Dr Tom Deakin

Senior Research Associate

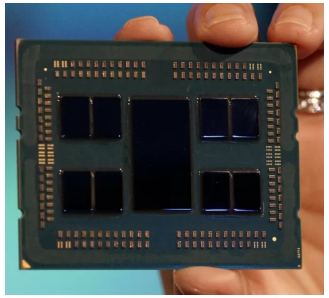
HPC research group

University of Bristol

Performance Portability across Diverse Computer Architectures

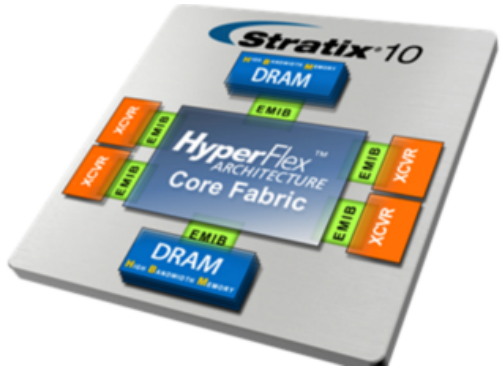
Recent processor trends in HPC

Many-core CPUs



GPUs/accelerators

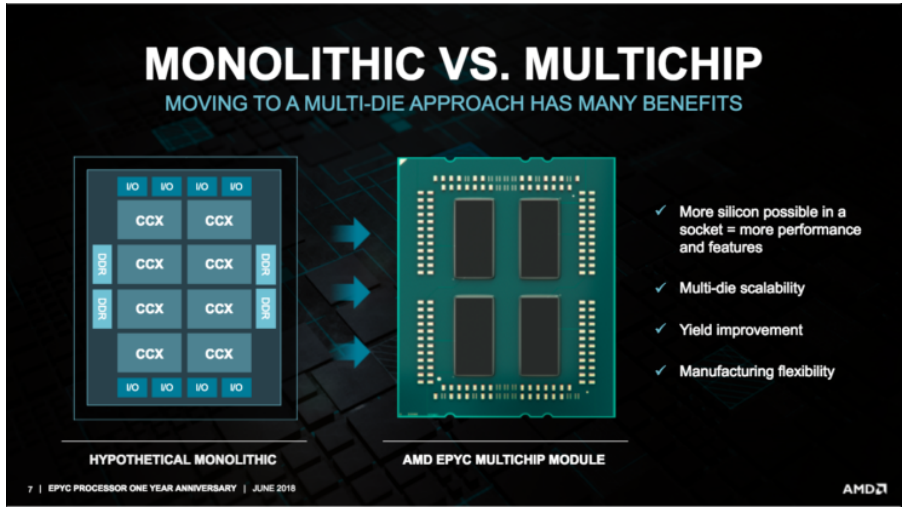
FPGAs

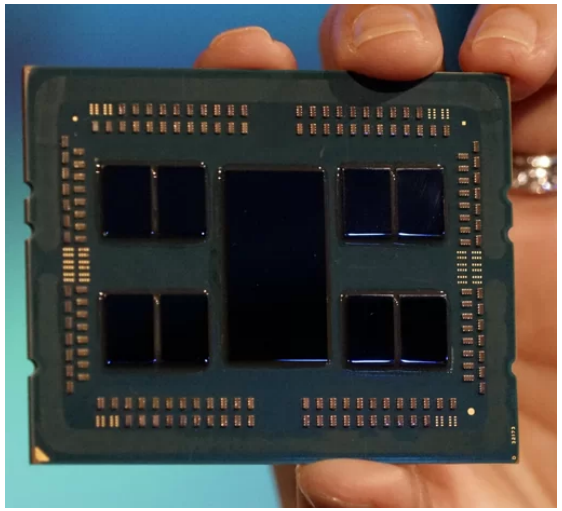


AMD's Rome showing where mainstream CPUs are heading

From late 2019:

- Up to 64 heavyweight x86 cores per CPU
- Uses 8 chiplets of 8 cores each, plus an I/O chiplet





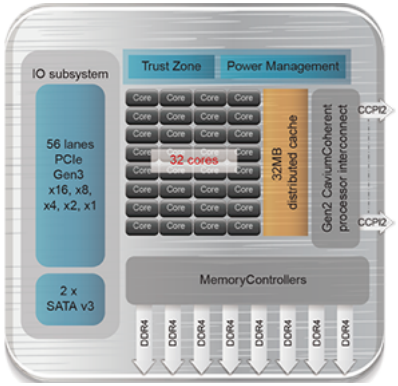
Chiplets likely to be an important future trend…

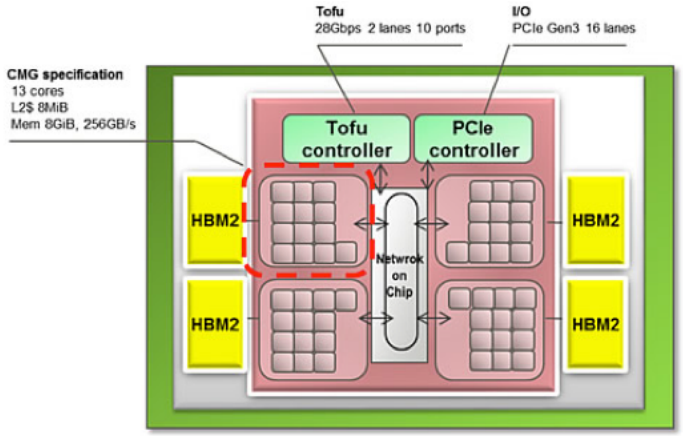
Emerging competition from Arm CPU vendors

