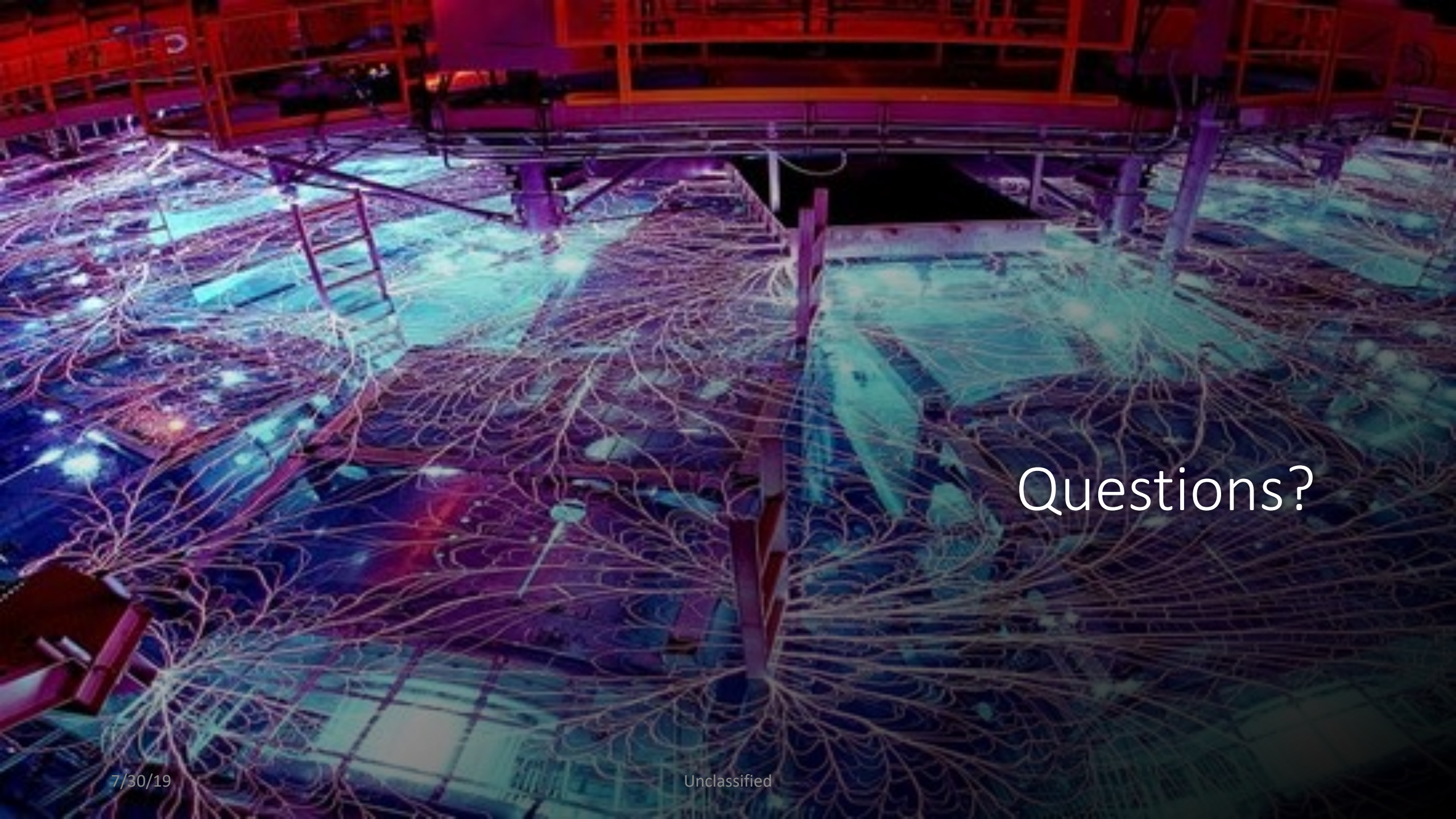
- <https://developer.arm.com/technologies/neon/intrinsics>
- Worst case: default to a slow code.

Update, and possibly fix, your test suite

- Regression tests are a porter’s best friend.
- Beware of tests that expect exactly the same answer on all architectures!

Know architectural features and what they mean for your code

- Arm’s weak memory model.
- Division by zero is silently zero on Arm.



Questions?