Scaling Results From the First Generation of Arm-based Supercomputers

Simon McIntosh-Smith, James Price, Andrei Poenaru, Tom Deakin
Department of Computer Science
University of Bristol
Bristol, UK
Email: S.McIntosh-Smith@bristol.ac.uk

***Abstract*—In this paper we present the first scaling results from Isambard, the first production supercomputer to be based on Arm CPUs that have been optimised specifically for HPC. Isambard is a Cray XC50 'Scout' system, combining Marvell ThunderX2 Arm-based CPUs with Cray's Aries interconnect. The full Isambard system was delivered in late 2018 and consists of a full cabinet of 168 dual-socket nodes, for a total of 10,752 heavyweight Arm cores. In this work, we build on the single-node results we presented at CUG 2018, and present scaling results for the full system. We compare Isambard's scaling results with XC systems based on the Aries interconnect and x86 CPUs, including Intel Skylake and Broadwell. We focus on a range of applications and mini-apps that are important to the UK national HPC service, ARCHER, and to Isambard project partners.**

***Keywords*-XC50; Arm; ThunderX2; benchmarking; scaling;**

I. 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. 2018 saw the delivery of the first generation of competitive HPC systems, including the 'Astra' system[1] 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 2019 we expect to see Arm-based HPC server CPUs ship from Marvell, Fujitsu, Ampere and Huawei.

In response to these developments, the 'Isambard' system[2] has been designed as the first Arm-based 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 available to any researcher funded by the UK's Engineering and Physical Sciences Research Council (EPSRC) as part of the UK national HPC ecosystem[3]. 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 250GB/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. We chose CUG 2018 as the venue to disclose the first Isambard single node results, based on early-access, "white box" nodes. In the CUG 2018 single node paper, we demonstrated that, at least at the node level, Arm-based HPC systems were performance-competitive with the best x86-based systems at the time [1]. This year we have chosen CUG 2019 to disclose system scalability results for the first time. The full Isambard system arrived in November 2018, and so this paper will present early scaling results. These results will be among the first 'at scale' performance results published for any Arm-based supercomputer, and the first results showing how well ThunderX2 performs in co-operation with Cray's Aries interconnect.

A selection from the top ten most heavily used codes that are run on the UK's national supercomputer, 'Archer' (a Cray XC30 system), along with a set of mini-apps, have been chosen to provide representative coverage of the types of codes used by citizens of today's HPC community [2]. 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.

II. 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 2.5GHz. The processors each have eight 2666 MHz DDR4 channels, yielding a measured STREAM triad bandwidth of 246 GB/s per node. 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,

[1]https://share-ng.sandia.gov/news/resources/news_releases/top_500/
[2]http://gw4.ac.uk/isambard/
[3]http://www.hpc-uk.ac.uk/facilities/

or 10,496 compute cores. Pictures of a Scout blade and an XC50 cabinet are shown in Figure 1.

The results presented in this paper are based on work performed since April 3rd 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 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 Compiler CCE, performance libraries, and analysis tools. In addition to Cray's compiler, we also used Arm's Clang/LLVM-based HPC Compiler, and the most recent versions of GCC. It should be noted that all of these compilers and libraries are still relatively early in their support for HPC-optimised Arm CPUs, and we continue to observe significant performance improvements with each new release of these tools. On the x86 platforms we also used the latest Intel compiler, and Intel MKL for BLAS and FFT routines. Details of which versions of these tools were used are given in Table II.

III. Benchmarks

A. Mini-apps

In this section we give a brief introduction to the mini-apps used in this scaling study. 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 modelling and algorithm characterisation, and can demonstrate the potential performance of the latest computer architectures.

STREAM: McCalpin's STREAM has long been the gold-standard benchmark for measuring the achievable sustained memory bandwidth of CPU architectures [3]. The benchmark is formed of simple element-wise arithmetic operations on long arrays (vectors), and for this study we consider the Triad kernel of $a(i) = b(i) + \alpha c(i)$. The achieved memory bandwidth is easily modelled as three times the length of the arrays divided by the fastest runtime for this kernel. Arrays of 2^{25} double-precision elements were used in this study, with the kernels run 200 times. While STREAM only enables node-level performance comparisons, we include it in this study to characterise the node-level memory bandwidth of the various systems in our test. This is especially important for Isambard since the node-level performance has increased since the recent upgrades to both hardware and software.

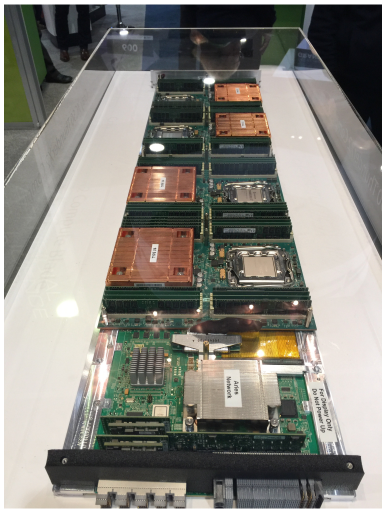
CloverLeaf: this hydrodynamics mini-app solves Euler's equations of compressible fluid dynamics, under a Lagrangian-Eulerian scheme, on a two-dimensional spatial regular structured grid [4]. These equations model the conservation of mass, energy and momentum. The mini-app is an example of a stencil code and is known to be memory bandwidth–bound. CloverLeaf is regularly used to study performance portability on many different architectures [5]. 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 [6].

TeaLeaf: this heat diffusion mini-app solves the linear heat conduction equation on a spatially decomposed regular grid, utilising a five point finite difference stencil [7]. A range of linear solvers are included in the mini-app, but the baseline method we use in this paper is the matrix-free conjugate gradient (CG) 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 in [7]). This utilises a 4000×4000 spatial grid, running for ten timesteps. Like CloverLeaf, TeaLeaf is also a member of the Mantevo mini-app suite [6].

SNAP: this is a proxy application for a modern deterministic discrete ordinates transport code [8]. 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 utilising 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 analysis of this mini-app on ThunderX2 processors [1], but is representative of running at scale where spatial concurrency becomes limited.

B. 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 Engineering and Physical Sciences Research Council (EPSRC) [9]. As such, it is important to ensure that the most heavily used codes from these national systems are tested and evaluated. To that end, a number of real applications have been selected for this study taken from the top ten most used codes


(a) An XC50 Scout blade

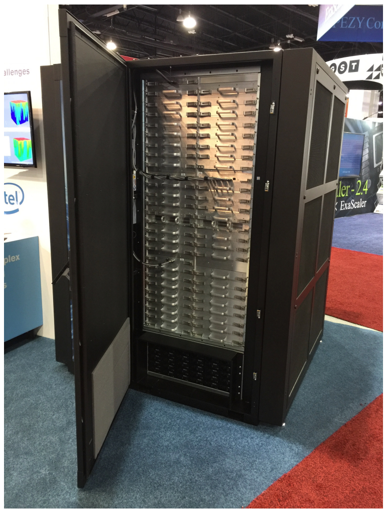

(b) An XC50 cabinet

Figure 1. Isambard hardware. Pictures © Simon McIntosh-Smith, taken at SC'17.

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 full Isambard system, we had to choose test cases that were of scientific merit, yet that could be scaled from a single node up to the full Isambard system of 164 nodes (10,496 cores).