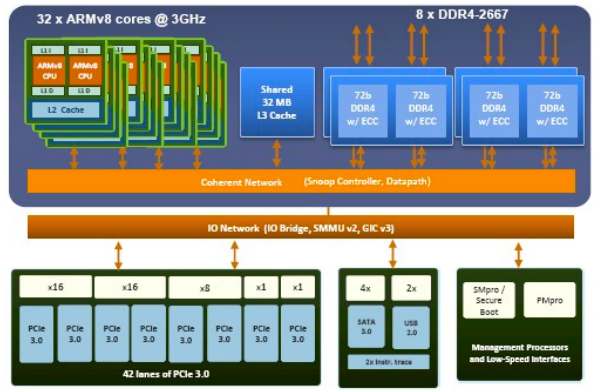
CAVIUM

FUJITSU

AMPERE™







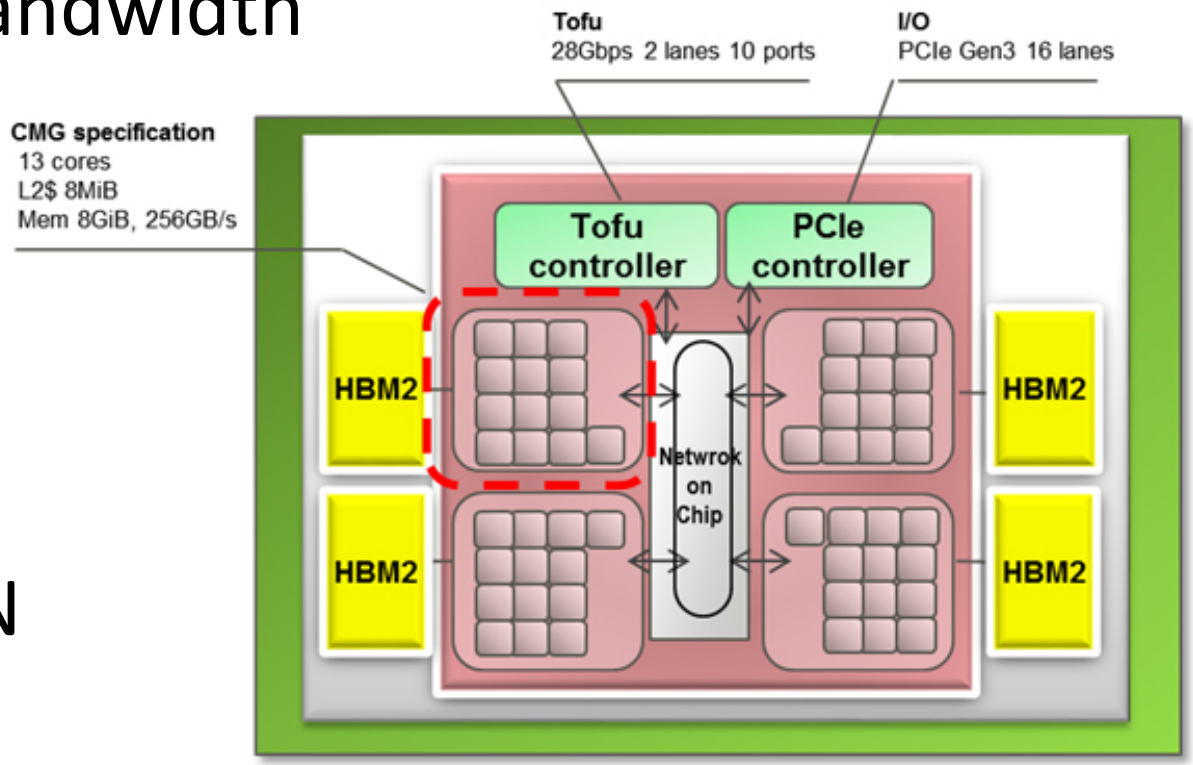
University of BRISTOL

http://uob-hpc.github.io

GW4

An example forthcoming Arm-based CPU: Fujitsu's A64FX

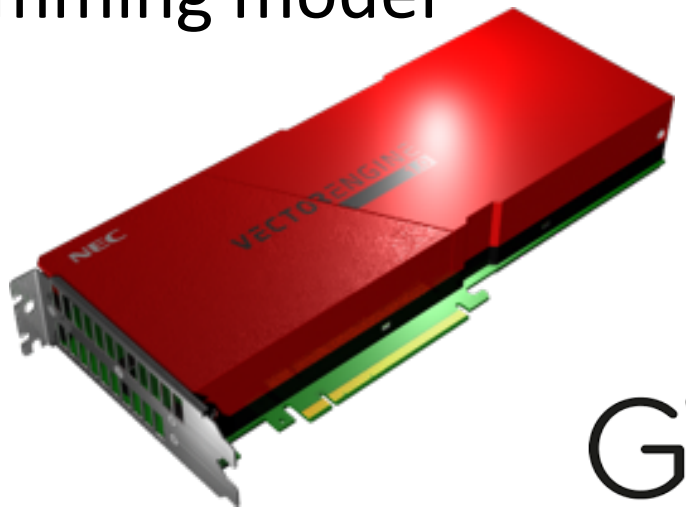
- 48 cores, no hyperthreading
- 2.7 TFLOP/s double precision, 512-bit vectors (SVE)
- 1 TeraByte/s main memory bandwidth
  - 32 GB HBM2
- ~170 Watts
- High speed interconnect
- 8.7B transistors, 7nm
- Fugaku now installed at RIKEN
  - 158,976 A64FX processors



http://www.fujitsu.com/global/about/resources/news/press-releases/2018/0822-02.html

NEC Aurora Vector Engine sit at technology intersection

- Relatively few cores (8) compared to CPUs (64) and GPUs (80)
  - More cores increase FLOPs, but also increase aggregate cache size and bandwidth
- Hierarchy of large caches like a CPU
- Wide vectors like a GPU (32 VPUs / 256 FP64 vectors)
- Uses HBM2 memory technology
- Accelerator form-factor, but traditional programming model
  - Reverse offload MPI+OpenMP
  - Standard offload an option too



Recent architectural trends

- CPUs have evolved to include **lots of cores** and **wide vector units**
- 32 core CPUs now common (AMD Naples, Marvell ThunderX2)
- 48, 64 core CPUs arrive within the next 12 months (A64fx, Rome)
- Chiplet manufacturing processes likely to be an important future trend
- This **renewed competition in CPUs** is crucial to the health of the HPC ecosystem, and for performance per dollar

- GPUs incorporating latest memory technologies (HBM)
  - So do KNL and A64FX CPUs
- GPUs have **lots of cores** and very **wide vector units**
- Lightweight cores becoming more complex (caches, specialised accelerators, etc)
- Vendor competition increasing (AMD GPUs in Frontier, Intel GPUs in Aurora, NVIDIA GPUs pre-Exascale Perlmutter)

Isambard system specification

- **10,752** Armv8 cores (168n x 2s x 32c)
  - **Marvell ThunderX2 32 core 2.1→2.5GHz**
  - **256 GB RAM per node, bandwidth >240 GB/s**
- 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):**
  - Installed in November 2018, accepted in 1 week!
  - Upgraded silicon (to B2), firmware and stack Mar19
  - As of June (since increased):
    185 registered users, 63 are external
  - **PRODUCTION SERVICE opened to all users May 2019**
    - **First Arm-based production service in the world!**

University of BRISTOL

http://gw4.ac.uk/isambard/

GW4

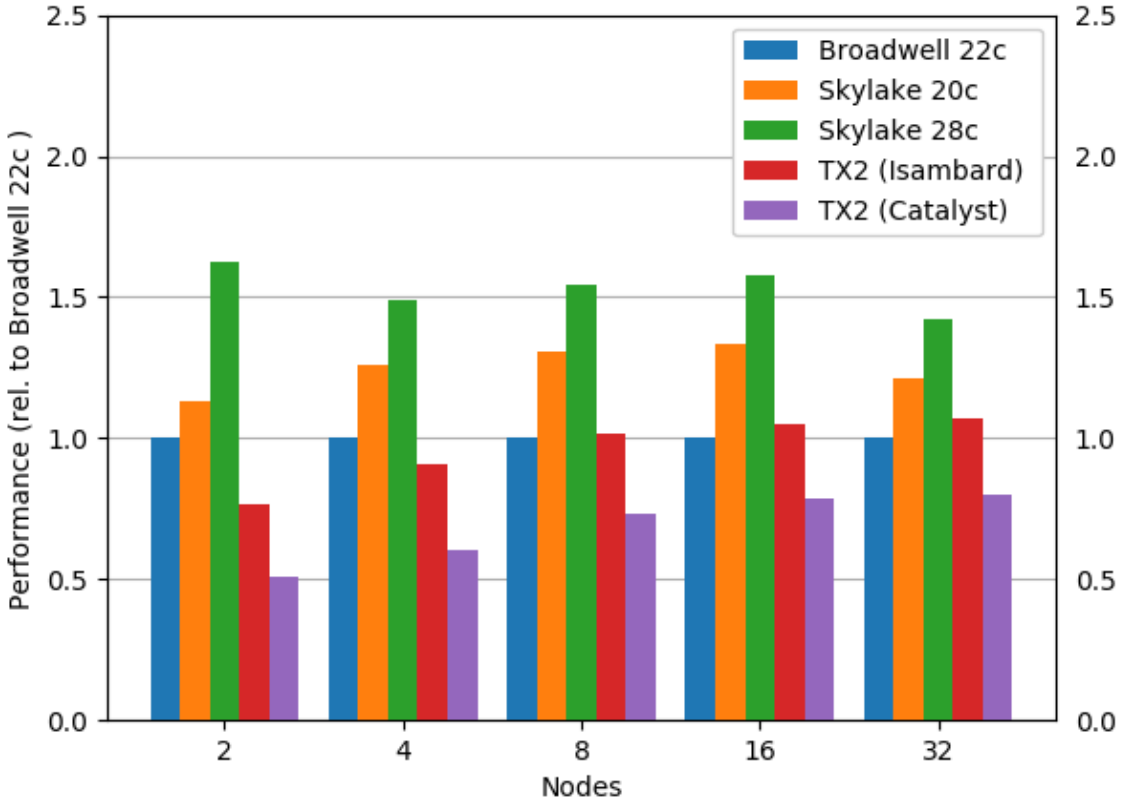
Comparing between multiple Arm-based supercomputers

- **Bristol** is one of the few sites in the world with multiple different Arm-based supercomputers
  - Added an HPE Catalyst system in 1Q2019
    - 64-node Apollo70 system, 4,096 cores, ThunderX2 CPUs

