Reviewing the Computational Performance of Deterministic Sn Transport Sweeps on Many-core Architectures

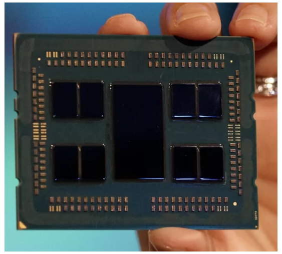
Tom Deakin, Simon McIntosh-Smith (UoBristol)

Justin Lovegrove, Richard Smedley-Stevenson, Andrew Hagues (AWE)

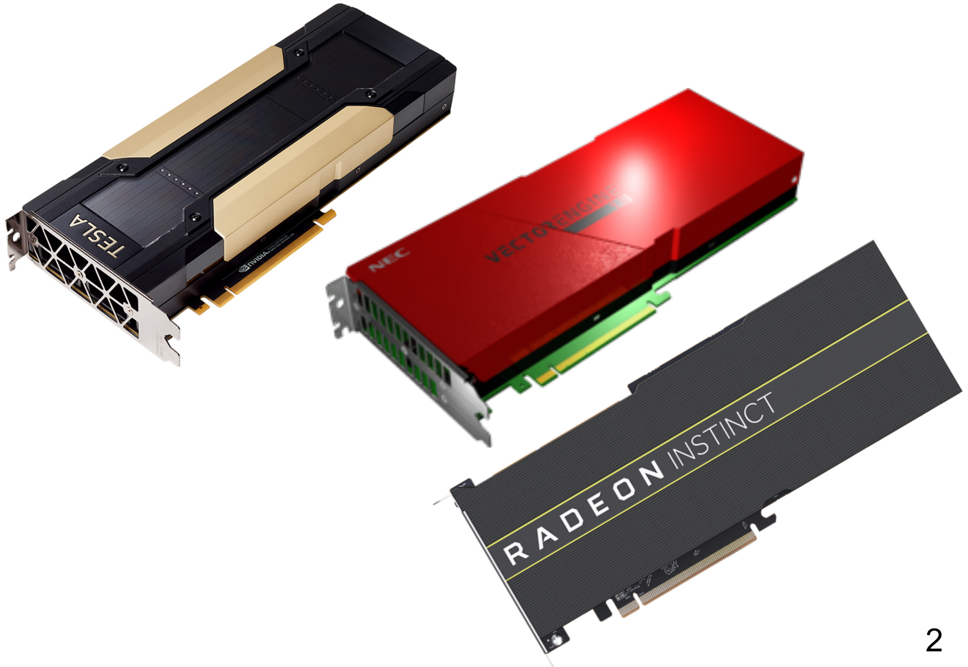
tom.deakin@bristol.ac.uk

Recent processor trends in HPC

Many-core CPUs



GPUs/accelerators



http://uob-hpc.github.io

Architectural trends

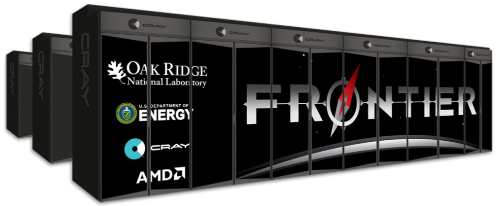
- CPUs evolved to have wide vectors and lots of cores.
- 32-core CPUs now common.
- Expecting 64-core CPUs to arrive within next 12 months.
- Renewed competition in CPUs crucial to health of HPC ecosystem and performance/$.

- GPUs including latest high bandwidth memory technology.
- Architectures with many-cores and wide vectors.
- Cores becoming more complex, including caches, specialised accelerators, etc.
- Increased competition from vendors is GPGPUs.

http://uob-hpc.github.io

Supercomputers

- Next generation of world-class large supercomputers will contain a diverse range of architectures.
  - Frontier: AMD EPYC CPUs and Radeon GPUs
  - Aurora: Intel Xeon CPUs and Xe GPUs
  - Perlmutter: AMD EYPC CPUs and NVIDIA GPUs
  - El Capitan: TBA
  - Fugaku: Fujitsu A64FX Arm CPUs
- We're going to have to ensure our codes run on diverse systems.