GROMACS[4]: this widely-used molecular dynamics package 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 vectorised code using compiler intrinsics in order to ensure an optimal sequence of these operations [10]. This approach unfortunately results in GROMACS not being supported by some compilers—such as the Cray Compiler—because they do not implement all of the required intrinsics. For each supported platform, computation is packed so that it saturates the native vector length of the platform, e.g. 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 [11], running for 800 timesteps. 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. For this study we run a one MPI rank per core, using OpenMP for SMT.

NEMO: the Nucleus for European Modelling of the Ocean[5] (NEMO) code is one ocean modelling framework used by the UK's Met Office, and is often used in conjunction with the Unified Model atmosphere simulation code. The code consists of simulations of the ocean, sea-ice and marine biogeochemistry under an automatic mesh refinement scheme. As a structured grid code, the performance-limiting factor is typically memory bandwidth at the node level, however communication overheads start to significantly impact performance at scale. The benchmark we used was derived from the `GYRE_PISCES` reference configuration, with a $1/12°$ resolution and 31 model levels, resulting in 2.72M points, running for 720 time-steps. We used a pre-release of NEMO version 4.0, and we ran with one MPI rank per core for all platforms, without using SMT.

OpenFOAM: originally developed as an alternative to early simulation engines written in Fortran, OpenFOAM is a modular C++ framework aiming to simplify writing custom computational fluid dynamics (CFD) solvers [12]. In this paper, we use the `simpleFoam` solver for incompressible, turbulent flow from version 1712 of OpenFOAM[6], the most recent release at the time we began benchmarking the Isambard system. The input case is based on the RANS DrivAer generic car model, which is a representative case

[4]http://www.gromacs.org

[5]https://www.nemo-ocean.eu

[6]https://www.openfoam.com/download/install-source.php

of real aerodynamics simulation and thus should provide meaningful insight of the benchmarked platforms' performance [13]. The decomposed grid consists of approximately 64 million cells. OpenFOAM is memory bandwidth–bound, at least at low node counts.

OpenSBLI: this is a grid-based finite difference solver[7] 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 (OPS) 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[8], which solves a Taylor-Green vortex on a grid of $1024 \times 1024 \times 1024$ cells (around a billion cells). We ran with one MPI rank per core, without using SMT.

VASP: the Vienna Ab initio Simulation Package[9] (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 which affect performance, from domain decomposition options to maths 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 [14]. The benchmark utilised 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 [15]. We ran with one MPI rank per core, without using SMT.

IV. Results

A. Platforms

The full 'Phase 2' part of the Isambard system was used to produce the Arm results presented in this paper. Each of these Cray XC50 Arm nodes houses two 32-core Marvell ThunderX2 processors running at 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 appeared to run at the 2.5GHz turbo speed all of the time, no matter what code or benchmark we ran, including HPL. Each node includes 256 GB of DDR4 DRAM clocked at 2400 MHz, slightly below the 2666 MHz maximum memory speed that ThunderX2 can support. On May 7th 2018, Marvell announced the general availability of ThunderX2, with an RRP for the 32c 2.2GHz part of $1,795 each. The ThunderX2 processors support 128-bit vector Arm Advanced SIMD instructions (sometimes referred to as 'NEON'), and each core is capable of 4-way simultaneous multithreading (SMT), for a total of up to 256 hardware threads per node. The processor's on-chip data cache is organised 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 utilises 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 $8,719 each.
- Intel Xeon Gold 6148 (Skylake) 20-core @ 2.4 GHz, dual-socket, with 192 GB of DDR4-2666 DRAM. RRP $3,078 each.
- Intel Xeon E5-2699 v4 (Broadwell) 22-core @ 2.2 GHz, dual-socket, with 128 GB of DDR4-2400 DRAM. RRP $4,115 each.

The recommended retail prices (RRP) were correct at the time of writing for the single-node performance paper we published at CUG 2018 (May 2018), and taken from Intel's website at that time[10]. The Skylake processors provide an AVX-512 vector instruction set, meaning that each FP64 vector operation processes eight elements at once; by comparison, Broadwell utilises 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 in order 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 I, 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. There are several important

[7]https://opensbli.github.io
[8]http://www.archer.ac.uk/community/benchmarks/archer/
[9]http://www.vasp.at
[10]https://ark.intel.com/

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	1.13	320

Table I
HARDWARE INFORMATION (PEAK FIGURES)

Benchmark	ThunderX2	Broadwell	Skylake
STREAM	GCC 8.2	Intel 2019	CCE 8.7
CloverLeaf	CCE 8.7	Intel 2019	Intel 2019
TeaLeaf	CCE 8.7	Intel 2019	Intel 2019
SNAP	CCE 8.7	Intel 2019	Intel 2019
GROMACS	GCC 8.2	GCC 8.2	GCC 8.2
NEMO	CCE 8.7	CCE 8.7	CCE 8.7
OpenFOAM	GCC 7.3	GCC 7.3	GCC 7.3
OpenSBLI	CCE 8.7	Intel 2019	CCE 8.7
VASP	GCC 7.3	Intel 2019	Intel 2019

Table II
COMPILERS USED FOR BENCHMARKING

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 datapaths 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 (DVFS) makes it harder to reason about the percentage of peak performance that is being 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 2.1GHz. 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 51%, respectively.

In order to measure the sustained cache bandwidths as presented in Figure 2, we used the methodology described in our previous work [16]. The Triad kernel from the STREAM 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 modelled 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 hand-written benchmarks which only work on their target architectures [17].

We are currently in the process of evaluating the three major compiler families available for ThunderX2: 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 date that has compared all three of these compilers targeting Arm [1]. The compiler that achieved the highest performance in each case is used in the results graphs displayed below. Likewise, for the Intel processors we used GCC, Intel, and Cray CCE. Table II details which compiler was used for each benchmark on each platform. This data is still changing at the time of writing, and we are even finding cases where, for a given code, one compiler might be fastest up to a certain number of nodes, then another compiler is faster at higher node counts.

Most of the scaling results we show in the rest of this paper scale up to 64 nodes. This limit was imposed by the x86 node counts in the Swan and Horizon 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, OpenFOAM and OpenSBLI we start from either two or four nodes due to memory and runtime constraints.

B. *Mini-apps*

CloverLeaf: The normalised 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 [5]. 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. There is a slight drop-off in parallel efficiency for CloverLeaf relative to the other platforms, which is currently under investigation.

TeaLeaf: Figure 4a compares the performance of the TeaLeaf mini-app between the four systems up to 64 nodes.