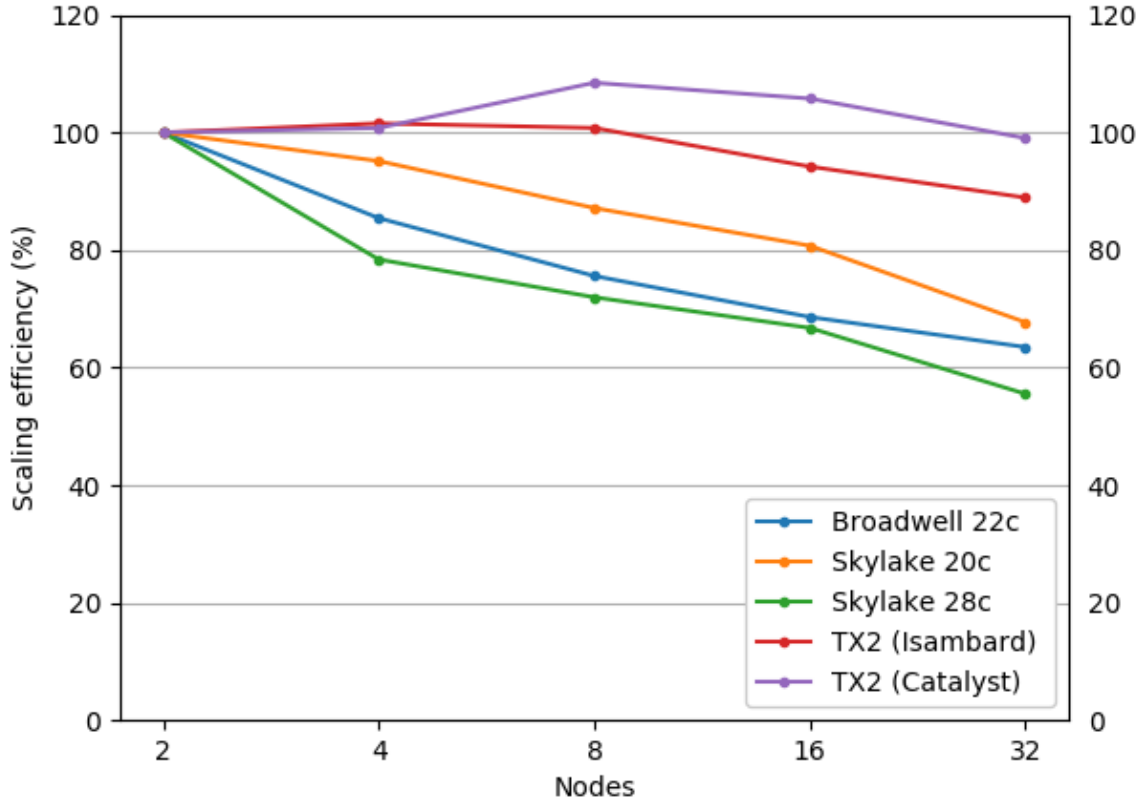
- **Isambard** and **Catalyst** together enable us to compare across:
  - **Networks**: Cray Aries vs Mellanox IB
  - **Software stacks**: open source vs Cray

- For comparisons see following references:
  - Talk from AHUG @ISC'19
  - https://doi.org/10.1002/cpe.5110

GROMACS (42 million atoms, ARCHER benchmark)



Relative performance



Parallel efficiency

University of BRISTOL

Scaling Results From the First Generation of Arm-based Supercomputers
S. McIntosh-Smith, J. Price, A. Poenaru and T. Deakin, CUG 2019, Montreal

GW4

Isambard 2 Tier-2 service designed to explore these opportunities

Diverse range of architectures:

- CPUs:
  - Arm: Fujitsu, Marvell
  - X86: AMD, Intel
  - IBM POWER
- GPUs:
  - NVIDIA
  - AMD
  - Intel

GW4

Met Office

CRAY THE SUPERCOMPUTER COMPANY

EPSRC

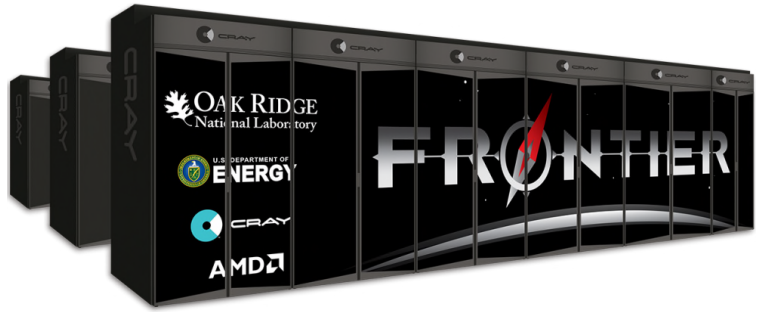
University of BRISTOL

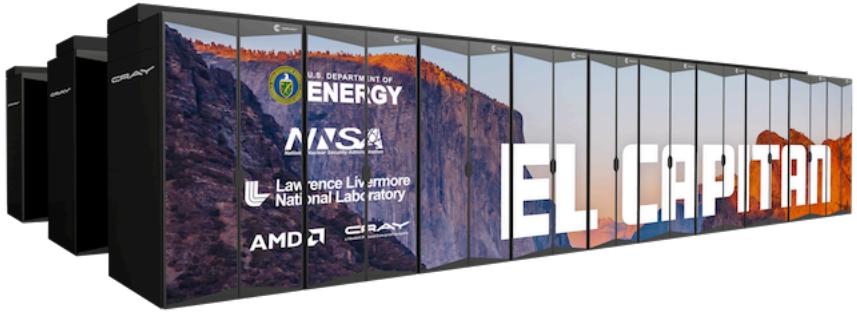
http://uob-hpc.github.io

GW4

Challenges at Exascale

- The coming generation of Exascale supercomputers will contain a diverse range of architectures at massive scale
  - **Perlmutter**: AMD EYPC CPUs and NVIDIA GPUs (pre-Exascale)
  - **Frontier**: AMD EPYC CPUs and Radeon GPUs
  - **Aurora**: Intel Xeon CPUs and Xe GPUs
  - **El Capitan**: AMD EPYC CPUs and Radeon GPUs
  - **Fugaku**: Fujitsu A64fx Arm CPUs





The Next Platform, Jan 13th 2020: "HPC in 2020: compute engine diversity gets real"
https://www.nextplatform.com/2020/01/13/hpc-in-2020-compute-engine-diversity-gets-real/

https://uob-hpc.github.io/publications/

University of Bristol

What do we mean by "performance portability?"

"A code is performance portable if it can achieve a similar fraction of peak hardware performance on a range of different target architectures."

Questions:

- **Does it have to be a "good" fraction?** YES! Within 20% of "best achievable", i.e. of hand-optimized OpenMP, CUDA, …
- **How wide is the range of target architectures?** Depends on your goal, but important to allow for future architectural developments

S.J. Pennycook, J.D. Sewall, V.W. Lee, A metric for performance portability,in:Proceedings of the International Workshop on Performance Modeling,BenchmarkingandSimulation,2016.URLhttp://arxiv.org/abs/1611.07409.

A Metric for Performance Portability

S. J. Pennycook, J. D. Sewall and V. W. Lee
Intel Corporation
Santa Clara, California
{john.pennycook,jason.sewall,victor.w.lee}@intel.com

***Abstract*—The term "performance portability" has been informally used in computing to refer to a variety of notions which generally include: 1) the ability to run one application across multiple hardware platforms; and 2) achieving some notional level of performance on these platforms. However, there has been a noticeable lack of consensus on the precise meaning of the term, and authors' conclusions regarding their success (or failure) to achieve performance portability have thus been subjective. Comparing one approach to performance portability with another has generally been marked with vague claims and verbose, qualitative explanation of the comparison. This paper presents a concise definition for performance portability, along with a simple metric that accurately captures the performance and portability of an application across different platforms. The utility of this metric is then demonstrated with a retroactive application to previous work.**

and demonstrate its accuracy and utility for quantifying an application's performance *and* portability; and

3) We retroactively apply our metric to a number of published application studies, thereby highlighting the utility of a shared metric when comparing and contrasting different approaches to performance portability.

