Received: 31 August 2019 | Accepted: 18 September 2019

DOI: 10.1002/cpe.5569

WILEY

SPECIAL ISSUE PAPER

Benchmarking the first generation of production quality Arm-based supercomputers

Simon McIntosh-Smith | James Price | Andrei Poenaru | Tom Deakin

HPC Research Group, Department of Computer Science, University of Bristol, Bristol, UK

Correspondence
Simon McIntosh-Smith, HPC Research Group, Department of Computer Science, University of Bristol, Bristol BS8 1QU, UK.
Email: S.McIntosh-Smith@bristol.ac.uk

Funding information
Engineering and Physical Sciences Research Council (EPSRC), Grant/Award Number: EP/P020224/1; GW4 Alliance; Met Office, Cray, and Arm; ASiMoV EPSRC, Grant/Award Number: EP/S005072/1

Abstract

In this paper, we present scaling results from two production quality supercomputers that use the first generation of Arm-based CPUs that have been optimized for scientific workloads. Both systems use Marvell ThunderX2 CPUs, which deliver high core counts and class-leading memory bandwidth. The first system is Isambard, a Cray XC50 "Scout" system operated by the GW4 Alliance and the UK Met Office as a Tier-2 national HPC service. The second system is one of three Arm-based HPE Apollo 70 systems delivered as part of the Catalyst UK project, running at the University of Bristol. We compare scaling results from these two systems with three Cray XC50 systems based on Intel Skylake and Broadwell CPUs. We focus on a range of applications and mini-apps that are important to the UK national HPC service, ARCHER, and to our project partners. We also compare the performance and maturity of the state-of-the-art toolchains available on Arm-based HPC systems.

KEYWORDS

Arm, benchmarking, HPC, ThunderX2

1 | INTRODUCTION

The development of Arm processors has been driven by multiple vendors for the fast-growing mobile space, resulting in rapid innovation of the architecture, greater choice for consumers, and competition between vendors. On May 7, 2018, Marvell announced the general availability of ThunderX2, with a recommended retail price (RRP) for the 32c 2.2 GHz part of $1795 each. The year 2018 also saw the delivery of the first generation of competitive high-performance computing (HPC) systems based on these processors, including the "Astra" system[*] at Sandia National Laboratory, the first Arm-based supercomputer to achieve a listing in the TOP500. High-performance Arm server CPUs are starting to emerge from more vendors, and in the next couple of years, we expect to see Arm-based HPC server CPUs shipping from Marvell, Fujitsu, Ampere, and Huawei.

In response to these developments, the "Isambard" system[†] was developed as the world's first Arm-based production supercomputer, and delivered as a Cray XC50 (Scout) system. Based on Marvell ThunderX2 32-core CPUs, Isambard is different from most of the other early Arm systems since it is intended to be a production service, rather than a prototype or testbed machine—Isambard is openly available to any researcher funded by the UK's Engineering and Physical Sciences Research Council (EPSRC) as part of the UK national HPC ecosystem.[‡] Isambard is also being widely used by researchers outside the UK who want to evaluate the Arm architecture for their scientific workloads. ThunderX2 CPUs are noteworthy in their focus on delivering class-leading memory bandwidth: each 32-core CPU uses eight DDR4 memory channels to deliver STREAM triad memory bandwidth of around 240 GB/s. The Isambard system represents a collaboration between the GW4 Alliance (the universities of Bristol, Bath, Cardiff, and Exeter), the UK's Met Office, Cray, Arm, and Marvell, with funding coming from EPSRC. At CUG,[1]

[*] https://share-ng.sandia.gov/news/resources/news_releases/top_500/
[†] http://gw4.ac.uk/isambard/
[‡] http://www.hpc-uk.ac.uk/facilities/

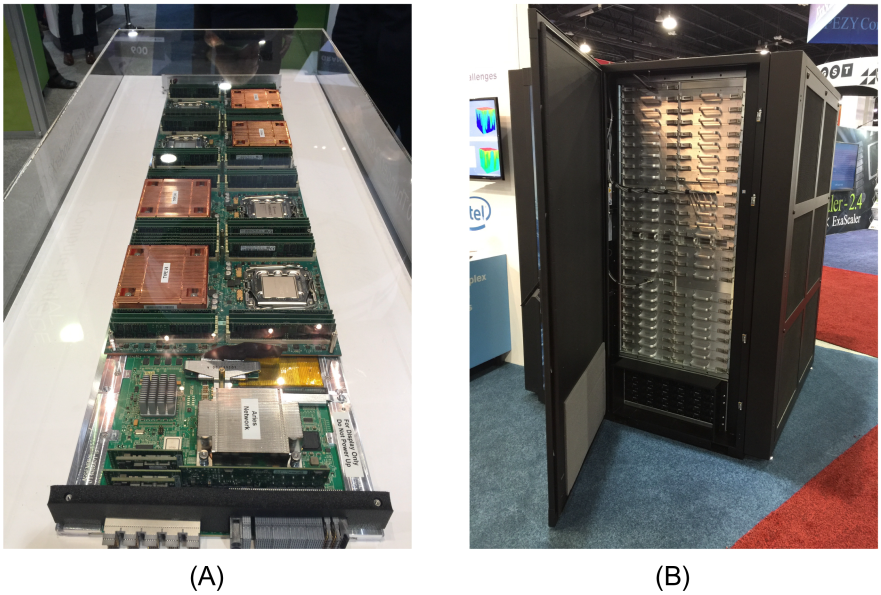


FIGURE 1 Isambard hardware. Pictures © Simon McIntosh-Smith

we showed initial single-node results based on early-access systems that demonstrated that the performance of the Arm-based ThunderX2 CPUs was comparable to state-of-the-art x86 processors available at the time. The full Isambard system was delivered in November 2018, and we presented our initial scaling results at CUG,[2] showing that Arm-based supercomputers remain competitive at scale, running real scientific workloads. Since then, the machine has opened up to computational scientists across the UK for production use.

Hewlett Packard Enterprise (HPE), another prominent HPC systems vendor, is also investing heavily in delivering an Arm-based product portfolio. In collaboration with Arm, SUSE, and three UK universities, HPE announced the Catalyst UK project in April 2018, which aims to strengthen the Arm software ecosystem for scientific computing.[§] As part of this program, HPE delivered an Apollo 70 system to each of EPCC, the University of Bristol, and the University of Leister. These systems make use of the same Marvell ThunderX2 32-core processors found in Isambard, coupled with the latest Mellanox InfiniBand interconnect. The University of Bristol received their system in December 2018, and it is currently being used both for benchmarking and by scientists across the university running production workloads.

In this work, we derive our benchmarks from two sources. Some of our benchmarks come from the most heavily used codes that are run on the UK's national supercomputer, ARCHER (a Cray XC30 system), and to these, we add a set of well-known mini-apps. Together, this set provides representative coverage of the types of codes used by citizens of today's HPC community.[3] Being a standard XC50 system, Isambard presents a unique opportunity for comparative benchmarking against XC50 machines based on mainstream x86 CPUs, including Broadwell and Skylake processors. With near-identical Cray software stacks on both the Arm and x86 XC50 machines, and with a consistent Aries interconnect, Isambard enables as close an “apples-to-apples” comparison between Arm and x86-based processors as possible. The inclusion of our Catalyst system in this study allows us to highlight any differences in scalability when using the two different interconnects, both with the same ThunderX2 CPUs. We believe these results are the first to compare scalability across Arm systems using different interconnects.

2 | ISAMBARD: SYSTEM OVERVIEW

The Isambard system is a full cabinet of XC50 “Scout” with Marvell ThunderX2 CPUs, delivering 10 752 high-performance Armv8 cores. Each node includes two 32-core ThunderX2 processors running at a base clock speed of 2.1 GHz, and a turbo clock speed of 2.5 GHz. The processors each have eight 2666 MHz DDR4 channels, yielding a measured STREAM triad bandwidth of 244 GB/s per node, and a DRAM capacity of 256 GB. The XC50 Scout system integrates four dual-socket nodes into each blade, and then 42 such blades into a single cabinet. One blade of four nodes is reserved to act as head nodes for the rest of the system, leaving 164 compute nodes, or 10 496 compute cores. Pictures of a Scout blade and Isambard's XC50 cabinet are shown in Figure 1.