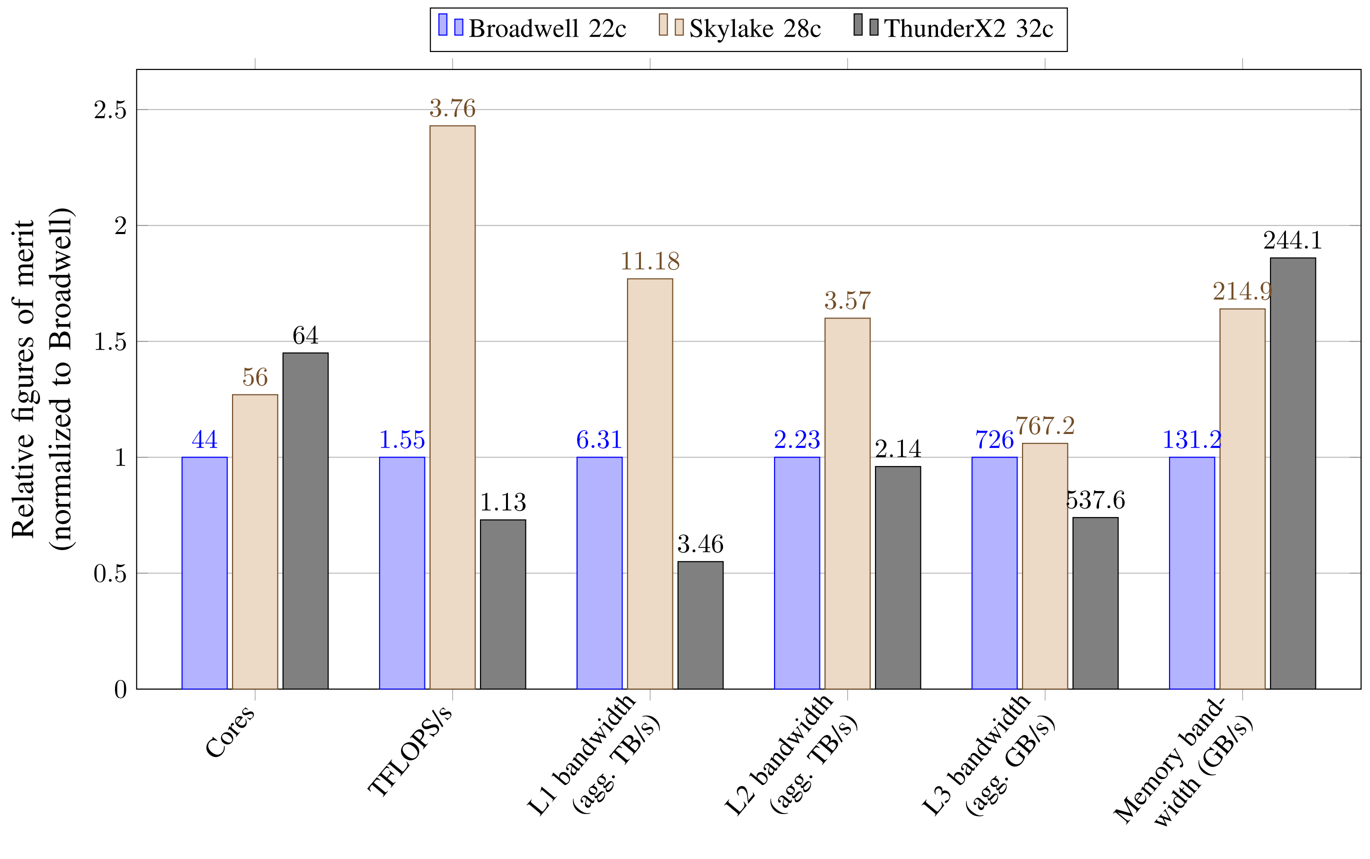


Figure 2. Comparison of properties of Broadwell 22c, Skylake 28c and ThunderX2 32c. Results are normalized to Broadwell.

The relative performance results on a single node are similar to those presented in [18], 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 ThunderX2 is 80% faster than Broadwell and up to 10% faster than Skylake. At scale, however, Isambard is unable to sustain this performance advantage for the problem that we are using.

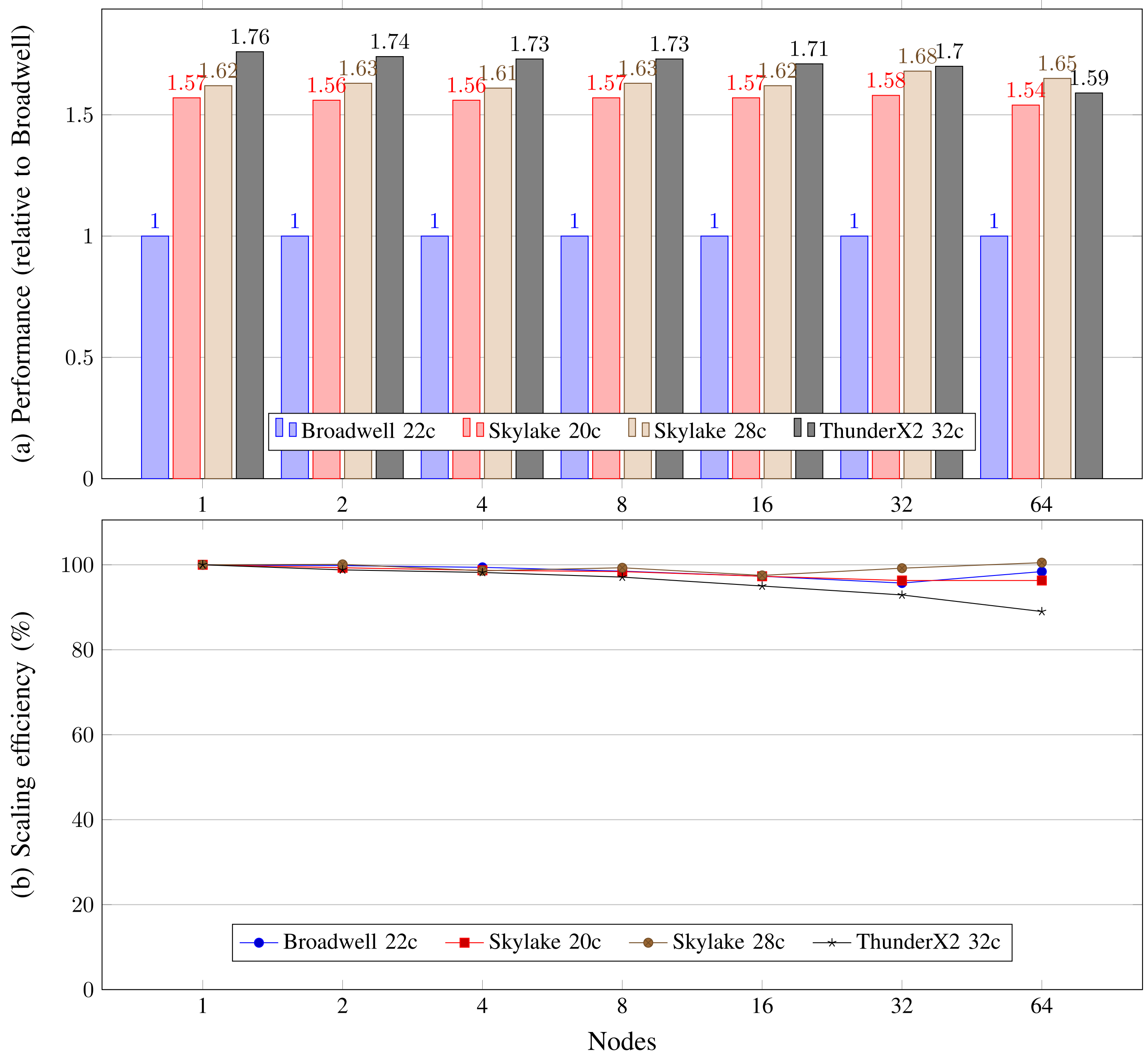
As shown in Figure 4b, all platforms achieve super-linear scaling behaviour up to around 16 nodes which, as observed in [7], is due to cache effects of strong-scaling over a relatively small data-set. The super-linear improvement is much less pronounced on ThunderX2 than with the x86 systems, which is 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 ends up at around 2× slower than the x86 systems when using 64 nodes. This issue is currently under investigation.

SNAP: Running the weak scaling setup described in Section III-A, the runtimes at all scales are similar across all the architectures tested, as seen 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 [19]. Therefore, a similar level of performance can be seen at modest scale irrespective of the architecture. Figure 5b also shows that each system has a very similar parallel efficiency up to 8 nodes and follows a similar trend to our previous work [19]. While the x86 systems we were using in these tests only had 64 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%.