http://uob-hpc.github.io

Structured transport on GPUs

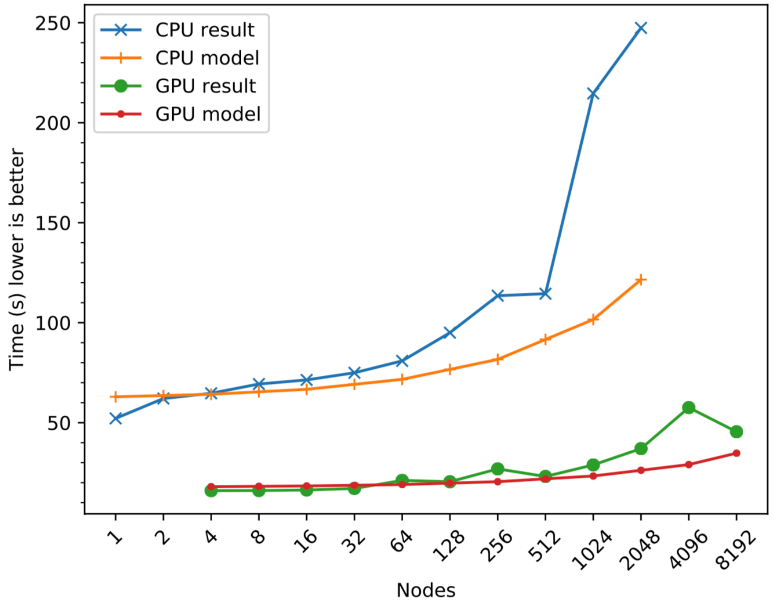
- Our earlier work [1-3] focused on performance limiting factors on many-core for finite difference, structured grid, deterministic Sn transport on GPUs.
- Developed a parallel scheme to best exploit the high memory bandwidth available on GPUs.
  - Originally written for NVIDIA K20X Kepler GPUs, but shown to be effective on P100 Pascal and V100 Volta GPUs.
- Needed to utilise concurrency in angles within an octant, all energy groups (Jacobi iterations as R. Baker MC2015 paper) **and** cells on a wavefront in local subdomain.
  - A scheme directly at odds with requirements for efficient sweeps (c.f. Adams @ TAMU/Bailey @ LLNL).
- Using LANL's SNAP mini-app, showed our parallel scheme is limited by GPU memory bandwidth, and achieved speedups over CPUs in line with this metric [1].
- LLNL's KRIPKE's flux-register algorithm (MC2019) uses similar parallelism but includes an optimisation to reduce memory movement at expense of reduced parallelism.
  - Note KRIPKE has a different spatial decomposition.

[1] T. Deakin, "Leveraging many-core technology for deterministic neutral particle transport at extreme scale," Ph.D. thesis, University of Bristol, 2018.
[2] T. Deakin, S. McIntosh-Smith, M. Martineau, and W. Gaudin, "An improved parallelism scheme for deterministic discrete ordinates transport," International Journal of High Performance Computing Applications, sep 2016.
[3] T. Deakin, S. McIntosh-Smith, and W.Gaudin, Many-Core Acceleration of a Discrete Ordinates Transport Mini-App at Extreme Scale. Cham: Springer International Publishing, 2016, pp. 429–448.

Performance models for structured transport

- Developed performance models for scaling on GPU systems [3].
  - Modelling computation cost as grind time x amount of work inaccurate on GPUs.
  - More accurately modelled as product of number of local wavefronts in local sweep (#angles/groups somewhat unrelated).
- At high processor counts, point-to-point communication becomes the bottleneck.
  - At 2048 GPUs, 80% of time on Titan and 60% time on Piz Daint in communications.
  - In part due to idle time in startup/teardown, but also due to acceleration of solve time vs message cost.
- Results show KBA scaled well enough up to 8000 GPU.
  - 1 MPI rank/GPU.
  - Problem size we could test limited by K20X capacity (6 GB).
  - Larger problems would scale better.