The results presented in this paper are based on work performed since April 3, 2019, when an upgrade to Isambard's hardware and software was completed. Specifically, Isambard's Marvell ThunderX2 CPUs were upgraded to B2 silicon stepping, the latest version of CLE (7.0) based on SLES 15 was installed, as was new firmware which enabled the ThunderX2's turbo mode for the first time. The Cray Programming Environment includes Arm versions of all the software components we needed for benchmarking the ThunderX2 CPUs: the Cray Compiling Environment (CCE), performance libraries, and analysis tools. In addition to Cray's compiler, Isambard also provides Arm's Clang/LLVM-based HPC Compiler, and GNU Compiler Collection (GCC).

[§] https://www.hpe.com/us/en/newsroom/press-release/2018/04/academia-and-industry-collaborate-to-drive-uk-supercomputer-adoption.html

3 | CATALYST: SYSTEM OVERVIEW

The Catalyst system comprises two racks containing 64 HPE Apollo 70 nodes, totaling 4096 Marvell ThunderX2 cores. As with Isambard, each node is a dual-socket 32-core ThunderX2 system (64 cores per node), with 16 DDR4 dual inline memory modules providing 256 GB of dynamic random access memory (DRAM) capacity. The base clock of the Apollo 70 nodes is 2.2 GHz, and turbo is not currently active, though may be enabled with a future firmware update. Due to the difference in clock speeds, we observe a slightly lower STREAM Triad bandwidth of 232 GB/s on our Catalyst system, compared to Isambard. Each node contains a 100-Gb/s Mellanox InfiniBand card, connecting to a nonblocking fat tree network. The full system, including additional nodes used for service, login, and storage, consumes approximately 30 kW of power.

The system provides toolchains using both GCC and Arm's HPC Compiler (based on LLVM), along with Arm's Allinea performance analysis and debugging tools. We primarily use HPE's HMPT implementation of message passing interface (MPI), although we also have OpenMPI available for use when necessary. The system is running SLES 12 SP3, although will be upgraded to SLES 15 in due course.

4 | BENCHMARKING RESULTS

4.1 | Platforms

For each benchmark, we show scaling results for both the Isambard and Catalyst systems. Aside from the differences in interconnect (Aries versus InfiniBand), the other main difference between these systems at the hardware level is the clock speed. Our Catalyst system currently has a fixed clock speed of 2.2 GHz. On Isambard, the processors run with a 2.1-GHz base clock speed, and a 2.5-GHz turbo clock speed. We should note that, in our testing, all the Isambard CPUs appear to run at the 2.5-GHz turbo speed all of the time, no matter what code or benchmark we ran, including HPL. The ThunderX2 processors support 128-bit vector Arm Advanced single-instruction multiple-data (SIMD) instructions (sometimes referred to as "NEON"), and each core is capable of four-way simultaneous multithreading (SMT), for a total of up to 256 hardware threads per node. On Isambard, the compute nodes are booted with all four hardware threads enabled, whereas the Catalyst system is booted using only two hardware threads per core. The processor's on-chip data cache is organized into three levels: a private L1 and L2 cache for each core, and a 32 MB L3 cache shared between all the cores. Finally, each ThunderX2 socket utilizes eight separate DDR4 memory channels running at up to 2666 MHz.

The Cray XC40 supercomputer "Swan" was used for access to Intel Broadwell and Skylake processors, with an additional internal Cray system, "Horizon", providing access to a more mainstream SKU of Skylake:

- Intel Xeon Platinum 8176 (Skylake) 28-core @ 2.1 GHz, dual socket, with 192 GB of DDR4-2666 DRAM. RRP $8719 each.
- Intel Xeon Gold 6148 (Skylake) 20-core @ 2.4 GHz, dual socket, with 192 GB of DDR4-2666 DRAM. RRP $3072 each.
- Intel Xeon E5-2699 v4 (Broadwell) 22-core @ 2.2 GHz, dual socket, with 128 GB of DDR4-2400 DRAM. RRP $4115 each.

The RRPs were correct at the time of writing, and taken from Intel's website at that time.[¶] The Skylake processors provide an AVX-512 vector instruction set, meaning that each FP64 vector operation processes eight elements at once; by comparison, Broadwell utilizes AVX2, which is 256 bits wide, simultaneously operating on four FP64 elements at a time, per SIMD instruction. The Xeon processors all have three levels of on-chip (data) cache, with an L1 and L2 cache per core, along with a shared L3. This selection of CPUs provides coverage of both the state-of-the-art and the *status quo* of current commonplace HPC system design. We include high-end models of both Skylake and Broadwell to make the comparison as challenging as possible for ThunderX2. It is worth noting that in reality, most Skylake and Broadwell systems will use SKUs from much further down the range, of which the Xeon Gold part described above is included as a good example. This is certainly true for the current TOP500 systems.

A summary of the hardware used, along with peak floating-point and memory bandwidth performance, is shown in Table 1, while a chart comparing key hardware characteristics of the main CPUs in our test (the three near-top-of-bin parts: Broadwell 22c, Skylake 28c, and ThunderX2 32c) is shown in Figure 2. We use McCalpin's STREAM benchmark for measuring the achievable sustained memory bandwidth of each platform,[4] using arrays of 2^{25} double-precision elements, with the kernels run 200 times. The number reported in the chart corresponds to the achieved bandwidth for the Triad kernel. To measure the sustained cache bandwidths, we used the methodology described in our previous work,[5] utilizing BabelStream.[6] The Triad kernel from the BabelStream benchmark was run in a tight loop on each core simultaneously, with problem sizes selected to ensure residency in each level of the cache. The bandwidth is then modeled using the array size, number of iterations, and the time for the benchmark to run. This portable methodology was previously shown to attain the same performance as handwritten benchmarks, which only work on their target architectures.[7]

There are several important characteristics that are worthy of note. First, the wider vectors in the x86 CPUs give them a significant peak floating-point advantage over ThunderX2. Second, wider vectors also require wider data paths into the lower levels of the cache hierarchy. This results in the x86 CPUs having an L1 cache bandwidth advantage, but we see the advantage reducing as we go up the cache levels, until once at external memory, it is ThunderX2 which has the advantage, due to its greater number of memory channels. Third, as seen in most benchmark studies in recent years, dynamic voltage and frequency scaling makes it harder to reason about the percentage of peak performance that is being

[¶] https://ark.intel.com/

TABLE 1 Hardware information (peak figures)

Processor	Cores	Clock speed GHz	TDP Watts	FP64 TFLOP/s	Bandwidth GB/s
Broadwell	2 × 22	2.2	145	1.55	154
Skylake Gold	2 × 20	2.4	150	3.07	256
Skylake Platinum	2 × 28	2.1	165	3.76	256
ThunderX2	2 × 32	2.1 (2.5)	175	1.28	320