C. Applications

GROMACS: 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 1.66× faster than Broadwell, while ThunderX2 is 1.23× slower. As the node count

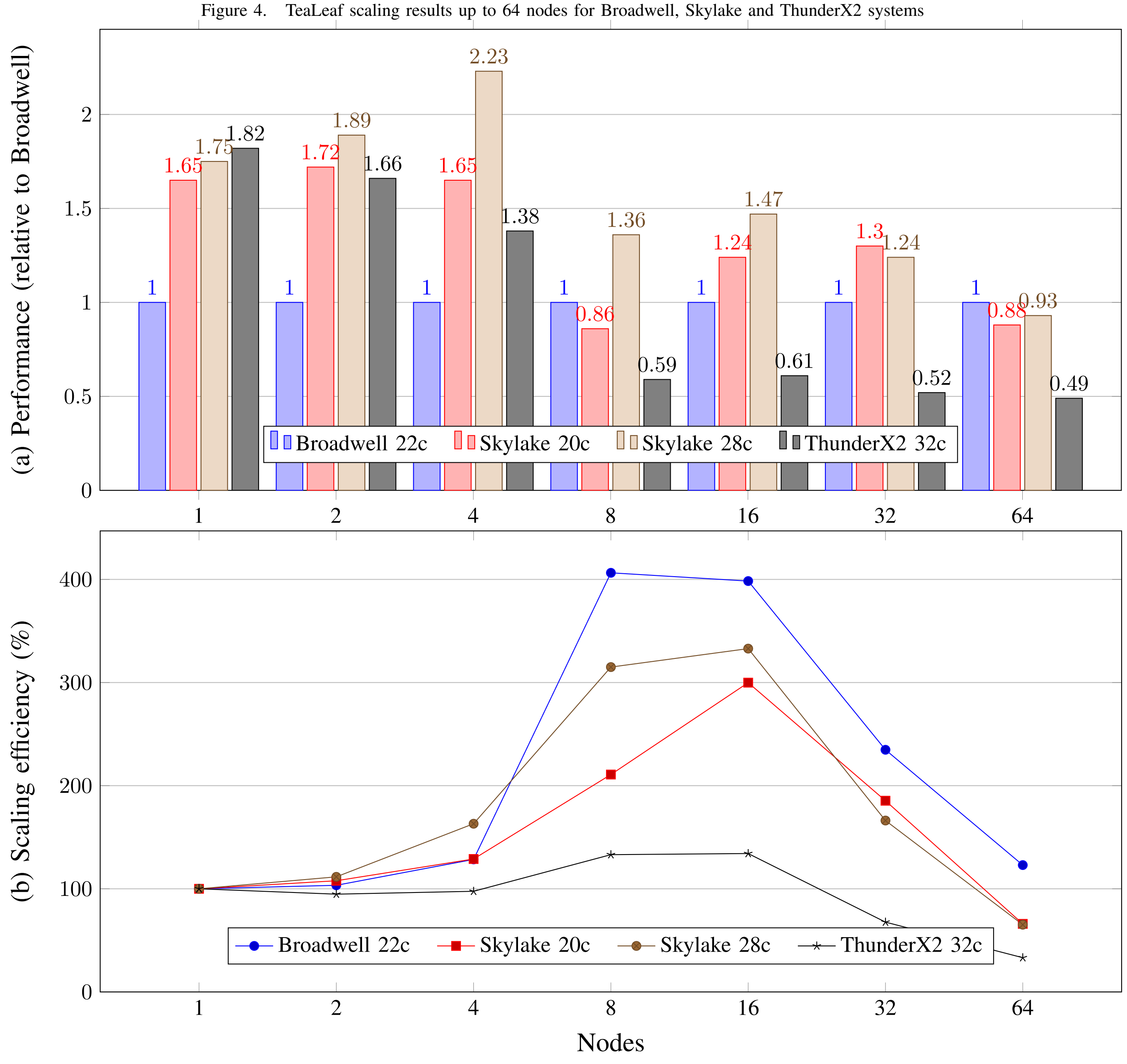
Figure 3. CloverLeaf scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



increases, the performance becomes increasingly affected by communication costs. Figure 6b shows that the scaling efficiency drops to below 40% for Skylake Platinum at 64 nodes, with MPI communications accounting for 72% of the total runtime. Since the node-level performance is lower, ThunderX2 is able to achieve a scaling efficiency of 70% for 64 nodes. As a result of this, Isambard achieves performance almost on par with both Skylake SKUs at 64 nodes, making up for the lower floating-point throughput and cache bandwidth.

NEMO: Figure 7b shows the scaling efficiency of the NEMO benchmark up to 64 nodes. This benchmark produces super-linear scaling behaviour up to eight nodes on the x86 systems since the working data starts fitting into the caches. As also observed with the TeaLeaf results, the ThunderX2 processors benefit from caching effects much less than the x86 processors, and while some individual components do experience super-linear scaling behaviour, the overall efficiency does not. On a single node, ThunderX2 is $1.45\times$ faster than Broadwell and around $1.14\times$ faster than Skylake Gold, while just a $1.03\times$ slower than the Skylake Platinum system (see Figure 7a). Due to the scaling behaviour described above, the performance at scale is less competitive, dropping to around half the performance of the