Weak scaling SNAP on Titan [1,3]

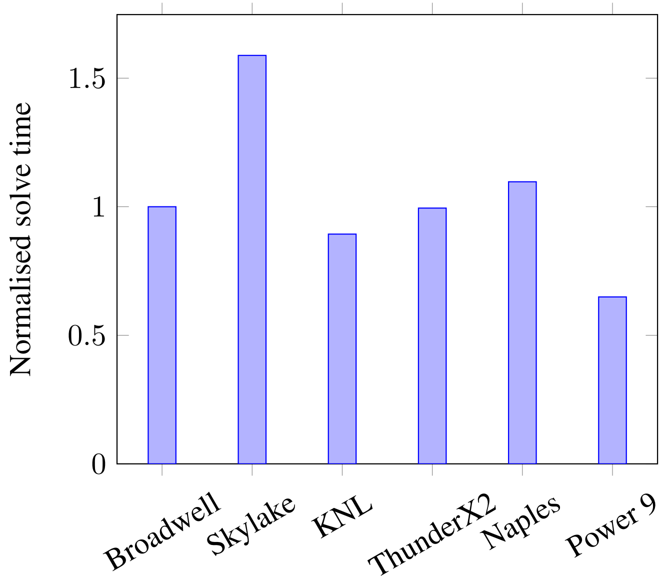


[1] T. Deakin, “Leveraging many-core technology for deterministic neutral particle transport at extreme scale,” Ph.D. thesis, University of Bristol, 2018.
[2] T. Deakin, S. McIntosh-Smith, M. Martineau, and W. Gaudin, “An improved parallelism scheme for deterministic discrete ordinates transport,” International Journal of High Performance Computing Applications, sep 2016.
[3] T. Deakin, S. McIntosh-Smith, and W.Gaudin, Many-Core Acceleration of a Discrete Ordinates Transport Mini-App at Extreme Scale. Cham: Springer International Publishing, 2016, pp. 429–448.

Structured transport on many-core CPUs

- Parallelism for LANL's SNAP provides sufficient work for CPU cores:
  - MPI KBA spatial decomposition (pipelined sweeps)
  - OpenMP threads of energy groups
  - Compiler automatic vectorisation over angles in octant
- As a tool to explore performance bounds, we developed the mega-sweep mini-mini-app, extracting just the sweep kernel.
- Computational intensity is low (low ratio of FLOPs/bytes), so Cache-aware Roofline model classifies as memory bandwidth bound.
- But performance results don't correlate with available memory bandwidth of processors.

80x8x8 cells, 48 angles/octant, 64 groups



https://github.com/uk-mac/mega-stream

Deakin, T., Gaudin, W., & McIntosh-Smith, S. (2017). On the Mitigation of Cache Hostile Memory Access Patterns on Many-Core CPU Architectures (pp. 348–362). Frankfurt: Springer International Publishing. https://doi.org/10.1007/978-3-319-67630-2_26
Tom Deakin, John Pennycook, Andrew Mallinson, Wayne Gaudin and Simon McIntosh-Smith. The MEGA-STREAM Benchmark on Intel Xeon Phi Processors (Knights Landing). The Intel Xeon Phi Users Group Spring Meeting, 2017.
T. Deakin, "Leveraging many-core technology for deterministic neutral particle transport at extreme scale," Ph.D. thesis, University of Bristol, 2018.

Core-bound, or not core-bound?

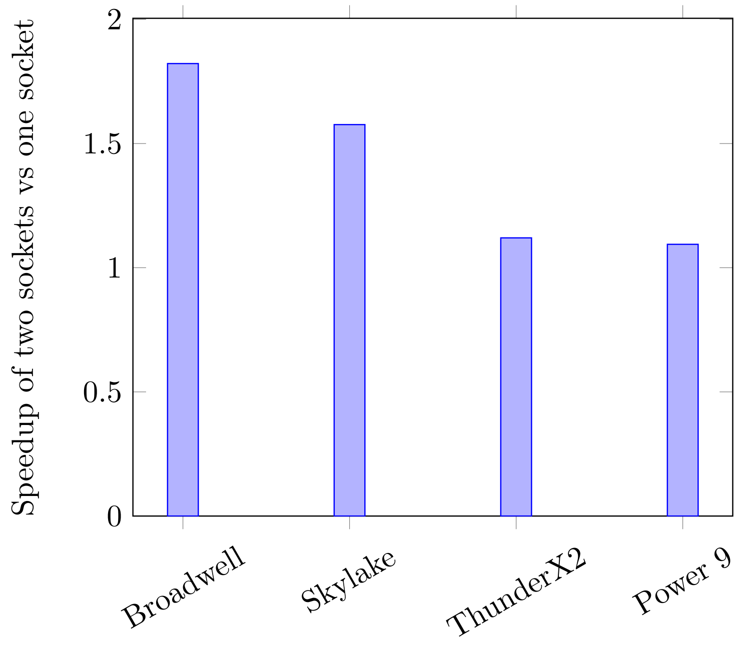
- Follow a procedure by Voysey (Met Office) to help discover performance limiting factors:
  1. Run on all cores of one socket. (e.g. 18 cores of one Broadwell socket)
  2. Run on half of cores of both sockets. (e.g. 2 x 9 cores)
- If performance improves, performance is bound in shared resources such as memory bandwidth.
  – E.g. Two sockets give you twice the main memory bandwidth of one socket.
- Otherwise, bound by on-core resources.
  – Same number of cores, so have same number of FLOPs, same cache bandwidth/size, etc.
- Warning! Sometimes see increase in clock speed for the two-socket run.