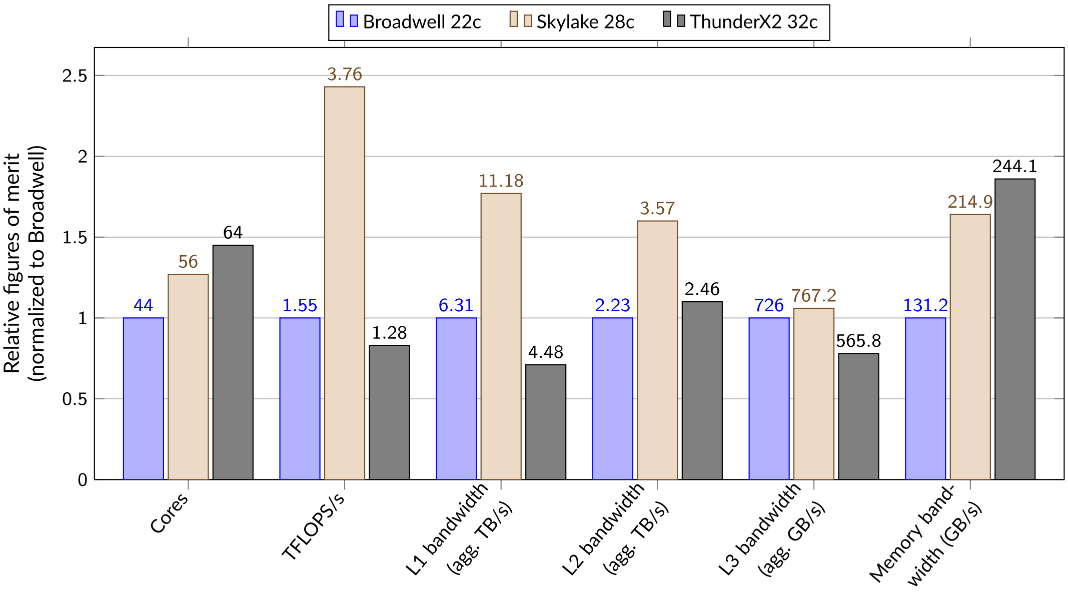


FIGURE 2 Comparison of properties of Broadwell 22c, Skylake 28c, and ThunderX2 32c. Results are normalized to Broadwell

TABLE 2 Best performing compilers determined during benchmarking

Benchmark	Broadwell	Skylake	Isambard	Catalyst
CloverLeaf	Intel 2019	Intel 2019	CCE 9.0	GCC 8.2
TeaLeaf	Intel 2019	Intel 2019	GCC 8.3	Arm 19.2
SNAP	Intel 2019	Intel 2019	CCE 9.0	GCC 8.2
GROMACS	GCC 8.3	GCC 8.3	Arm 19.2	Arm 19.2
OpenFOAM	GCC 7.3	GCC 7.3	GCC 7.3	GCC 7.1
OpenSBLI	CCE 9.0	GCC 8.3	GCC 8.3	Arm 19.2
VASP	Intel 2019	Intel 2019	GCC 7.3	-

achieved. For example, while measuring the cache bandwidth results shown in Figure 2, we observed that our Broadwell 22c parts consistently increased their clock speed from a base of 2.2 GHz up to 2.6 GHz. In contrast, our Skylake 28c parts consistently *decreased* their clock speed from a base of 2.1 GHz down to 1.9 GHz, a 10% reduction in clock speed. By comparison, during all our tests, Isambard's ThunderX2 CPUs ran at a consistent 2.5 GHz, their turbo speed, which was 19% faster than their base clock speed of 2.1 GHz. At the actual, measured clock speeds, the fraction of theoretical peak bandwidth achieved at L1 for Broadwell 22c, Skylake 28c, and ThunderX2 32c, was 57%, 55%, and 58%, respectively.

To evaluate the state of the software ecosystem for Arm, we used all three compiler families available to us: GCC, the LLVM-based Arm HPC Compiler, and Cray's CCE. The Isambard node-level performance paper at CUG 2018 was the first study to compare all three of these compilers targeting Arm.[1] Likewise, for the Intel processors, we used GCC, the Intel compilers, and Cray's CCE. For benchmarks that make use of Basic Linear Algebra Subprogram (BLAS) and fast Fourier transform (FFT) routines, we also compared the difference between Cray's LibSci, FFTW, and the Arm performance libraries. For each benchmark, the toolchain that achieved the highest performance for the largest node count was used in the results graphs displayed below (see Table 2). It should be noted that all of these compilers and libraries are still relatively early in their support for HPC-optimized Arm CPUs, and we continue to observe significant performance improvements with each new release of these tools. Section 4.5 compares the performance of the different toolchains in detail.

Most of the scaling results we show in the rest of this paper scale up to 32 nodes. This limit was imposed by the node counts in the x86 Swan systems that we compare to. With the exception of SNAP (which has very high memory usage), all of the results are produced by strong-scaling a single input problem. In most cases, we begin scaling from a single node; however, for GROMACS and OpenSBLI, we start from either two or four nodes due to memory and runtime constraints. Where an application presents a hybrid MPI/OpenMP implementation, we tune the number of MPI ranks per node for the highest node count on each platform separately.

4.2 | Mini-apps

In this section, we evaluate the performance of three mini-apps, scaling up to 32 nodes on each target platform. The mini-apps themselves are all performance proxies for larger production codes, encapsulating important performance characteristics such as floating-point intensity, memory access, and communication patterns of their parent applications, but without the complexities that are often associated with “real” codes. As such, they are useful for performance modeling and algorithm characterization, and can demonstrate the potential performance of the latest computer architectures.

4.2.1 | CloverLeaf

CloverLeaf is a hydrodynamics mini-app that solves Euler's equations of compressible fluid dynamics, under a Lagrangian-Eulerian scheme, on a two-dimensional spatial regular structured grid.[8] These equations model the conservation of mass, energy, and momentum. The mini-app is an example of a stencil code and is known to be memory bandwidth bound. CloverLeaf is regularly used to study performance portability on many different architectures.[9] The `bm_256` test case that we used here consists of a grid of 15360 × 15360 cells and is suitable for strong-scaling up to a system the size of Isambard. CloverLeaf is a member of the Mantevo suite of mini-apps from Sandia National Laboratory.[10]

The normalized results for the CloverLeaf mini-app in Figure 3A are consistent with those for STREAM on low node counts. CloverLeaf is a structured grid code, and the majority of its kernels are bound by the available memory bandwidth. It has been shown previously that the memory bandwidth increases from GPUs result in proportional improvements for CloverLeaf.[9] The same is true on the processors in this study, with the single-node improvements on ThunderX2 coming from its greater memory bandwidth. Therefore, for structured grid codes, we indeed see that the runtime is proportional to the external memory bandwidth of the system, and the ThunderX2 provides the highest bandwidth out of the processors tested. At higher node counts, the relative performance changes slightly due to the impact of communication overheads, with the end result being that both SKUs of Skylake and the ThunderX2 CPUs all perform similarly well at scale; the parallel efficiency graph in Figure 3B shows how both the x86 and Arm-based platforms scale similarly for CloverLeaf. Comparing the scaling of the two Arm-based systems, we see that the Aries-based Isambard scales better than the Infiniband-based Catalyst system. For a well-behaved structured grid code such as CloverLeaf, we would expect most systems to scale similarly well, so this is an unexpectedly large difference. We therefore believe that these differences are likely due to known factors, such as the MPI tuning parameters on our Catalyst system, which are still immature.

4.2.2 | TeaLeaf

TeaLeaf is a heat diffusion mini-app that solves the linear heat conduction equation on a spatially decomposed regular grid, utilizing a five-point finite difference stencil.[11] A range of linear solvers is included in the mini-app, but the baseline method we use in this paper is the matrix-free conjugate gradient solver. TeaLeaf is memory bandwidth bound at the node level, but, at scale, the solver can become bound by communication. We used the `bm_5` input deck for the strong-scaling tests in this paper, which represents the largest mesh size that is considered to be scientifically interesting for real-world problems (as discussed by McIntosh-Smith et al[11]). This utilizes a 4000 × 4000 spatial grid, running for ten timesteps. Like CloverLeaf, TeaLeaf is also a member of the Mantevo mini-app suite.[10]