Figure 4. TeaLeaf scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems

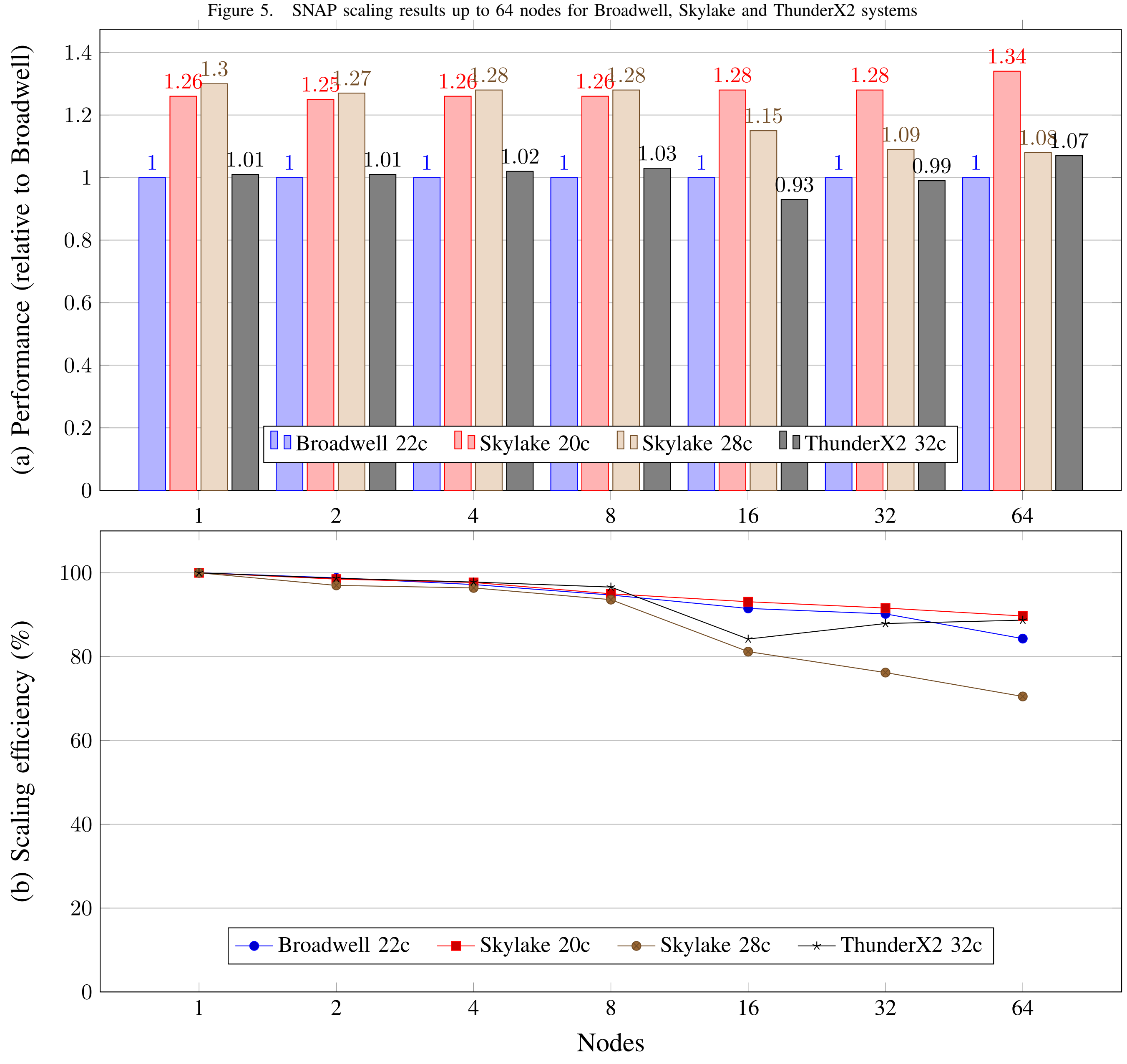


Broadwell and Skylake systems (which all achieve similar performance from eight nodes onwards).

OpenFOAM: The OpenFOAM results shown in Figure 8a start off following the STREAM behaviour 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 1.58× and 1.65× on Skylake 20c and 28c, respectively. At higher node counts, other factors come into play, where in Figure 8b we see Broadwell scaling the best of all the platforms, Skylake also maintaining good scaling, and ThunderX2 scaling the least well, with parallel efficiency dropping to below 80%. From early investigations, we suspect this is the same issue as with TeaLeaf, and related to how Cray's MPI collective operations are implemented on ThunderX2. At the time of writing we are investigating this with Cray.

OpenSBLI: The scaling efficiency for OpenSBLI, shown in Figure 9b, is similar across the four systems tested. Each system sustains efficiency above 85% up to 64 nodes, with some slight super-linear scaling behaviour observed due to caching effects. At low node counts, performance of the OpenSBLI benchmark is dominated by bandwidth to DRAM

Figure 5. SNAP scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems

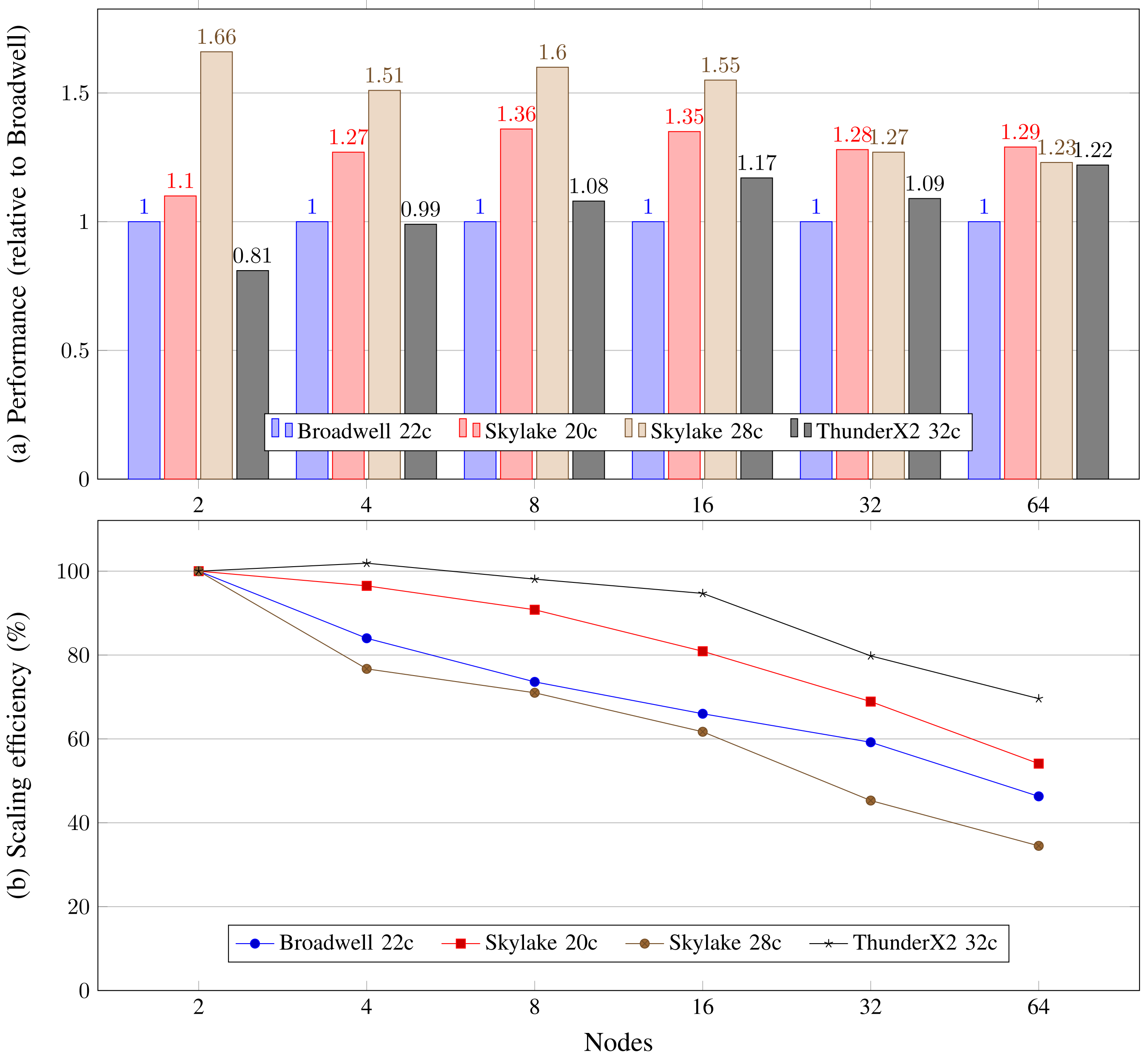


and L3 cache. The Skylake Platinum system is the fastest at four nodes, beating both ThunderX2 and Skylake Gold by around 25% (see Figure 9a). This lead diminishes at higher node counts as communication costs begin to take their toll. At 64 nodes, Isambard achieves 30% higher performance than the Broadwell system, and is competitive with the two SKUs of Skylake.

VASP: The scaling efficiency for VASP, shown in Figure 10b, is similar across the four systems tested. At 16 nodes, the ThunderX2 and Skylake systems are all below 50% 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 favour the higher floating-point throughput and cache bandwidth of the x86 processors with their wider vector units. The net result (shown in Figure 10a) is that, at 16 nodes, Isambard is a $1.2\times$ slower than the Broadwell system, and $1.32-1.52\times$ slower than the Skylake systems.