II. Related Work

There have been a number of efforts to develop new programming models, languages and tools that provide users with a productive means of achieving performance portability. Some have proposed the use of domain-specific languages (DSLs), providing a limited set of high-level abstractions for a spe-

Two ways to measure Performance Portability

Definitions from the Pennycook, Sewall and Lee paper:

1. **Architectural efficiency**:
   Achieved performance as a ***fraction of peak theoretical hardware performance***. This represents the ability of an application to utilize hardware efficiently;

2. **Application efficiency**:
   Achieved performance as a ***fraction of best observed performance***. This represents the ability of an application to use the most appropriate implementation and algorithm for each platform

A systematic evaluation of Performance Portability

- Studying Performance Portability is ***hard***!
  - Have to be **rigorous** about doing as well as possible across a wide range issues: architectures, programming languages, algorithms, compilers, …
- It takes a lot of effort to do this well
- Motivated by our results so far, in Bristol we have initiated a wide-ranging evaluation of Performance Portability:
  - Across many codes
  - Across many programming languages
  - Across many architectures
- Our goal is to share these codes and results to further the fundamental understanding of performance portability

Codes in the Bristol Performance Portability study

BabelStream:	simple measure of achievable memory bandwidth
CloverLeaf:	structured grid hydrodynamics
TeaLeaf:	structured grid heat diffusion
Neutral:	Monte Carlo neutral particle transport
MiniFMM:	fast multipole method
SNAP*:	structured grid deterministic neutral particle transport
unSNAP*:	unstructured grid deterministic neutral particle transport
MG-CFD*:	unstructured grid CFD
Mini-PRECISE:	combustion code

* = work in progress

Parallel programming languages in the Bristol PP study

- OpenMP
- OpenMP target
- Kokkos CPU
- Kokkos GPU
- OpenACC
- CUDA
- OpenCL
- RAJA*
- SYCL*
- Flat MPI*

* = to come

Target hardware platforms

CPUs:

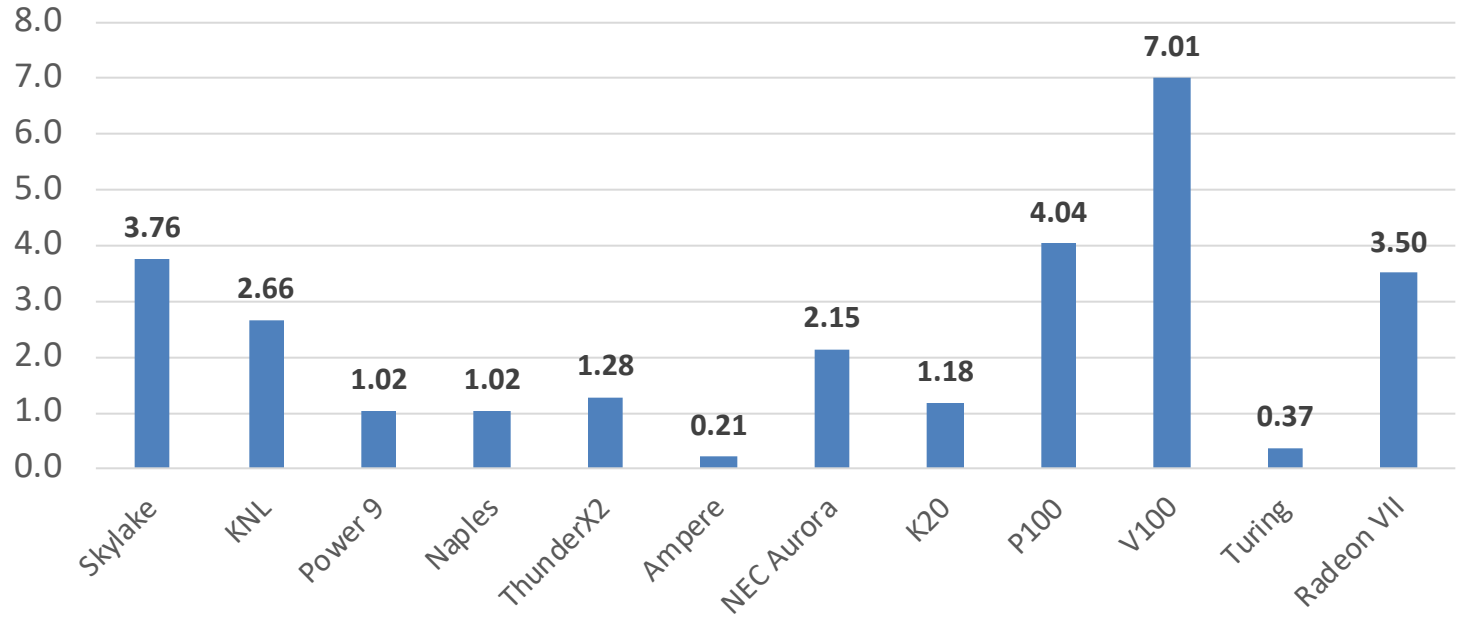
- Intel Skylake
- Intel KNL
- AMD Naples, Rome*
- IBM POWER9
- Marvell ThunderX2
- Marvell ThunderX3/4/5*
- Ampere eMAG
- Fujitsu A64fx*

Accelerators:

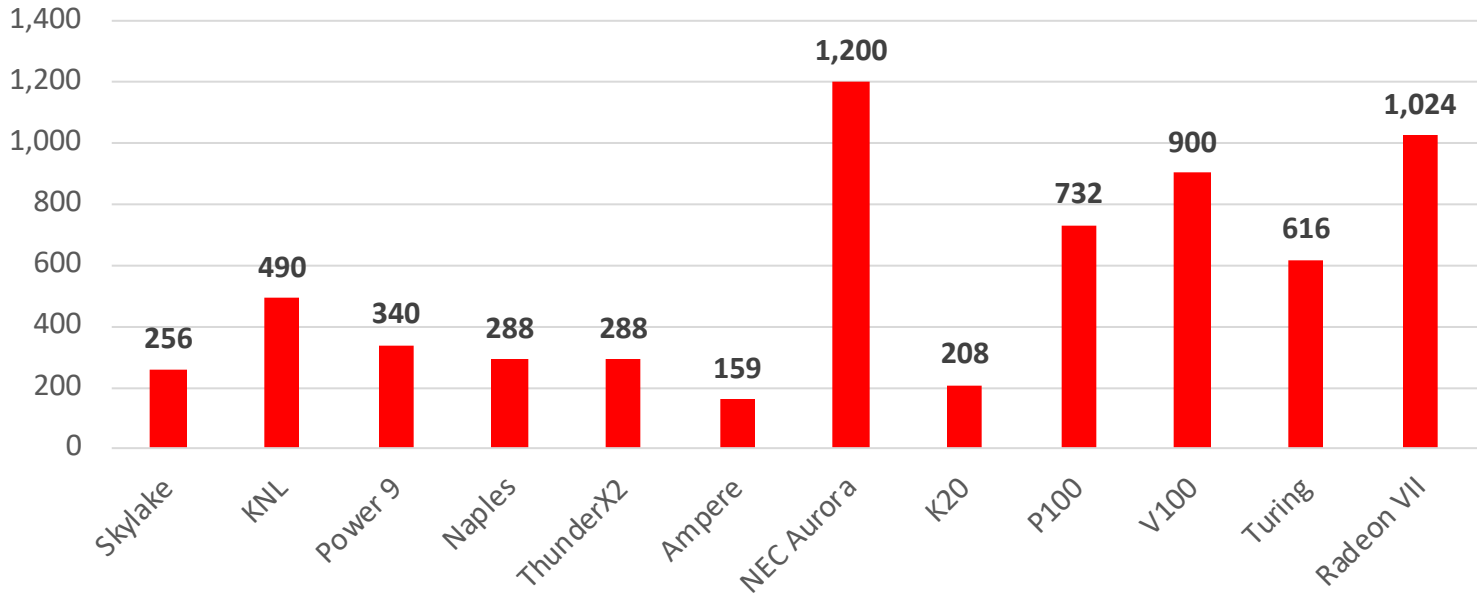
- NEC Aurora
- NVIDIA Turing
- NVIDIA Volta
- NVIDIA Pascal
- NVIDIA Kepler
- AMD Radeon VII
- FPGAs*

* = to come

Peak D.P. FLOP/s



Peak BW GB/s



University of BRISTOL

GW4

Bristol Performance Portability study

Latest results

BabelStream

- BabelStream benchmark written to measure achievable (main) memory bandwidth
- Based on McCalpin STREAM benchmark, but with a number of key differences:
  - Arrays allocated on the heap
  - Problem size known only at runtime
  - Range of programming models to widen support for CPUs and GPUs
- Constructed of simple vector operations:
  - c[i] = a[i]
  - b[i] = scalar * c[i]
  - c[i] = a[i] + b[i]
  - a[i] = b[i] + scalar * c[i]
  - sum += a[i] * b[i]