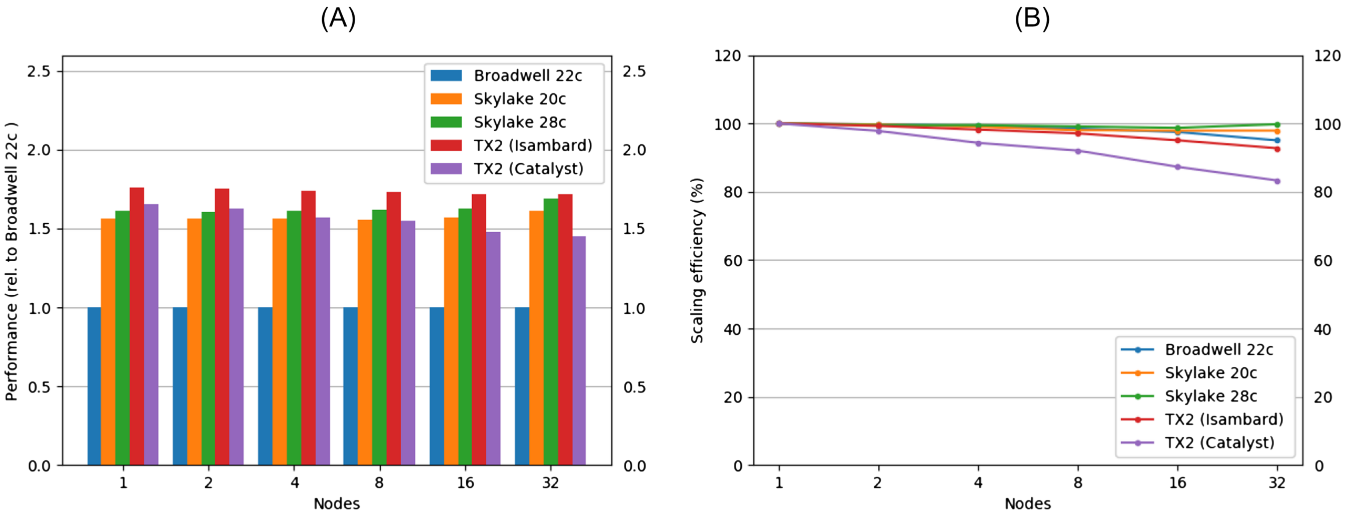


FIGURE 3 CloverLeaf scaling results up to 32 nodes

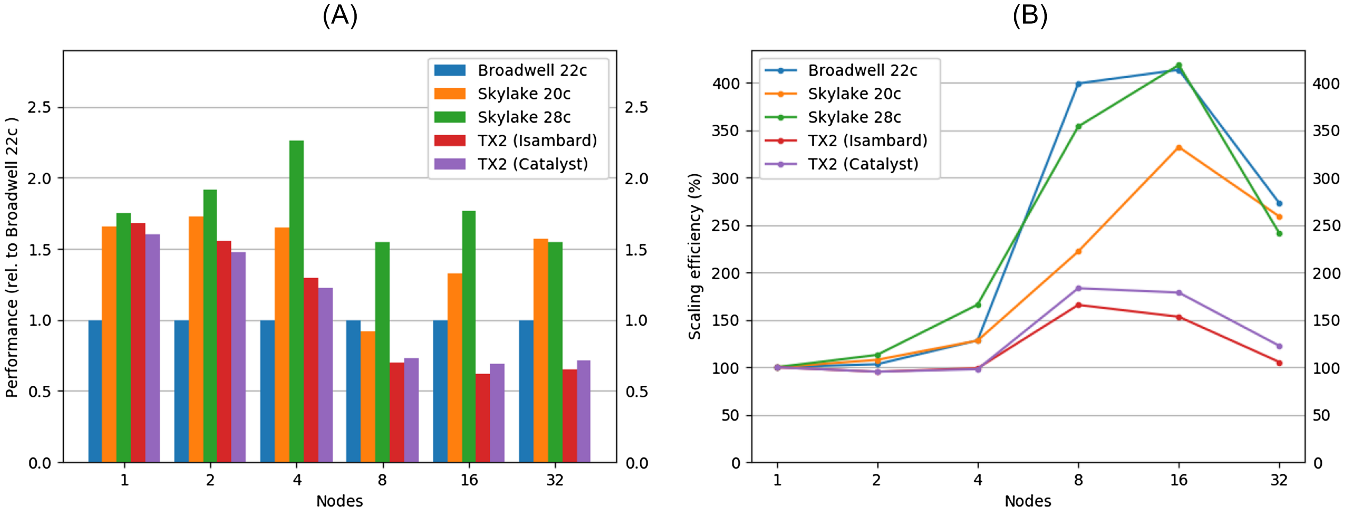


FIGURE 4 TeaLeaf scaling results up to 32 nodes

Figure 4A compares the performance of the TeaLeaf mini-app between the four systems up to 32 nodes. The relative performance results on a single node are similar to those presented by McIntosh-Smith et al,[12] with small differences arising from the use of newer compilers and a different platform for ThunderX2. TeaLeaf is largely dominated by DRAM memory bandwidth on a single node, which is reflected in these results wherein the Isambard nodes are 80% faster than Broadwell and similar in performance to the Skylake systems. At scale, however, the x86 systems show a performance advantage for the problem that we are using.

As shown in Figure 4B, all platforms achieve superlinear scaling behavior up to around 16 nodes which, as observed by McIntosh-Smith et al,[11] is due to cache effects of strong-scaling over a relatively small data set. The superlinear improvement is much less pronounced on ThunderX2 than with the x86 systems, which is primarily due to the much smaller ratio of DRAM to L3 cache bandwidth (as shown in Figure 2) and a smaller L3 cache capacity. In addition, the overheads of some of the MPI communication routines such as the halo exchange and `MPI_AllReduce` operations appear to be greater on ThunderX2, further impacting scalability. As a result of this, Isambard and Catalyst end up at around 2× slower than the x86 systems when using 32 nodes. For the two Arm-based systems, the situation is reversed relative to CloverLeaf; here, the Catalyst system scales slightly better than Isambard. There are a number of reasons why this is the case: Aries favors small (64-byte) messages versus IB's preference for longer messages, and for CloverLeaf, the latter is a small advantage. In addition, Aries has a hardware-accelerated `MPI_AllReduce` feature, which is not enabled on Isambard by default. At the scale of the results shown in Figure 4, most of the time is spent in the dot product, relying on this all reduced feature. Experiments performed by Cray showed that by tuning the MPI rank placement, turning on Aries' hardware reduction engine, and tweaking Cray MPI's E0 threshold to 8KB so that TeaLeaf's short messages are covered, Isambard's performance for TeaLeaf on 64 nodes improved significantly. It should be noted that these gains also benefited the Broadwell and Skylake systems in our test, but to a lesser degree. Because we wanted to focus on the kind of performance that most users would experience “out of the box”, we did not use Cray's improved TeaLeaf performance results in Figure 4.

4.2.3 | SNAP

SNAP is a proxy application for a modern deterministic discrete ordinates transport code.[13] As well as having a large memory footprint, this application has a simple finite difference kernel, which must be processed according to a wavefront dependency, which introduces associated communication costs. SNAP is unique in the applications in this study in that parallelism is exposed in two levels: spatially with MPI and over the energy domain utilizing OpenMP. As such and as is common within the transport community, we have opted to explore the weak scalability of this application. We have used a problem size of $1024 \times 12 \times 12$ cells per MPI rank, with 32 energy groups and 136 angles per octant, chosen to fit within the memory capacity of our baseline Broadwell system. We run with one MPI rank per socket, and use OpenMP threads to saturate all cores of the socket. This configuration differs from our previous single-node analysis of this mini-app on ThunderX2 processors[1] but is representative of running at scale where spatial concurrency becomes limited.

Running the weak scaling setup described above, the runtimes at all scales are similar across all the architectures tested, as shown in Figure 5A, with the advantage initially seen on Skylake reducing at higher node counts. Our earlier analysis of the scalability of SNAP showed that, even at relatively modest node counts such as those used in this study, the runtime is dominated by network communications.[14] Therefore, a similar level of performance can be seen at modest scale irrespective of the architecture. The Catalyst system however performs notably worse than the others, which is attributed to the inability of GCC to efficiently vectorize the main computational kernel (on Isambard, we were able to use the Cray compiler instead). Figure 5B also shows that each system has a very similar parallel efficiency up to eight nodes and follows a similar trend to our previous work.[14] While in the x86 systems we were using in these tests only had 32 nodes, we were able to scale up the SNAP run to a higher node count on Isambard. At these higher node counts, the ThunderX2 scaling efficiency settles at around 80%.