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

Figure 6. GROMACS scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



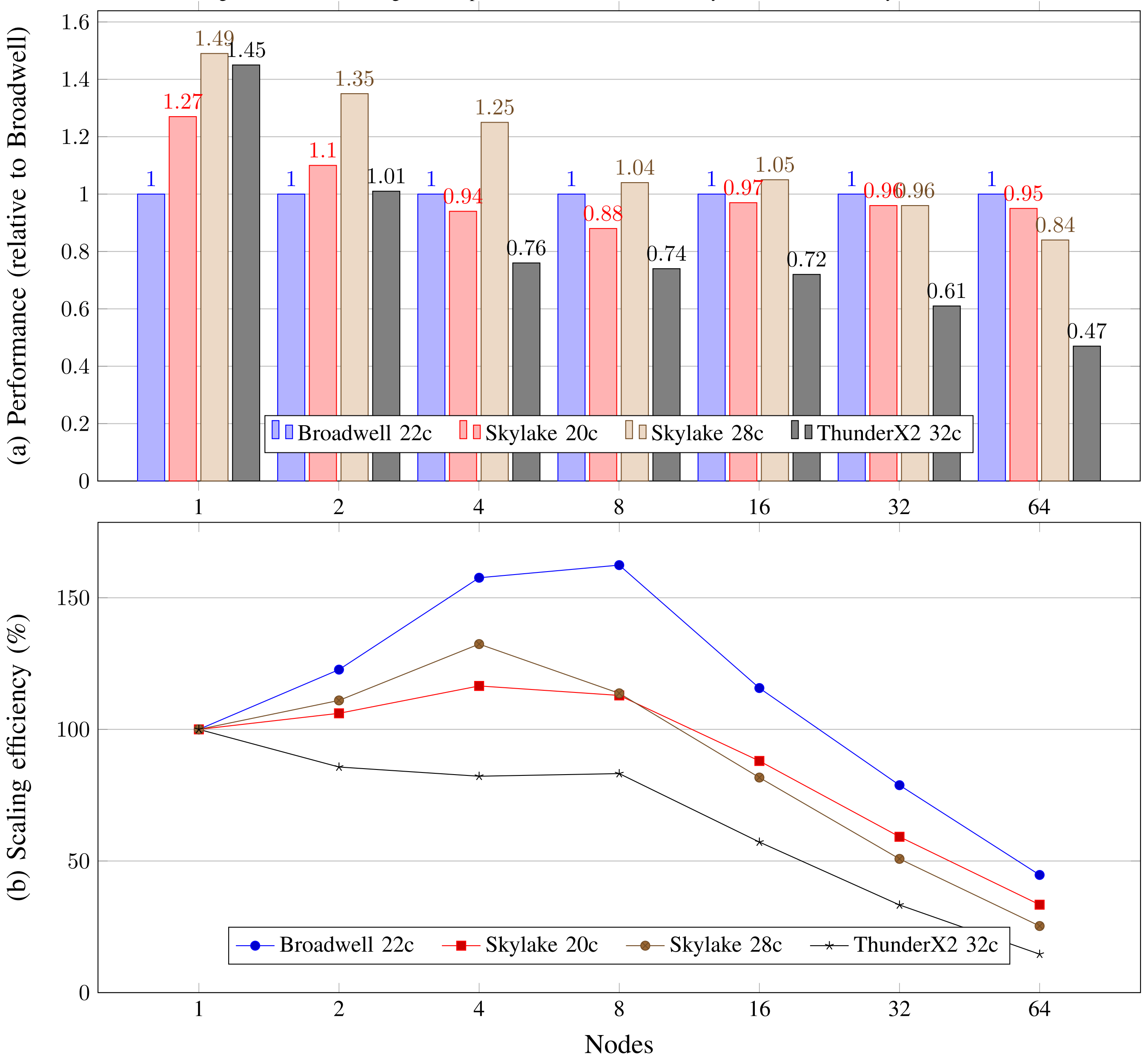
significant performance improvements when an application's performance is limited by external memory bandwidth, but are slower in cases where codes are compute-bound. At higher node counts, the differences between node-level peak bandwidth or FLOP/s becomes less significant, with often the network becoming the limiting factor. Given that, by design, all 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. 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.

For the codes where we observed that the Arm-based system does not scale as well as the x86-based ones, such as TeaLeaf and OpenFOAM, our investigations indicate that specific issues in the implementation of Cray's current MPI collective operations are the likely cause. These issues are currently being pursued.

V. Reproducibility

With an architecture such as Arm which is new to mainstream HPC, it is important to make any benchmark

Figure 7. NEMO scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



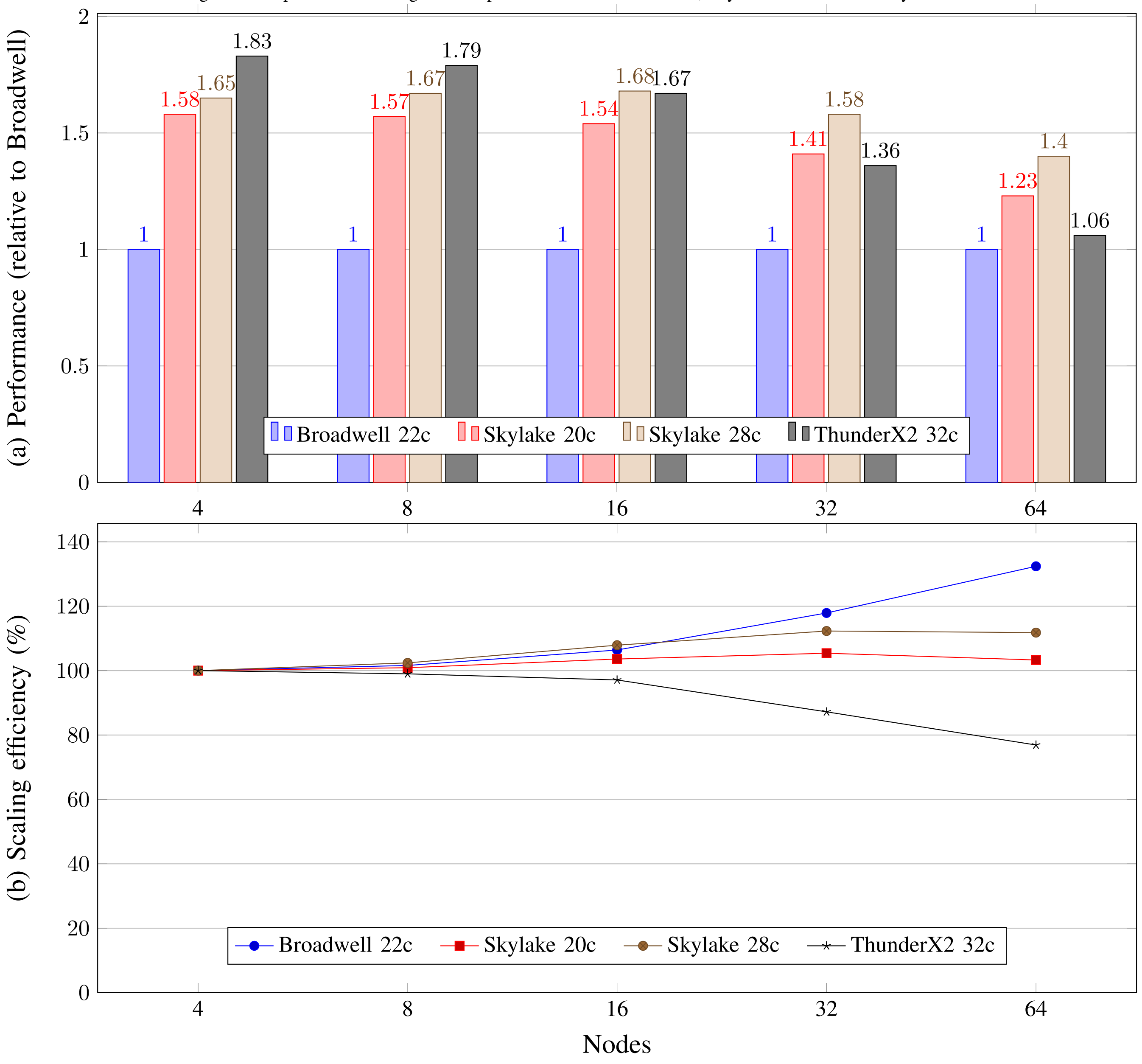
comparisons as easy to reproduce as possible. To this end, the Isambard project is making all of the detailed information about how each code was compiled and run, along with the input parameters to the test cases, available as an open source repository on GitHub[11]. The build scripts will show which compilers were used in each 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 parameterised. These two sets of scripts should enable any third party to reproduce our results, provided that they have access to similar hardware. The scripts do assume a Cray-style system, but should be easily portable to other versions of Linux on non-Cray systems.