A. Voysey and M. Glover. “Performance of Met Office Weather and Climate Codes on Cavium ThunderX2 Processors.” (2018). URL https://www.youtube.com/watch?v=xSLY0RJBEAQ. Presentation at Arm Research Summit, Austin, Texas.

Relative performance improvement

- On Power 9 and ThunderX2, little improvement from second socket.
  - Bound by on-core resources (L1/L2 cache).
- On Intel Xeons, second socket does improve performance.
  - Bound by off-core resources.
- But on Skylake < 2X difference, and clock speed increased by 10%, so…
  - …unlikely to be main memory bandwidth alone:
  - Non-temporal stores improve performance (~1.4X).
  - ThunderX2 has more main memory bandwidth per socket than Xeon, but doesn't result in faster runtimes here.
  - Two sockets also give more L3 cache…
- Performance of deep memory hierarchy is multifaceted problem.
  - Currently doing further experiments to pin down the cache feature exactly.

80x8x8 cells, 48 angles/octant, 64 groups



Modelling runtime with cache misses

- Using Linux perf, we can gather performance counter data during a run of mega-sweep on ThunderX2.
  - Collect L1 data cache misses and total number of cycles.
  - Calculate an average miss latency using appropriate counters with help from your vendor.
    - This is an application dependent value due to prefetching behaviour.
- Example run, for the 4 backwards sweeps (-x):
  - 106,932,262,330 cycles (107 billion cycles)
  - 3,104,413,875 L1 misses (3.1 billion)
  - Derived 31 cycles for a miss using vendor-known counters for this binary.
- L1 misses * latency = 96.4 billion cycles, 90% of total cycles.
- For forwards sweeps (with similar maths) we see 63% of total cycles.
  - Prefetching is proving more accurate on forwards sweeps here.
- Therefore, execution time (equiv. number of cycles) is pretty similar to the time spent waiting for cache misses to be satisfied.

Thanks to Rabin Sugumar at Marvell

Structured transport performance bounds

- Performance bounds for structured transport depends on where you look!
- At scale, becomes bound by cost (latency) of communications.
- CPU solve bound by cache performance.
  - Local spatial concurrency disrupts memory access pattern.
  - Performance dictated by cache architecture supporting high reuse of neighbouring flux arrays **and** streaming of angular flux.
- GPU solve bound by device memory bandwidth.
  - Expose maximal concurrency and utilise latency tolerant properties of GPUs to leverage high performance.

T. Deakin, “Leveraging many-core technology for deterministic neutral particle transport at extreme scale,” Ph.D. thesis, University of Bristol, 2018.

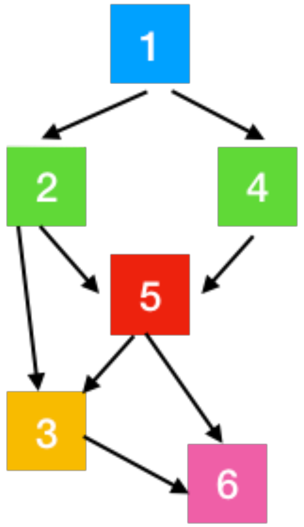
The UnSNAP mini-app

- Port of (structured) SNAP from Los Alamos National Laboratory.
  - Use SNAP's Sn quadrature, material data, material layout, source update approximations, etc.
  - Maintain R. Baker's use of Jacobi in energy groups for concurrency.
- Unstructured mesh of 3D hexahedral DG elements.
  - Allows for arbitrary order discontinuous Lagrange elements.
    - Nodes on vertices, edges, faces and inside volume.
  - Method allows for distorted and curved elements.
- Matrix-free finite element solve.
  - Basis function integration precomputed.
- Sweep schedule computed for each angle and stored in graph.
  - Fully upwind sweep schedule on each node.
  - This study focuses on on-node performance of different architectures.
    - Consider multi-node in the future leveraging existing body of research.