University of BRISTOL

https://github.com/UoB-HPC/BabelStream

GW4

BabelStream

Achieved bandwidth (GB/s)

Higher is better

	OpenMP	Kokkos	CUDA	OpenACC	OpenCL
Skylake	205	174	-	83.0	107
KNL	452	304	-	444	286
Power 9	248	250	-	247	-
Naples	190	181	-	-	-
ThunderX2	246	244	-	-	-
Ampere	106	91.1	-	-	-
NEC Aurora	976	-	-	-	-
K20	144	152	150	-	151
P100	553	557	552	552	551
V100	774	828	833	829	839
Turing	528	554	556	555	554
Radeon VII	-	-	-	-	814

Architectural efficiency
(Fraction of hardware peak)

Higher is better

	OpenMP	Kokkos	CUDA	OpenACC	OpenCL
Skylake	80.2%	68.1%	-	32.4%	41.8%
KNL	92.2%	62.1%	-	90.7%	58.4%
Power 9	72.8%	73.6%	-	72.5%	-
Naples	65.9%	62.7%	-	-	-
ThunderX2	85.3%	84.7%	-	-	-
Ampere	66.4%	57.3%	-	-	-
NEC Aurora	81.3%	-	-	-	-
K20	69.2%	72.9%	72.3%	-	72.8%
P100	75.5%	76.1%	75.4%	75.3%	75.3%
V100	86.0%	92.0%	92.6%	92.1%	93.2%
Turing	85.7%	90.0%	90.2%	90.1%	89.9%
Radeon VII	-	-	-	-	79.4%

University of BRISTOL

GW4

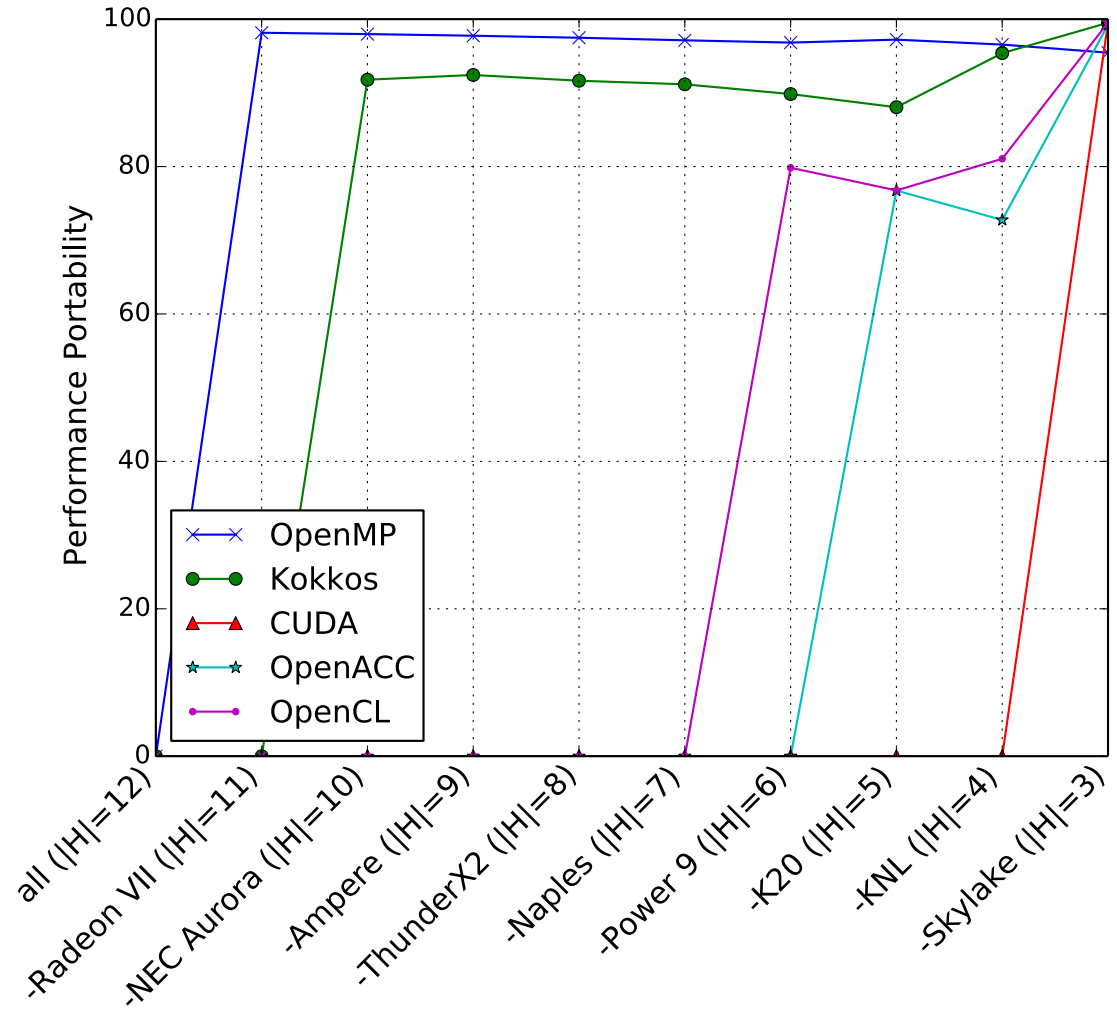
Performance portability

Higher is better

	OpenMP	Kokkos	CUDA	OpenACC	OpenCL
Skylake	80.2%	68.1%	-	32.4%	41.8%
KNL	92.2%	62.1%	-	90.7%	58.4%
Power 9	72.8%	73.6%	-	72.5%	-
Naples	65.9%	62.7%	-	-	-
ThunderX2	85.3%	84.7%	-	-	-
Ampere	66.4%	57.3%	-	-	-
NEC Aurora	81.3%	-	-	-	-
K20	69.2%	72.9%	72.3%	-	72.8%
P100	75.5%	76.1%	75.4%	75.3%	75.3%
V100	86.0%	92.0%	92.6%	92.1%	93.2%
Turing	85.7%	90.0%	90.2%	90.1%	89.9%
Radeon VII	-	-	-	-	79.4%

- Heatmaps can give an intuitive view on performance portability
  - Want to be rigorous, so use the Performance Portability metric to quantify the intuition
- The challenge is that *no* language runs successfully on all our platforms.
- We automatically create platform subsets and compute performance portability of application efficiency for each subset
  - Start with all platforms (PP = 0)
  - Remove the platform which is the least supported (the one with the most missing results)
  - If tied, remove the platform which causes biggest change in L_2-norm of performance portability from current platform subset

Observations on BabelStream Performance Portability

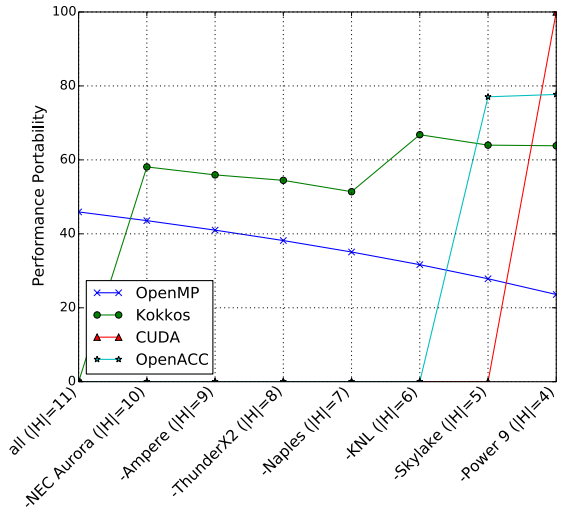


- If we exclude the AMD Radeon GPU, then **OpenMP** successfully runs on all the remaining platforms, with PP = 97.5%
- Excluding the NEC Aurora, then **Kokkos** can run across the remaining set with PP = 89.3%
- If we further exclude all non-Intel CPUs, then **OpenCL** runs with PP = 76.7% (|H|=8)
  - Improves if only consider GPUs due to NUMA related runtime issues
- Also excluding Power 9, AMD Naples and NVIDIA K20, **OpenACC** will run with similar portability to **OpenCL**.
  - Do have Power 9 result, but heuristic chose to keep K20 where we don't