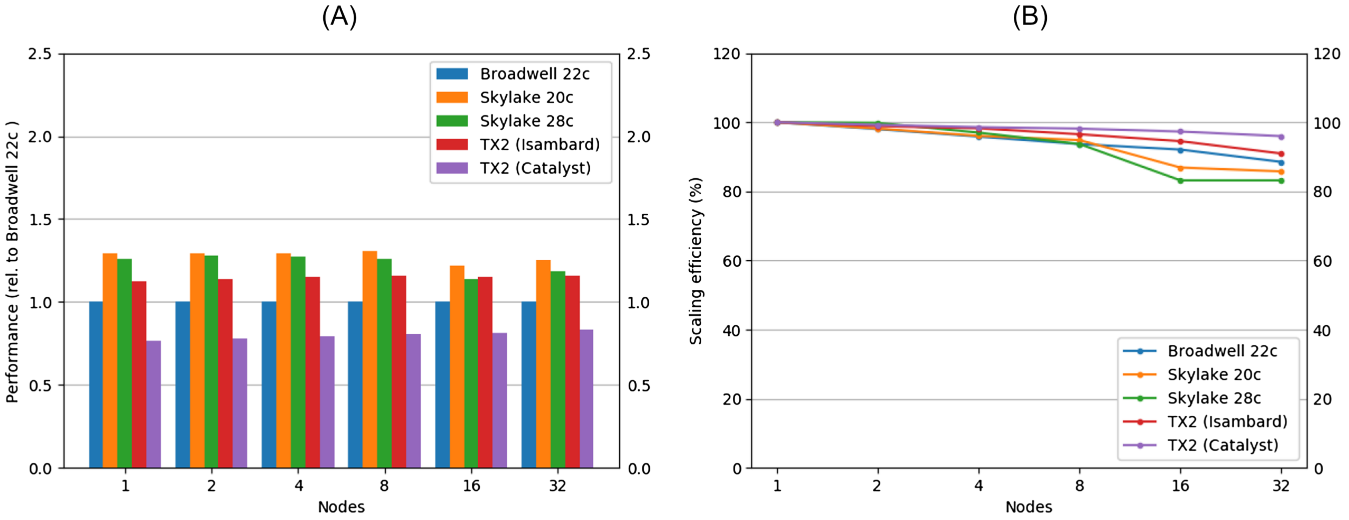


FIGURE 5 SNAP scaling results up to 32 nodes

4.3 | Scientific applications

The Isambard system has been designed to explore the feasibility of an Arm-based system for real HPC workloads on national services, such as ARCHER, the UK's National Supercomputer for researchers funded by the EPSRC.[3] To that end, a number of real applications have been selected for this study taken from the top ten most used codes on ARCHER, The applications we have selected represent over 30% of the usage of the whole ARCHER system, in terms of node hours. Therefore, the performance of these codes on any architecture captures the interests of a significant fraction of UK HPC users, and any changes in the performance of these codes directly from the use of different architectures is important to quantify. The test cases were chosen by the group of core application developers and key application users who came to two Isambard hackathons held in October 2017 and February 2018; details of the attendees are found in the Isambard paper from CUG 2018, which focused on single node performance.[1] Given that we wish to focus on scaling across the Isambard and Catalyst systems, we had to choose test cases that were of scientific merit, yet that could be scaled from a single node up to a useful fraction of both machines.

4.3.1 | GROMACS

GROMACS is a widely used molecular dynamics package that is used to solve Newton's equations of motion.[#] Systems of interest, such as proteins, can contain up to millions of particles. It is thought that GROMACS is usually bound by the floating-point performance at low node counts, while becoming communication bound at higher node counts. The FLOP/s–bound nature of GROMACS at low node counts motivated the developers to handwrite vectorized code using compiler intrinsics in order to ensure an optimal sequence of these operations.[15] For each supported platform, computation is packed so that it saturates the native vector length of the platform, eg, 256 bits for AVX2, 512 bits for AVX-512, and so on. For this study, we used a 42 million atom test case from the ARCHER benchmark suite,[16] running for 800 time steps. On the ThunderX2 processors, we used the 128-bit `ARM_NEON_ASIMD` vector implementation, which is the closest match for the underlying Armv8.1-A architecture. We note that, within GROMACS, this NEON SIMD implementation is not as mature as the SIMD implementations targeting x86.

Figure 6a shows that at low node counts, GROMACS performance for this benchmark correlates to floating-point throughput and L1 cache bandwidth. At two nodes, Skylake Platinum is 1.62× faster than Broadwell, while Isambard is 1.22× slower. The Catalyst system is even slower again, due to the lower clock speeds delivering less cache bandwidth and lower FLOP/s. As the node count increases, the performance becomes increasingly affected by communication costs. Figure 6B shows that the scaling efficiency drops to below 60% for Skylake Platinum at 32 nodes, with MPI communications accounting for 72% of the total runtime. Since the node-level performance is lower, Isambard is able to achieve a scaling efficiency of 90% for 32 nodes, and Catalyst reaches close to 100%. As a result of this, Isambard achieves performance almost on par with the Skylake Gold SKU when using 64 nodes, making up for the lower floating-point throughput and cache bandwidth.

4.3.2 | OpenFOAM

OpenFOAM is a modular C++ framework aiming to simplify writing custom computational fluid dynamics solvers.[17] In this paper, we use the `simpleFoam` solver for incompressible, turbulent flow from version 1712 of OpenFOAM[‖], the most recent release at the time we began benchmarking the Isambard system. The input case is based on the RANS Driver generic car model, which is a representative case of real aerodynamics simulation and thus should provide meaningful insight of the benchmarked platforms' performance.[18] The decomposed grid consists of approximately 64 million cells. OpenFOAM is memory bandwidth bound, at least at low node counts.

[#] http://www.gromacs.org
[‖] https://www.openfoam.com/download/install-source.php