T. Deakin, S. McIntosh-Smith, J. Lovegrove, R. Smedley-Stevenson, and A. Hagues. "UnSNAP: a mini-app for exploring the performance of deterministic discrete ordinates transport on unstructured meshes." In Cluster Computing, IEEE International Conference on, pp. 566–574. Belfast (2018).

CPU parallelism

- Each t-level in sweep schedule for angle contains list of elements to solve.
  - Nested loop over elements in level and energy groups, threaded with OpenMP.
    - ➢ Solving energy groups in element helps with memory contiguity and sufficient parallelism (see WRAp paper for details).
- Small linear system assembled and solved from sources, cross-sections, Sn quadrature and precomputed integrals of basis pairs.
  - Gaussian Elimination for solution. Numerically stable enough, and full factorization very expensive.
  - SIMD vectorization of element nodes.
    - ➢ Corresponds to vectorising rows of matrix and of the right-hand side vector.



GPU parallelism

- Early results from CUDA implementation, linear elements showing promising performance.
  - Full port of UnSNAP so data is resident in GPU memory.
- Combine schedule for angles in angle set as need more concurrency for sufficient work on GPU.
- Launch a kernel for each t-level in combined graph.
- One thread block for each element, group, angle.
- For linear elements, 64 threads/block, one thread per matrix entry.
- This approach hits device limits for higher orders: thread over matrix rows.
- Use atomic updates for scalar flux reduction from parallel angle.
- Linear system assembled and solved in shared memory (manual caching).

Hardware

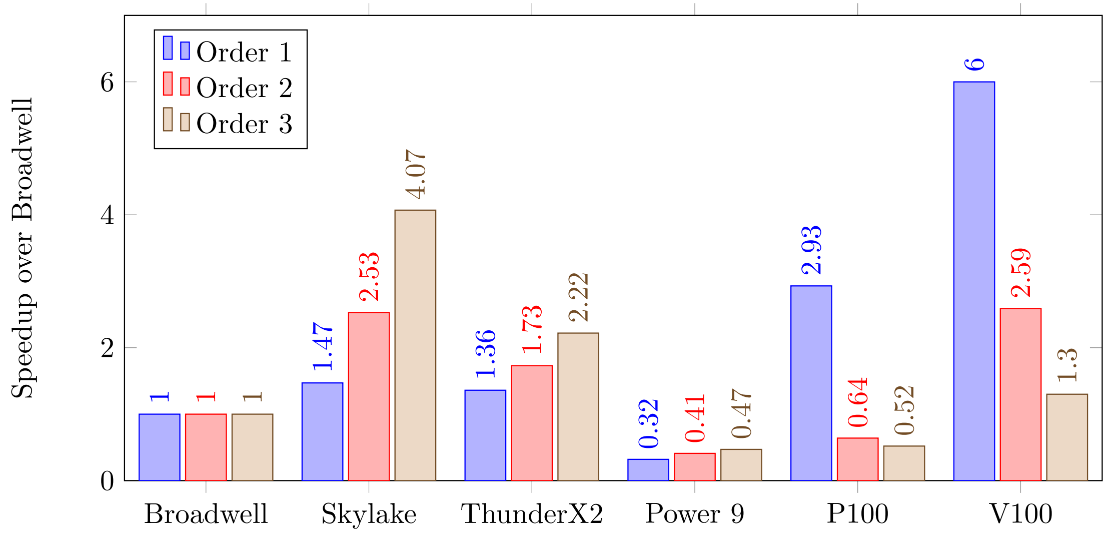
Architecture	Cores	Clock GHz	Peak FP64 TFLOP/s	Main memory bandwidth GB/s
Intel Broadwell	18×2	2.1	1.21	154
Intel Skylake	28×2	2.1	3.76 (3.1X)	256 (1.7X)
Marvell ThunderX2	32×2	2.5	1.28 (1.1X)	288 (1.9X)
IBM Power 9	20×2	3.2	1.02 (0.8X)	340. (2.2X)
NVIDIA P100	60 SMs	1.13	4.04 (3.3X)	732 (4.8X)
NVIDIA V100	84 SMs	1.37	7.01 (5.8X)	900 (5.8X)

- Broadwell as performance baseline.
- Dual-socket CPUs, and one GPU accelerator.
- Theoretical relative performance of these simple metrics shown in brackets.