TeaLeaf

Lower is better

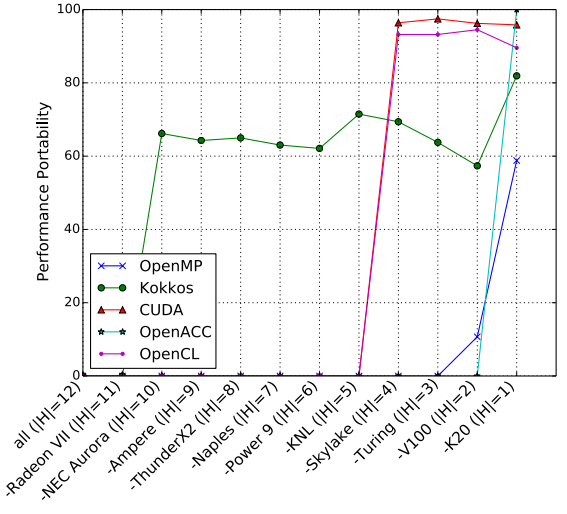
	OpenMP	Kokkos	CUDA	OpenACC
Skylake	317	370	-	-
KNL	191	885	-	-
Power 9	254	393	-	341
Naples	348	372	-	-
ThunderX2	314	439	-	-
Ampere	793	892	-	-
NEC Aurora	79.1	-	-	-
K20	1605	712	445	629
P100	190	187	122	153
V100	281	127	81.0	103
Turing	962	181	116	139



CloverLeaf

Lower is better

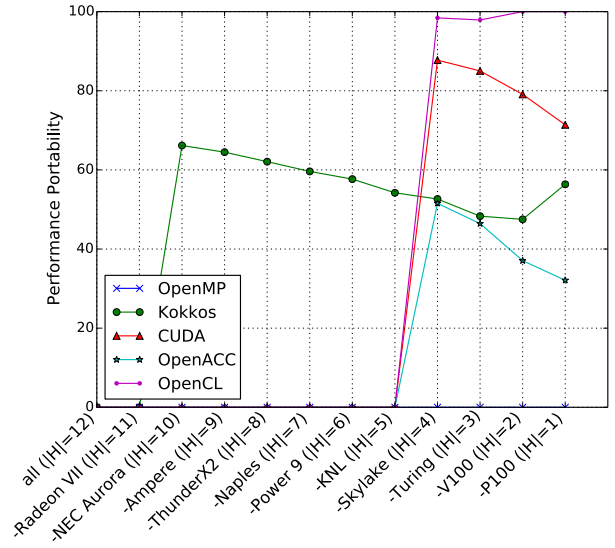
	OpenMP	Kokkos	CUDA	OpenACC	OpenCL
Skylake	376	463	-	877	-
KNL	250	666	-	698	-
Power 9	376	544	-	768	-
Naples	327	395	-	337	-
ThunderX2	457	772	-	-	-
Ampere	1309	1452	-	-	-
NEC Aurora	323	-	-	-	-
K20	9737	1297	592	-	572
P100	226	163	139	133	149
V100	-	108	88.8	90.1	97.9
Turing	-	211	213	199	213
Radeon VII	-	-	-	-	106



Neutral

Lower is better

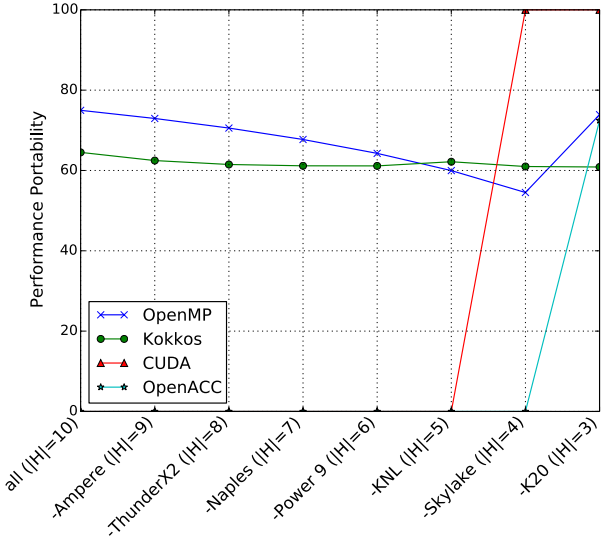
	OpenMP	Kokkos	CUDA	OpenACC	OpenCL
Skylake	8.0	13.0	-	-	-
KNL	23.8	28.1	-	-	-
Power 9	8.3	11.1	-	-	-
Naples	14.5	16.6	-	-	-
ThunderX2	12.6	13.5	-	-	-
Ampere	37.4	43.3	-	-	-
NEC Aurora	2784	-	-	-	-
K20	-	52.7	41.6	92.5	29.7
P100	-	9.5	4.4	8.9	3.9
V100	-	6.2	3.1	3.3	3.3
Turing	-	9.3	6.9	8.7	6.7
Radeon VII	-	-	-	-	3.7



MiniFMM

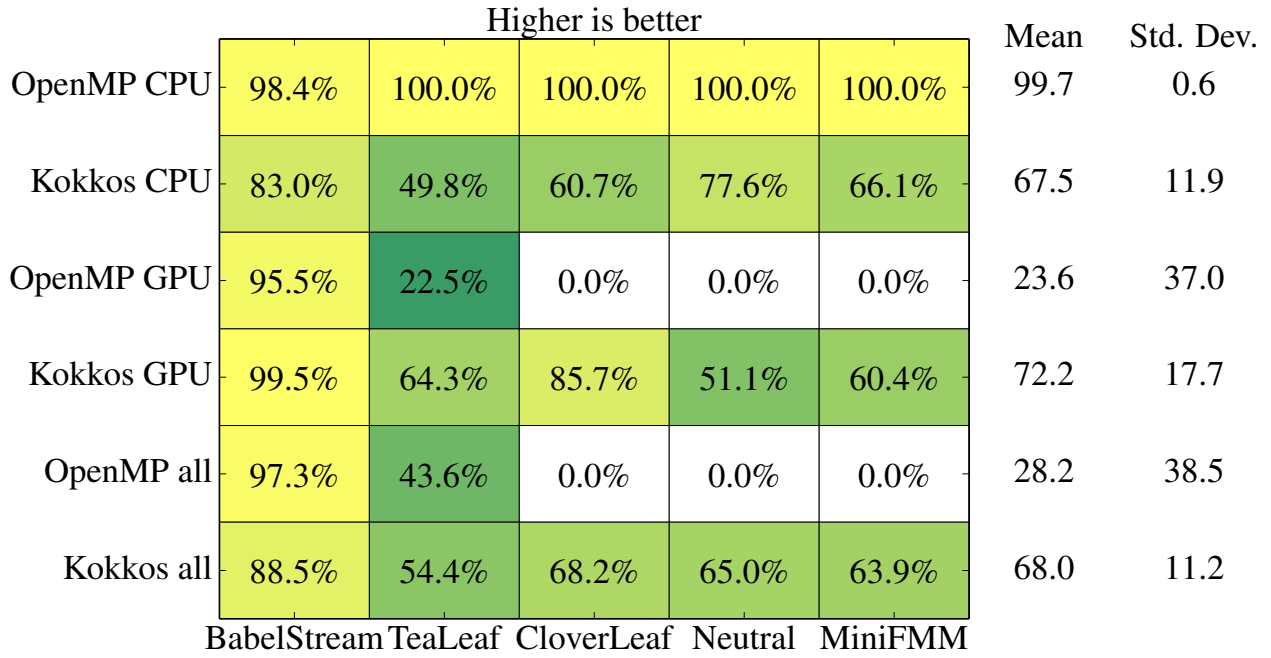
Lower is better

	OpenMP	Kokkos	CUDA	OpenACC
Skylake	8.7	12.9	-	-
KNL	11.4	20.2	-	-
Power 9	23.6	38.5	-	-
Naples	13.1	20.5	-	-
ThunderX2	21.9	30.6	-	-
Ampere	116	127	-	-
K20	56.7	28.2	17.3	-
P100	5.0	4.7	3.5	4.3
V100	3.1	4.4	2.5	3.8
Turing	3.2	4.2	2.3	3.2