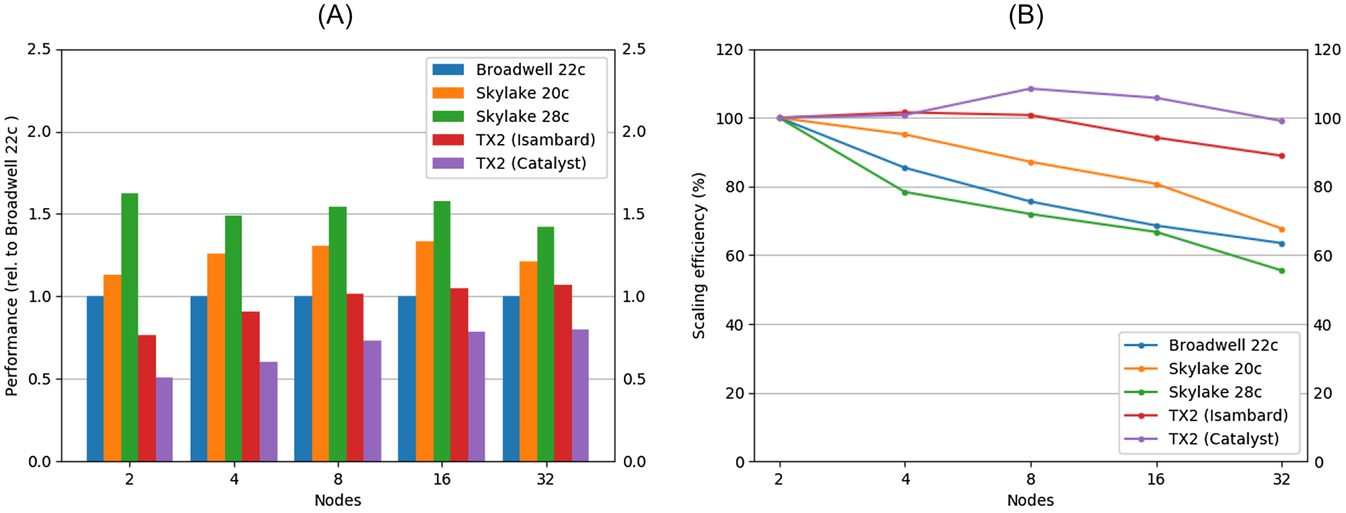


FIGURE 6 GROMACS scaling results up to 32 nodes

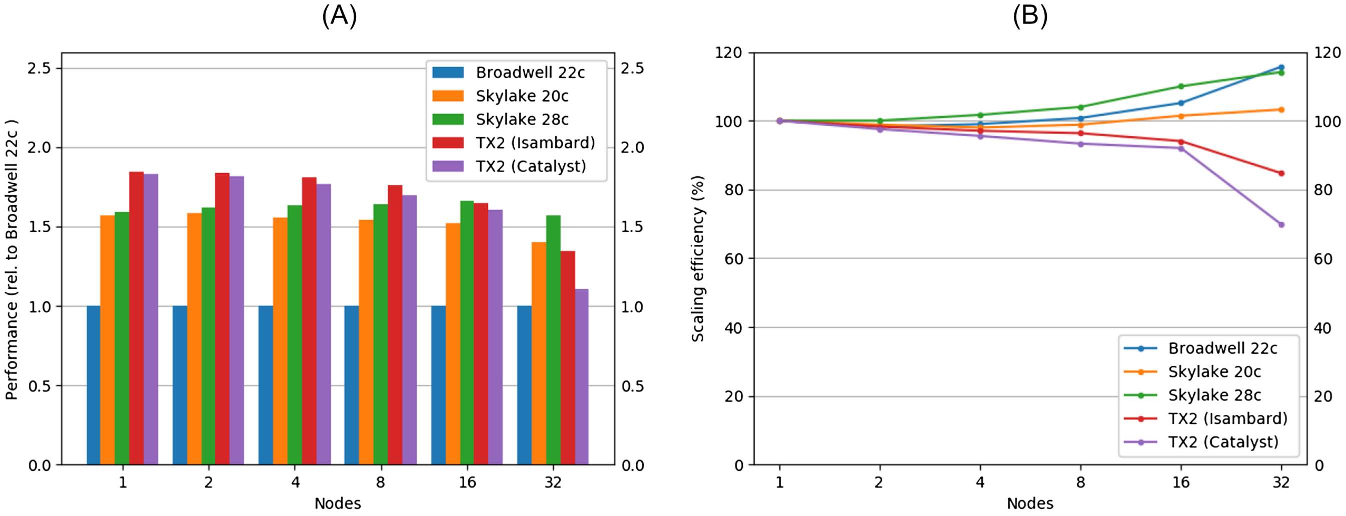


FIGURE 7 OpenFOAM scaling results up to 32 nodes

The OpenFOAM results shown in Figure 7A start off following the STREAM behavior of the three platforms closely, confirming that memory bandwidth is the main factor that influences performance at low node counts. With its eight memory channels, ThunderX2 yields the fastest result, at 1.83× the Broadwell performance on four nodes, compared to 1.57× and 1.59× on Skylake 20c and 28c, respectively. At higher node counts, other factors come into play, where in Figure 7B, we see Broadwell scaling the best of all the platforms, Skylake also maintaining good scaling, and the ThunderX2 systems scaling the least well, with parallel efficiency dropping to below 85%. We suspect that, as with TeaLeaf, the lower cache bandwidths on the ThunderX2 processors limit their ability to realize a superlinear speedup for kernels that begin to work out of cache, impacting overall scalability compared to the x86 systems.

4.3.3 | OpenSBLI

OpenSBLI is a grid-based finite difference solver[**] used to solve compressible Navier-Stokes equations for shock-boundary layer interactions. The code uses Python to automatically generate code to solve the equations expressed in mathematical Einstein notation, and uses the Oxford Parallel Structured software for parallelism. As a structured grid code, it should be memory bandwidth bound under the Roofline model, with low computational intensity from the finite difference approximation. We used the ARCHER benchmark for this paper,[††] which solves a Taylor-Green vortex on a grid of 1024 × 1024 × 1024 cells (around a billion cells). On each system we ran with one MPI rank per core, without using SMT.

The scaling efficiency for OpenSBLI, shown in Figure 8B, is similar across the four systems tested. At low node counts, performance of the OpenSBLI benchmark is dominated by bandwidth to DRAM and L3 cache. The ThunderX2 systems are around 1.5× faster than Broadwell at four nodes, and just a few percent slower than Skylake Platinum (see Figure 8A). Each system sustains efficiency above 80% up to 32 nodes.

[**] https://opensbli.github.io
[††] http://www.archer.ac.uk/community/benchmarks/archer/