Performance results

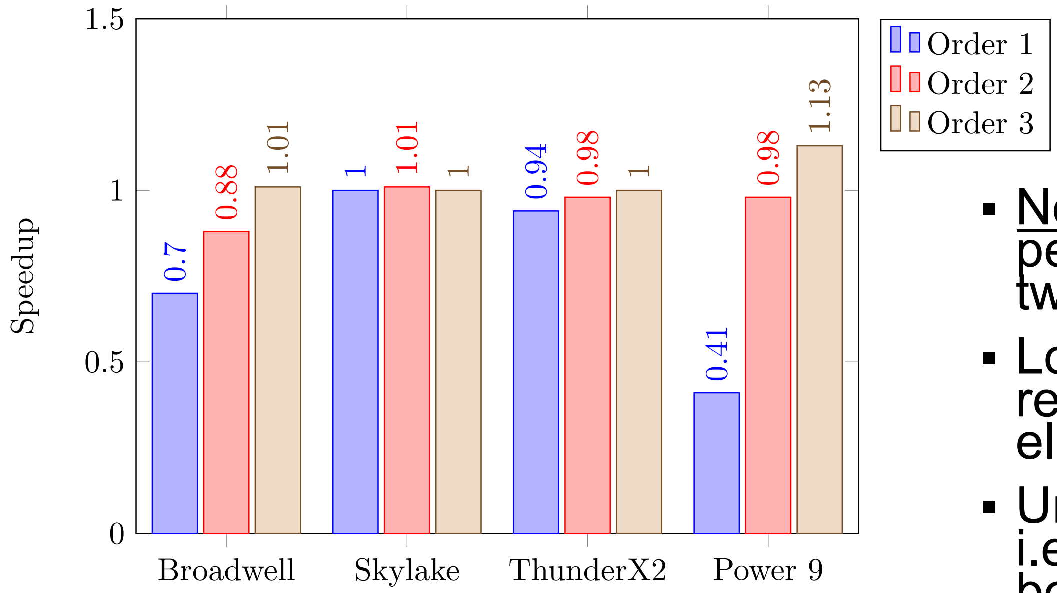
512 cells, 80 angles, 32 energy groups = 1.3M TDoF

10.5M-83.9M unknowns (depending on order)



- Performance normalised to Broadwell. All CPUs dual-socket, and compare to one GPU.
- For linear elements, ThunderX2 and Skylake performance similar, around 1.4X over Broadwell.
- 6X performance improvement over Broadwell for one V100 GPU for linear elements.
- Skylake fastest for higher orders.
- Improvements here don't relate to simple FLOP/s or main memory bandwidth of processors.

Relative performance improvement



- Comparing improvement from one fully populated socket to two half populated sockets.

- No CPU shows significant performance improvement from one to two sockets for any order.
- Low arithmetic intensity as more bytes read for assembly and Gaussian elimination than number of FLOPs.
- UnSNAP bound by on-core resources, i.e. not main memory bandwidth bound.
- Broadwell L1 cache hit rate over 95%.
  - L2 cache hit goes from 53% single socket to 7% two socket.
  - Note: Angular flux is larger than L3 cache for all orders.

Performance analysis

- Relative L1 cache bandwidth between processors not the whole picture.
- Each linear system small enough to fit in cache, so access to cache dictates the performance on CPU.
- Caches designed to be transparent to applications, so analysis is difficult.

- GPU also not limited by device memory bandwidth.
  - Nvprof shows very few device memory accesses.
  - Most memory requests satisfied by GPU caches rather than HBM.
    - L2 cache had 96% hit rate.
- Parallel angle schedule on GPU required for sufficient parallelism, 2.5X faster than serial angle schedule for linear elements.
  - On a CPU, parallel angle schedule much slower.

Summary

- Solving the transport equation is challenging!
- Our algorithms are going to need to expose sufficient concurrency to work efficiently on parallel many-core architectures.
- On CPUs, both structured and unstructured transport is limited by the performance of the cache.
- On GPUs, we see different performance bounds based on the discretisation method:
  - Structured grid/finite difference: main memory bandwidth.
  - Unstructured grid/DG finite element: device cache performance.

https://uob-hpc.github.io tom.deakin@bristol.ac.uk