[11]https://github.com/UoB-HPC/benchmarks/releases/tag/CUG-2019

VI. Conclusions

The results presented in this paper demonstrate that Arm-based processors are now capable of providing levels of performance competitive with state-of-the-art offerings from the incumbent vendors, while significantly improving Performance per Dollar. 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

Figure 8. OpenFOAM scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



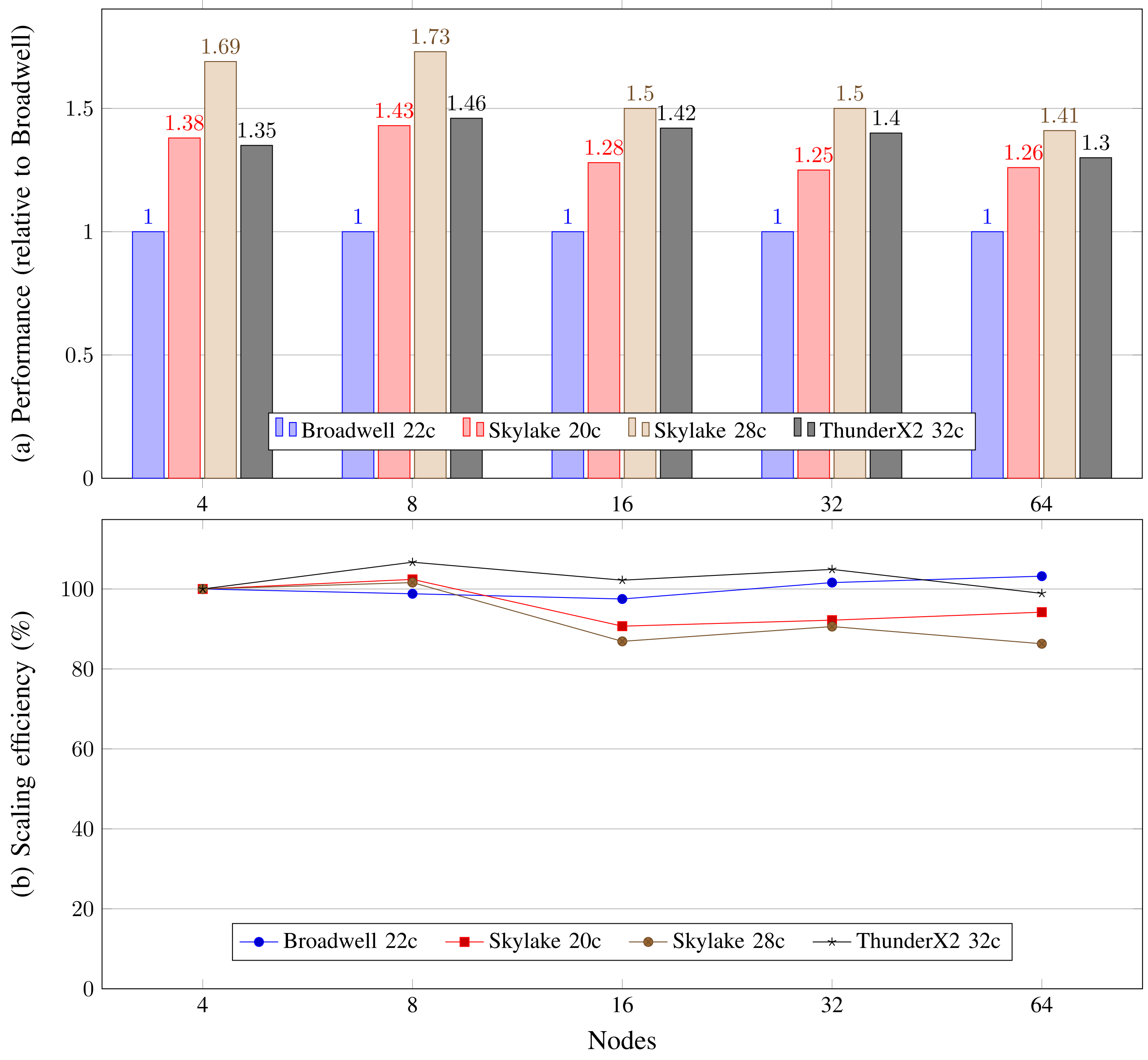
appropriate for real science runs, ThunderX2 often become even more competitive, due to its greater memory bandwidth benefiting communication performance. We also saw that most codes scaled similarly between x86 and ThunderX2, the first time this has been demonstrated between systems with the same interconnect and near identical software stacks. 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 procurements.

Overall, these results suggest that Arm-based server CPUs that have been optimised 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.

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 proposal, including: Prof James Davenport