Performance Portability of OpenMP and Kokkos

- Heatmap shows PP metric on chosen platform subsets
- Rows indicate how a model fairs across different applications
- OpenMP achieving best performance on CPUs but struggles on GPUs due to support
- Kokkos shows a small overhead on CPUs
  - PP metric tells us to expect the abstraction of OpenMP/CUDA to reduce performance by ~15-50%



- Final row here (Kokkos all) shows performance portability is possible
  - Mean and standard deviation shows we would expect Kokkos to achieve 59-79% of best application performance on average

Overall Performance Portability observations thus far

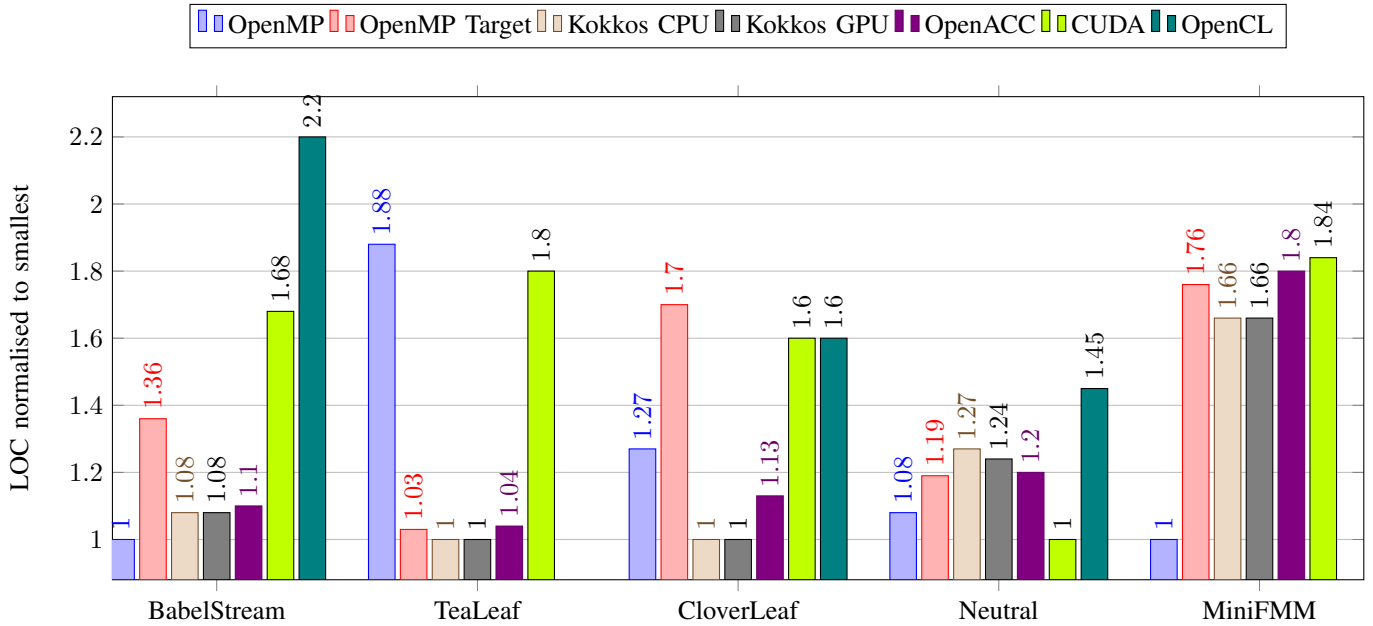
- Performance portability can be a very mixed bag
  - A language may do well on one code, then poorly on the next
- OpenMP and Kokkos achieving the best platform coverage
- Big differences between compilers for PP (esp. OpenMP target)
- Kokkos doing the best in allowing applications to achieve performance portability across architectures
  - Our PP goal was to achieve 20% of best performance, and Kokkos achieves within 32% of best performance on average
- OpenACC struggling for coverage on the CPUs (x86. A64fx? TX4?)
  - Symptom of vendor controlled, non-open/standardised programming models?
  - Hard to talk about performance portability when *portability* is limited

Where next for the Bristol Performance Portability study?

- Eventually aiming for about 10 codes in 8 languages across 14 platforms (980 combinations!)
- Can we achieve performance portability between CPUs and GPUs using widely supported (by vendors) industry standards?
- If not, what's stopping us, and what can we do about it?
- Can we find good examples of codes that work well, and codes that are inherently hostile to performance portability?
- Want to push improvements in compilers, run-times, libraries, and even architectures, to improve prospects for PP

Thoughts on productivity

- Lines of code of each mini-app/model shows expected trends in verbosity of some models
- Original CloverLeaf and TeaLeaf are rather long compared to the other ports
  - Is programmer style a factor here?
- More sophisticated productivity metrics require capturing information during development
  - Hard to quantify for existing codes/ports



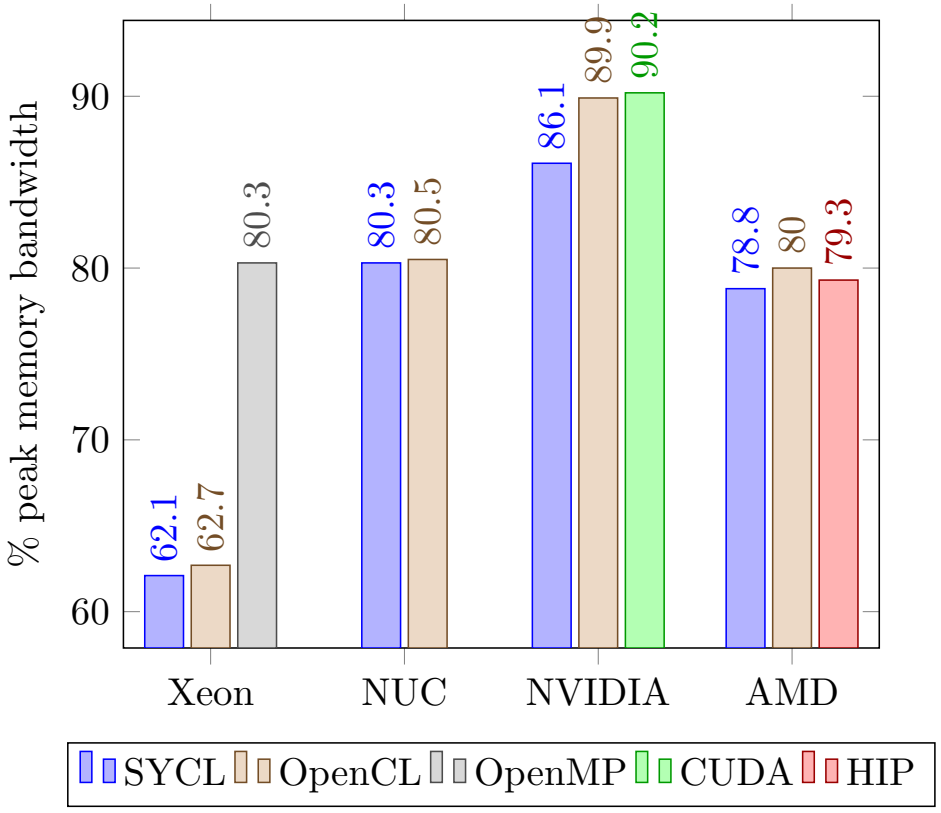
- Porting these mini-apps to each programming model took around 2 weeks
  - Lines of Code doesn't amortize away developer time saved once the first parallel loop is written

Performance Portability of SYCL

- SYCL is a single-source C++ parallel programming model for heterogenous platforms from Khronos
  - Open standard
  - Modern C++
  - Commercial support from Intel with oneAPI/DPC++ and Codeplay
  - Open-source support growing to support wider set of platforms
- One possible option for programming CPUs, GPUs, etc in a performance portable way

Performance Portability of SYCL

- Paper at IWOCL explored performance on Intel CPUs and GPUs from Intel, AMD and NVIDIA.
  - Comparisons with OpenCL, OpenMP, CUDA and HIP
  - Very promising results so far, but more work to do in the HPC ecosystem
  - Intel's OpenCL runtime on CPUs has known issues which hopefully will improve as part of oneAPI