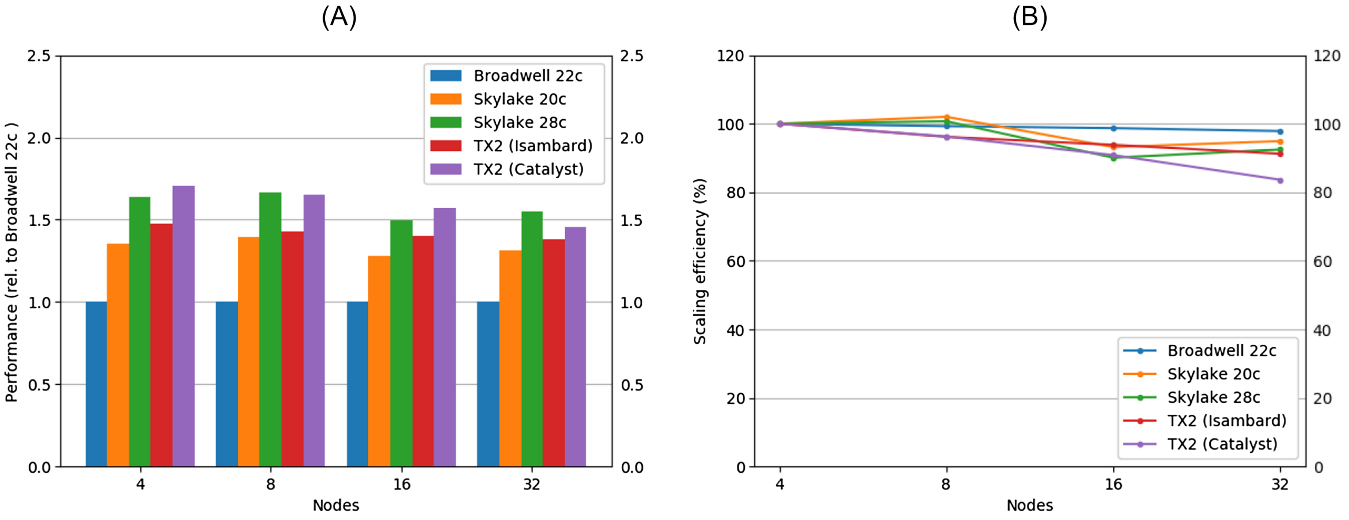


FIGURE 8 OpenSBLI scaling results up to 32 nodes

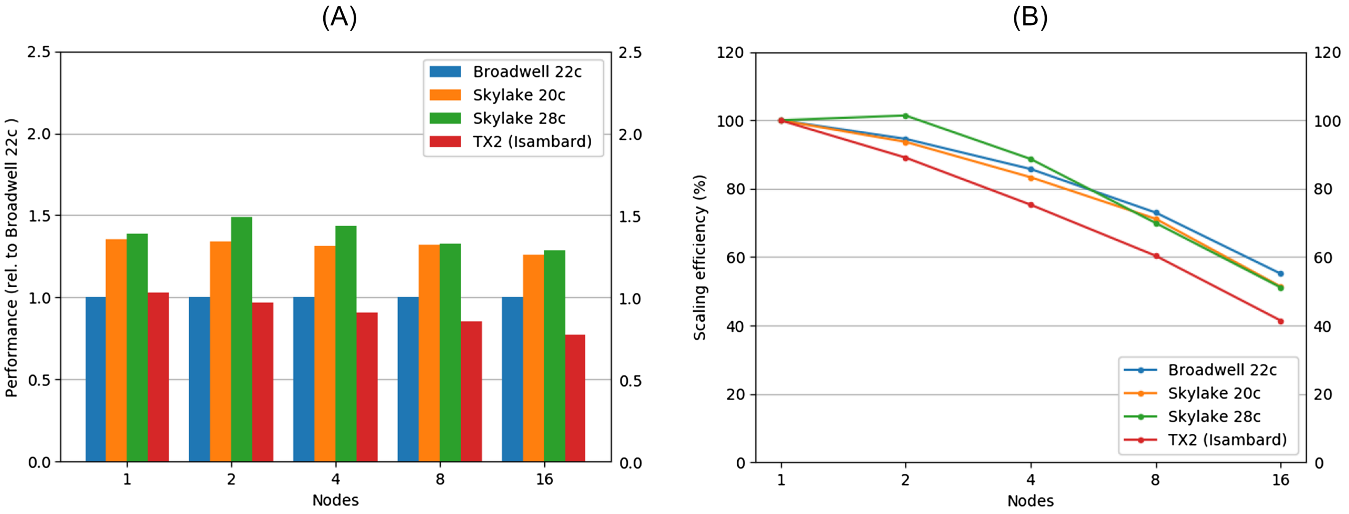


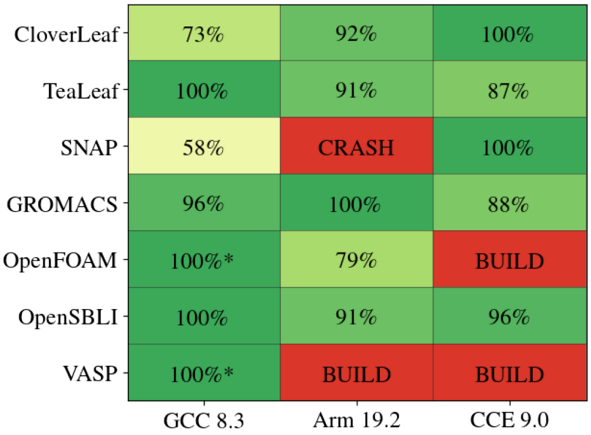
FIGURE 9 VASP scaling results up to 16 nodes

4.3.4 | VASP

The Vienna Ab initio Simulation Package[‡‡] (VASP) is used to model materials at the atomic scale, in particular performing electronic structure calculations and quantum-mechanical molecular dynamics. It solves the N-body Schrödinger equation using a variety of solution techniques. VASP includes a significant number of settings that affect performance, from domain decomposition options to math library parameters. Previous investigations have found that VASP is bound by floating-point compute performance at scales of up to a few hundred cores. For bigger sizes, its heavy use of MPI collectives begins to dominate, and the application becomes bound by communication latency.[19] The benchmark utilized is known as `PdO`, because it simulates a slab of palladium oxide. It consists of 1392 atoms, and is based on a benchmark that was originally designed by one of VASP's developers, who found that (on a single node) the benchmark is mostly compute-bound; however, there exist a few methods that benefit from increased memory bandwidth.[20] We ran with one MPI rank per core, without using SMT. We tuned the value of `NCORE`, a parameter which describes the parallel decomposition, for 16 nodes on each platform separately.

The scaling efficiency for VASP, shown in Figure 9B, is similar across the four systems tested. At 16 nodes, the ThunderX2 and Skylake systems are all below 60% efficiency, with up to half of the total runtime consumed by the MPI communication. The remainder of the runtime is split between DGEMM and 3D-FFT routines, which favor the higher floating-point throughput and cache bandwidth of the x86 processors with their wider vector units. The net result (shown in Figure 9A) is that, at 16 nodes, Isambard is a 1.29× slower than the Broadwell system, and 1.62 – 1.66× slower than the Skylake systems.

[‡‡] http://www.vasp.at



	GCC 8.3	Arm 19.2	CCE 9.0
CloverLeaf	73%	92%	100%
TeaLeaf	100%	91%	87%
SNAP	58%	CRASH	100%
GROMACS	96%	100%	88%
OpenFOAM	100%*	79%	BUILD
OpenSBLI	100%	91%	96%
VASP	100%*	BUILD	BUILD

FIGURE 10 Efficiency of different compilers running on Isambard. The `BUILD` and `CRASH` labels denote configurations that either failed to build or crashed at runtime, respectively. A * indicates the use of GCC 7.3, due to build failures with GCC 8.3

4.4 | Performance summary

Overall, the results presented in this section demonstrate that the Arm-based Marvell ThunderX2 processors are able to execute a wide range of important scientific computing workloads with performance that is competitive with state-of-the-art x86 offerings. At lower node counts, the ThunderX2 processors can provide significant performance improvements when an application's performance is limited by external memory bandwidth, but are slower in cases where codes are compute-bound. At higher node counts, the differences between node-level peak bandwidth or FLOP/s often become less significant, with the network becoming the limiting factor. Given that, by design, four of the systems in our comparison are Aries-based XC machines, one would expect to see performance between the systems converge, and this is indeed what we observe in most cases. For the codes where we observed that the Arm-based systems do not scale as well as the x86-based ones, such as TeaLeaf and OpenFOAM, we believe the lower cache bandwidth on the ThunderX2 CPUs is contributing to the lower performance (as strong-scaled workloads start fitting into the cache hierarchy), and this is something we expect to see addressed with future generations of Arm-based processors. The important conclusion is that Arm-based supercomputers can perform as well as x86-based ones at scale. The fact that the Arm-based processors may be significantly more cost effective than x86-based ones therefore makes them an attractive option.

4.5 | Toolchain comparison

Figure 10 compares the latest available versions of the three compilers on Isambard, normalized to the best performance observed for each benchmark, running on 32 nodes (16 for VASP). There are three cases that fail to build: OpenFOAM with CCE 9.0 and VASP with Arm 19.2 and CCE 9.0. It is worth noting that none of these issues appear to be specific to Arm platforms; CCE 9.0 fails to build OpenFOAM and VASP on x86 systems as well, and the Flang front end from which the Arm 19.2 Fortran compiler is derived also fails to build VASP. GCC 8 raises syntax errors in both OpenFOAM and VASP, indicating that these build failures may be issues with the applications rather than compiler bugs. In addition, the Arm 19.2 build of SNAP crashes at runtime, which is attributed to excessive stack usage in the SNAP code itself.

The largest performance difference between compilers is observed with SNAP, for which performance depends heavily on efficient vectorization of a fairly complex kernel. The Cray compiler yields a 1.7× improvement over GCC here, giving an advantage to Isambard over the HPE Catalyst system as noted in Section 4.2.3. Conversely, GCC provides a 15% improvement over CCE for TeaLeaf, generating more efficient code for the core solver routine. Interestingly however, for TeaLeaf at low node counts, the Cray compiler is the fastest, with a much faster reduction operation negating the slower solver routine. Overall, GCC is the fastest for four out of seven benchmarks, with the Cray compiler fastest for two and the Arm HPC compiler only fastest for GROMACS, where it just edges out GCC. Table 2 shows the best compilers used for each benchmark for all the systems used in this study, indicating that no single compiler dominates performance on x86 platforms either.

We also explored the use of the Arm Performance Libraries (ArmPL) for GROMACS and VASP, comparing to Cray's LibSci and tuned FFTW libraries that are the default on Cray systems. For GROMACS, using ArmPL instead of FFTW resulted in a small drop in performance at all node counts. For VASP, using ArmPL instead of LibSci for the BLAS routines resulted in a small but consistent improvement. We were unable to get VASP working when using ArmPL for the FFT routines however, and are still investigating this.

5 | REPRODUCIBILITY

With an architecture such as Arm, which is new to mainstream HPC, it is important to make any benchmark comparisons as easy to reproduce as possible. To this end, the authors are making all of the detailed information about how each code was compiled and run, along with the input parameters to the test cases, available as an open source repository on GitHub.[§§] The build scripts will show which compilers were used in each

[§§] https://github.com/UoB-HPC/benchmarks/releases/tag/CCPE-CUG-2019 case, what flags were set, and which math libraries were employed. The run scripts will show which test cases were used, and how the runs were parameterized. These two sets of scripts should enable any third party to reproduce our results, provided that they have access to similar hardware. Many of the scripts do assume a Cray-style system, but should be easily portable to other versions of Linux on non-Cray systems, as evidenced by the inclusion of scripts for our HPE Catalyst system.