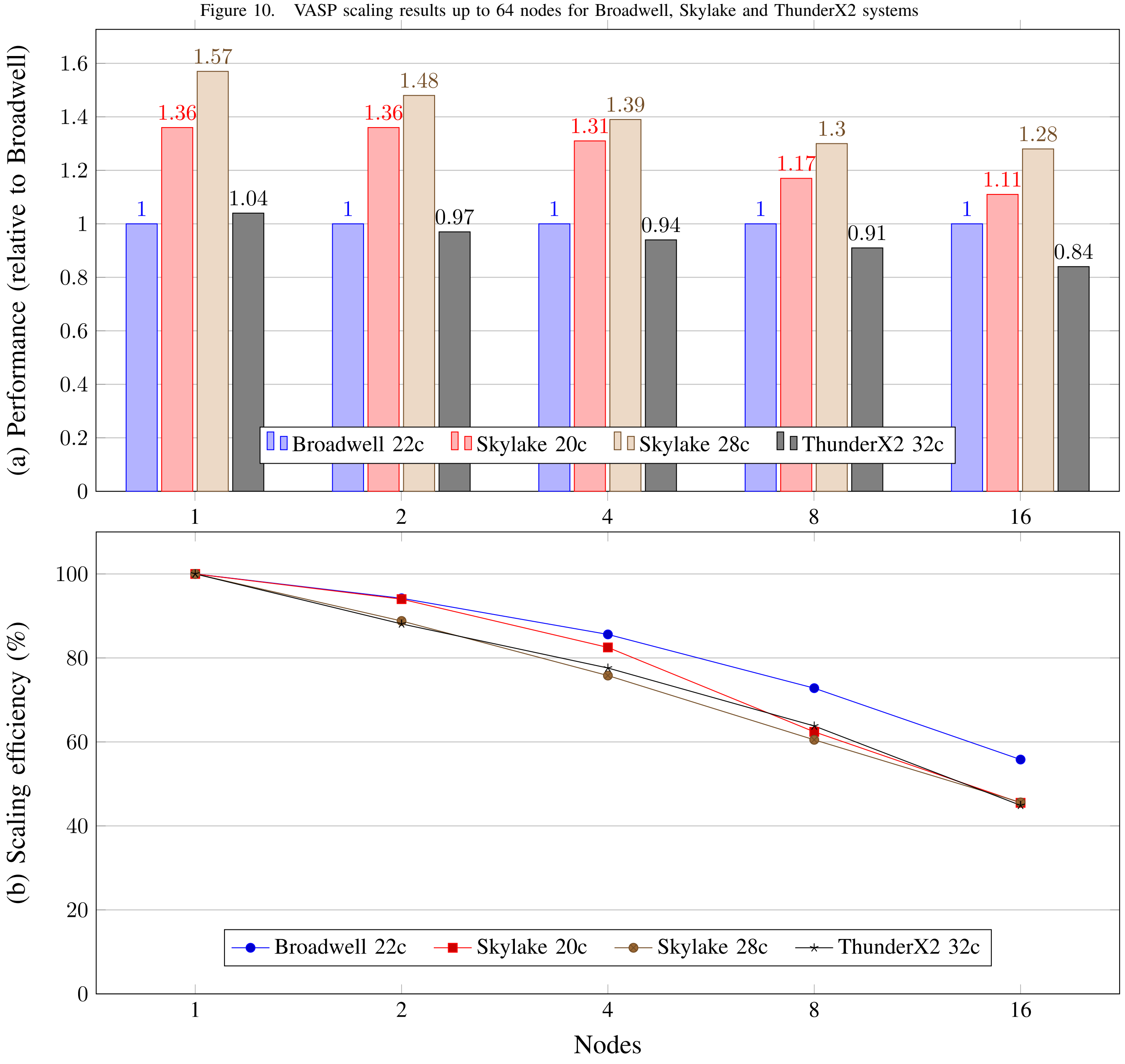
Figure 9. OpenSBLI scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



(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), 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); 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 organised these events. For the lists of attendees of the hackathons in November 2017 and March 2018, please see the Isambard CUG 2018 paper [1].

Access to the Cray XC40 supercomputers ‘Swan’ and ‘Horizon’ was kindly provided though Cray Inc.’s Marketing Partner Network. The Isambard project is funded by EPSRC, the GW4 alliance, the Met Office, Cray and Arm. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Figure 10. VASP scaling results up to 64 nodes for Broadwell, Skylake and ThunderX2 systems



The Isambard project is funded by EPSRC research grant number EP/P020224/1. Further Isambard-related research was funded by the ASiMoV EPSRC prosperity partnership project, grant number EP/S005072/1.

References

[1] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, "Comparative benchmarking of the first generation of HPC-optimised Arm processors on Isambard." in *Cray User Group meeting (CUG)*, 2018.

[2] A. Turner and S. McIntosh-Smith, "A survey of application memory usage on a national supercomputer: An analysis of memory requirements on ARCHER," in *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, S. Jarvis, S. Wright, and S. Hammond, Eds. Cham: Springer International Publishing, 2018, pp. 250–260.

[3] J. D. McCalpin, "Memory bandwidth and machine balance in current high performance computers," *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, Dec 1995.

[4] A. Mallinson, D. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. Jarvis, "CloverLeaf: Preparing hydrodynamics codes for exascale," in *The Cray User Group*, May 2013.

[5] S. McIntosh-Smith, M. Boulton, D. Curran, and J. Price, “On the performance portability of structured grid codes on many-core computer architectures,” *Supercomputing*, vol. 8488, pp. 53–75, 2014.

[6] M. Heroux, D. Doerfler *et al.*, “Improving Performance via Mini-applications,” Sandia National Laboratories, Tech. Rep. SAND2009-5574, 2009.

[7] S. McIntosh-Smith, M. Martineau, T. Deakin, G. Pawelczak, W. Gaudin, P. Garrett, W. Liu, R. Smedley-Stevenson, and D. Beckingsale, “TeaLeaf: A mini-application to enable design-space explorations for iterative sparse linear solvers,” in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, sep 2017, pp. 842–849. [Online]. Available: http://ieeexplore.ieee.org/document/8049027/

[8] R. J. Zerr and R. S. Baker, “SNAP: SN (discrete ordinates) application proxy - proxy description,” LA-UR-13-21070, Los Alamos National Labratory, Tech. Rep., 2013.

[9] A. Turner and S. McIntosh-Smith, “A survey of application memory usage on a national supercomputer: An analysis of memory requirements on ARCHER,” in *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, S. Jarvis, S. Wright, and S. Hammond, Eds. Cham: Springer International Publishing, 2018, pp. 250–260.

[10] S. Páll and B. Hess, “A flexible algorithm for calculating pair interactions on SIMD architectures,” *Computer Physics Communications*, vol. 184, no. 12, pp. 2641 – 2650, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465513001975

[11] A. Turner, “Single node performance comparison report,” Mar 2019.

[12] H. Jasak, A. Jemcov, Z. Tukovic *et al.*, “OpenFOAM: A C++ library for complex physics simulations,” in *International workshop on coupled methods in numerical dynamics*, IUC Dubrovnik, Croatia, September 2007, pp. 1–20. [Online]. Available: http://powerlab.fsb.hr/ped/kturbo/openfoam/papers/CMND2007.pdf

[13] A. I. Heft, T. Indinger, and N. A. Adams, “Introduction of a new realistic generic car model for aerodynamic investigations,” SAE Technical Paper, Tech. Rep., 2012. [Online]. Available: https://doi.org/10.4271/2012-01-0168

[14] R. Catlow, S. Woodley, N. D. Leeuw, and A. Turner, “Optimising the performance of the VASP 5.2.2 code on HECToR,” HECToR, Tech. Rep., 2010. [Online]. Available: http://www.hector.ac.uk/cse/distributedcse/reports/vasp01/vasp01_collectives/

[15] Z. Zhao and M. Marsman, “Estimating the performance impact of the MCDRAM on KNL using dual-socket Ivy Bridge nodes on Cray XC30,” in *Cray User Group Meeting (CUG 2016)*, 2016.

[16] T. Deakin, J. Price, and S. McIntosh-Smith, “Portable methods for measuring cache hierarchy performance (poster),” in *Supercomputing*, Denver, Colorado, 2017.

[17] J. Hofmann, G. Hager, G. Wellein, and D. Fey, “An analysis of core- and chip-level architectural features in four generations of Intel server processors,” ser. Lecture Notes in Computer Science, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, Eds. Cham: Springer International Publishing, 2017, vol. 10266, pp. 294–314. [Online]. Available: http://link.springer.com/10.1007/978-3-319-58667-0http://link.springer.com/10.1007/978-3-319-58667-0{_}16

[18] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, “A performance analysis of the first generation of HPC-optimized Arm processors,” *Concurrency and Computation: Practice and Experience*, vol. 0, no. 0, p. e5110, e5110 cpe.5110. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5110

[19] T. Deakin, S. McIntosh-Smith, and W. Gaudin, “Many-Core Acceleration of a Discrete Ordinates Transport Mini-App at Extreme Scale,” in *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings*, M. J. Kunkel, P. Balaji, and J. Dongarra, Eds. Cham: Springer International Publishing, 2016, pp. 429–448. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-41321-1_22