BabelStream Triad

University of BRISTOL

https://doi.org/10.1145/3388333.3388643

GW4

Performance Portability of SYCL

- Early work running SYCL on Arm
  - First results on ThunderX2 using hipSYCL on top of OpenMP
  - Performance close to native OpenMP for BabelStream
  - Known limitations of backend hipCPU implementation limit performance on all CPU platforms (both Intel and Arm) for more involved benchmarks
  - Currently exploring other avenues for SYCL on CPUs (Intel, AMD and Arm)

- SYCL's future is looking bright:
  - Early view of SYCL-2020 shows lots of new HPC-friendly features
    - https://www.iwocl.org/iwocl-2020/conference-program/#panel
  - Support for NVIDIA GPUs added to open-source version of DPC++
  - Critical part of Argonne National Laboratory path to Exascale with Aurora
  - Robust support from/for Arm and AMD the next step

IWOCL 2020
SYCLcon

The 8th International Workshop on OpenCL and the SYCL Developer Conference

Register for free and see the full programme online. Live panel with the OpenCL and SYCL standards developers on Tuesday April 28th from 4pm BST. Free SYCL tutorials on Monday April 27th and Wednesday April 29th:

www.iwocl.org

Thanks to our sponsors

KHRONOS GROUP intel codeplay University of BRISTOL TUM lrz SCIENTIFIC COMPUTING WORLD hgpu.org

SYCL Academy

SYCL Academy offers a fantastic set of open source materials and can be used to learn and teach SYCL™ development.

Visit GitHub.com

SYCL v1.2.1 Specification

Click here to read or download the full Khronos® specification for SYCL™ 1.2.1 in PDF format.

Browse Implementations

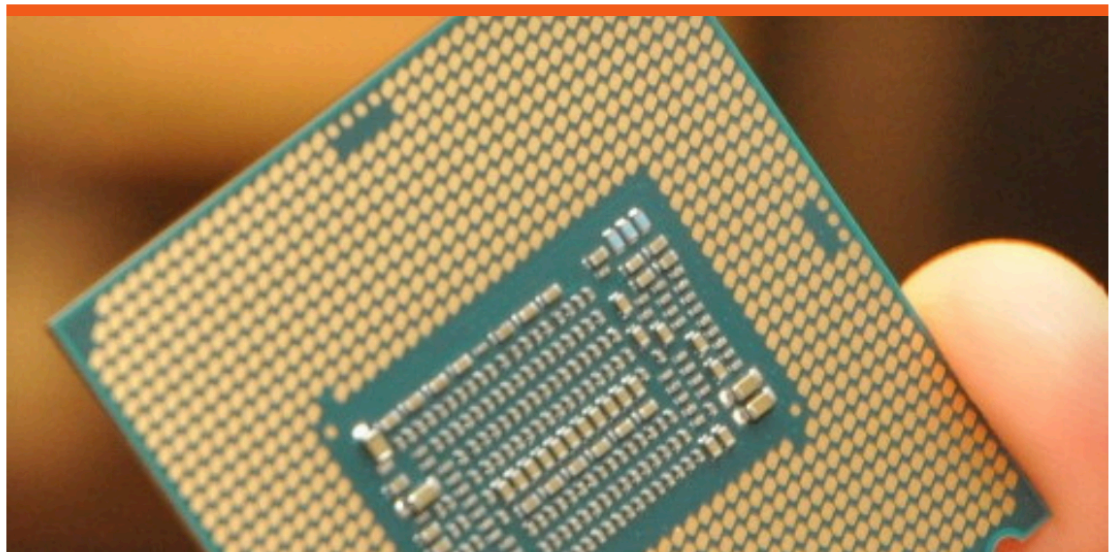
Click here to find out where to get all the available SYCL implementations from.

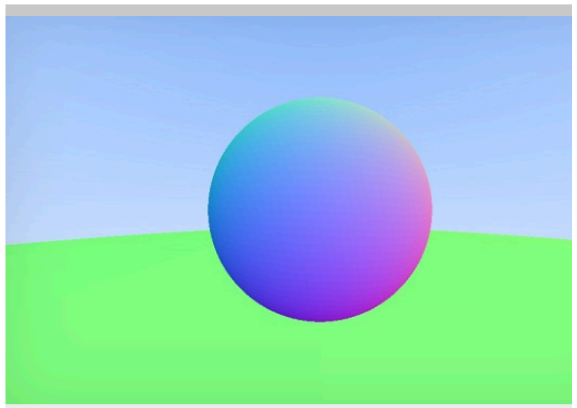
Conformance Test Suite

The test suite is open source and hosted on GitHub. Contributions from the community to the CTS are welcome.

SYCL Working Group

Visit the SYCL™ Khronos® working-group's home page to learn more about the SYCL technology.





20 May 2020

Ray-tracing in a Weekend

22 May 2020

Aurora Workshop Helps

Which performance portable programming model should I use?

- Want codes to run well everywhere, so how should I write them and what should I write them in?
- Tried a number of approaches:
  - Clone the code to allow study of performance portability
    - Multiple versions of the code exist
  - Lightweight interfaces to allow specialisation
    - Put simple library abstractions into the code
    - Portability layers like Kokkos is a grandiose approach to this
  - Performance portable standard programming models
    - OpenMP and SYCL offer the best hope today, but ecosystem needs support

Lessons learned about achieving performance portability

1. **Use open (standard) parallel programming languages** supported by multiple vendors across multiple hardware platforms
   - **E.g.** OpenMP, SYCL, Kokkos, Raja, …?
2. **Expose maximal parallelism** at all levels of the algorithm and application
3. **Avoid over-optimising** for any one platform
   - Optimise for at least two different platforms at once
4. **Multi-objective autotuning** can significantly improve performance
   - **Autotune for more than one target at once**
   - See: **Exploiting auto-tuning to analyze and improve performance portability on many-core architectures**, J.Price and S. McIntosh-Smith, P^3MA, ISC'17

- **High Performance *in silico* Virtual Drug Screening on Many-Core Processors**
  S. McIntosh-Smith, J. Price, R.B. Sessions, A.A. Ibarra, IJHPCA 2014
- **On the performance portability of structured grid codes on many-core computer architectures**
  S.N. McIntosh-Smith, M. Boulton, D. Curran, & J.R. Price
  ISC, Leipzig, June 2014. DOI: 10.1007/978-3-319-07518-1_4
- **Assessing the Performance Portability of Modern Parallel Programming Models using TeaLeaf**
  Martineau, M., McIntosh-Smith, S. & Gaudin, W.
  Concurrency and Computation: Practice and Experience (Apr 2016)
- **GPU-STREAM v2.0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models**
  Deakin, T. J., Price, J., Martineau, M. J. & McIntosh-Smith, S. N.
  First International Workshop on Performance Portable Programming Models for Accelerators (P3MA), ISC 2016
- **The Productivity, Portability and Performance of OpenMP 4.5 for Scientific Applications Targeting Intel CPUs, IBM CPUs, and NVIDIA GPUs**
  M. Martineau and S. McIntosh-Smith, IWOMP 2017, Stony Brook, USA.

- **Evaluating Attainable Memory Bandwidth of Parallel Programming Models via BabelStream**
  Deakin, T, Price, J, Martineau, M, and McIntosh-Smith, S
  International Journal of Computational Science and Engineering (special issue), vol 17., 2018
- **Pragmatic Performance Portability with OpenMP 4.x**
  Martineau, Matt, Price, James, McIntosh-Smith, Simon, and Gaudin, Wayne
  Proceedings of the 12th International Workshop on OpenMP, 2016
- **Performance Analysis and Optimization of Clang's OpenMP 4.5 GPU Support**
  Martineau, Matt, McIntosh-Smith, Simon, Bertolli, Carlo, et al
  Proceedings of the International Workshop on Performance Modelling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), 2016, SC'16
- **Exploiting auto-tuning to analyze and improve performance portability on many-core architectures**
  Price, J. & McIntosh-Smith, S., P^3MA, ISC High Performance 2017 International Workshops, Revised Selected Papers. Springer, Cham, p.538-556, vol. 10524 LNCS

For more information

Bristol HPC group: https://uob-hpc.github.io/

Build & run scripts:

https://github.com/UoB-HPC/benchmarks/tree/doe-p3-2019

Twitter: **@simonmcs**