6 | CONCLUSIONS

The results presented in this paper demonstrate that supercomputers built using Arm-based processors are now providing production performance competitive with state-of-the-art offerings from the incumbent processor vendors. We found that, even in cases where x86-based CPUs with higher peak floating point performance would beat ThunderX2 at low node counts, at realistic scales appropriate for real science runs, ThunderX2 often become even more competitive, due to its greater memory bandwidth benefiting communication performance. We also saw that most codes scaled similarly between x86 and ThunderX2, and between two ThunderX2 systems with different interconnect technologies. With future Arm-based processors set to increase floating point performance and cache bandwidth with the introduction of new technologies such as the Arm scalable vector extensions, the CPU offerings from these vendors are set to become even more compelling. The majority of our benchmarks compiled and ran successfully *out-of-the-box*, and no architecture-specific code tuning was necessary to achieve high performance. This represents an important milestone in the maturity of the Arm ecosystem for HPC, where these processors can now be considered as viable contenders for future production procurements.

Overall, these results demonstrate that Arm-based server CPUs that have been optimized for HPC are now genuine options for production systems, offering performance at scale competitive with best-in-class CPUs, while potentially offering attractive price/performance benefits. With multiple systems vendors now offering solutions using these CPUs as part of their standard product portfolio, Arm-based processors look to become part of the future of the scientific computing landscape.

ACKNOWLEDGMENTS

As the world's first production Arm supercomputer, the GW4 Isambard project could not have happened without support from a lot of people. First, the co-investigators at the four GW4 universities, the Met Office, and Cray who helped to write the original proposal, including Prof James Davenport (Bath), Prof Adrian Mulholland (Bristol), Prof Martyn Guest (Cardiff), Prof Beth Wingate (Exeter), Dr Paul Selwood (Met Office), and Adrian Tate (Cray). Second, the operations group who designed and now run the Isambard system, including Steven Chapman and Roshan Mathew (Bath) and Christine Kitchen and James Green (Cardiff); Dave Acreman, Rosie Rowlands, Martyn Brake, and John Botwright (Exeter); Simon Burbidge and Chris Edsall (Bristol); David Moore, Guy Latham, and Joseph Heaton (Met Office); and Steven Jordan and Jess Jones (Cray). And finally, to the attendees of the first two Isambard hackathons, who did most of the code porting that underpins the results in this paper, and to Federica Pisani from Cray who organized these events. For the lists of attendees of the hackathons in November 2017 and March 2018, please see the work of McIntosh-Smith et-al.[1] Access to the Cray XC50 supercomputer "Swan" was kindly provided by Cray Inc's Marketing Partner Network. The Isambard project is funded by EPSRC, the GW4 Alliance, the Met Office, Cray, and Arm. The Isambard project is funded by EPSRC research grant EP/P020224/1. Further Isambard-related research was funded by the ASiMoV EPSRC prosperity partnership project, grant EP/S005072/1.

ORCID

Simon McIntosh-Smith https://orcid.org/0000-0002-5312-0378
Tom Deakin https://orcid.org/0000-0002-6439-4171

REFERENCES

1. McIntosh-Smith S, Price J, Deakin T, Poenaru A. Comparative benchmarking of the first generation of HPC-optimised arm processors on Isambard. Paper presented at: 2018 Cray User Group Meeting; 2018; Stockholm, Sweden.
2. McIntosh-Smith S, Price J, Poenaru A, Deakin T. Scaling results from the first generation of arm-based supercomputers. Paper presented at: 2019 Cray User Group Meeting; 2019; Montreal, Canada.
3. Turner A, McIntosh-Smith S. A Survey of Application Memory Usage on a National Supercomputer: An Analysis of Memory Requirements on ARCHER. In: Stephen J, Steven W, Simon W, eds. *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*. Cham, Switzerland: Springer International Publishing; 2018:250-260.
4. McCalpin JD. Memory bandwidth and machine balance in current high performance computers. *IEEE Comput Soc Tech Comm Comput Archit Newsl*. 1995:19-25.
5. Deakin T, Price J, McIntosh-Smith S. Portable methods for measuring cache hierarchy performance. Paper presented at: 2017 Supercomputing Conference; 2017; Denver, CO.
6. Deakin T, Price J, Martineau M, McIntosh-Smith S. Evaluating attainable memory bandwidth of parallel programming models via BabelStream. *Int J Comput Sci Eng*. 2018;17(3):247-262.

7. Hofmann J, Hager G, Wellein G, Fey D. An analysis of core- and chip-level architectural features in four generations of Intel server processors. Paper presented at: 32nd ISC High Performance International Conference; 2017; Frankfurt, Germany.
8. Mallinson AC, Beckingsale DA, Gaudin WP, Herdman JA, Levesque JM, Jarvis SA. CloverLeaf: Preparing hydrodynamics codes for exascale. Paper presented at: 2013 Cray User Group Meeting; 2013; Napa Valley, CA.
9. McIntosh-Smith S, Boulton Michael CD, Price J. On the performance portability of structured grid codes on many-core computer architectures. Paper presented at: 29th International Supercomputing Conference; 2014; Leipzig, Germany.
10. Heroux MA, Doerfler DW, et al. *Improving Performance via Mini-applications*. Springfield, VA: US Department of Commerce; 2009.
11. McIntosh-Smith S, Martineau M, Deakin T, et al. TeaLeaf: A mini-application to enable design-space explorations for iterative sparse linear solvers. Paper presented at: 2017 IEEE International Conference on Cluster Computing (CLUSTER); 2017; Honolulu, HI.
12. McIntosh-Smith S, Price J, Deakin T, Poenaru A. A performance analysis of the first generation of HPC-optimized Arm processors. *Concurrency Computat Pract Exper*. 2019;31(16):e5110.
13. Zerr RJ, Baker RS. *SNAP: SN (Discrete Ordinates) Application Proxy - Proxy Description*; 2013. https://github.com/losalamos/SNAP. Accessed 2013.
14. Deakin T, McIntosh-Smith S, Gaudin W. Many-core acceleration of a discrete ordinates transport mini-app at extreme scale. Paper presented at: 31st ISC High Performance International Conference; 2016; Frankfurt, Germany.
15. Páll S, Hess B. A flexible algorithm for calculating pair interactions on SIMD architectures. *Comput Phys Commun*. 2013;184(12):2641-2650.
16. Turner A. *Single Node Performance Comparison Report*; 2019. https://zenodo.org/record/2616549#.XbqcDpozaUk. Accessed 2019.
17. Jasak H, Jemcov A, Tukovic Z, et al. OpenFOAM: A C++ library for complex physics simulations. Paper presented at: International Workshop on Coupled Methods in Numerical Dynamics; 2007; Dubrovnik, Croatia.
18. Heft AI, Indinger T, Adams NA. Introduction of a new realistic generic car model for aerodynamic investigations. Paper presented at: SAE 2012 World Congress & Exhibition; 2012; Detroit, MI.
19. Catlow R, Woodley S, Leeuw ND, Turner A. *Optimising the Performance of the VASP 5.2.2 Code on HECToR*. HECToR; 2010. http://www.hector.ac.uk/cse/distributedcse/reports/vasp01/vasp01_collectives.pdf
20. Zhao Z, Marsman M. Estimating the performance impact of the MCDRAM on KNL using dual-socket Ivy Bridge nodes on Cray XC30. Paper presented at: 2016 Cray User Group Meeting; 2016; London, UK.

How to cite this article: McIntosh-Smith S, Price J, Poenaru A, Deakin T. Benchmarking the first generation of production quality Arm-based supercomputers. *Concurrency Computat Pract Exper*. 2019;e5569. https://doi.org/10.1002/cpe.5569