Comparative Benchmarking of the First Generation of HPC-Optimised Arm Processors on Isambard

Simon McIntosh-Smith, James Price, Tom Deakin and Andrei Poenaru
High Performance Computing research group
Department of Computer Science
University of Bristol, Bristol, UK
Email: cssnmis@bristol.ac.uk

***Abstract*—In this paper we present performance results from Isambard, the first production supercomputer to be based on Arm CPUs that have been optimised specifically for HPC. Isambard is the first Cray XC50 'Scout' system, combining Cavium ThunderX2 Arm-based CPUs with Cray's Aries interconnect. The full Isambard system will be delivered in the summer of 2018 when it will contain over 10,000 Arm cores. In this work we present node-level performance results from eight early-access nodes that were upgraded to B0 beta silicon in March 2018. We present node-level benchmark results comparing ThunderX2 with mainstream CPUs, including Intel Skylake and Broadwell, as well as Xeon Phi. We focus on a range of applications and mini-apps important to the UK national HPC service, ARCHER, as well as to the Isambard project partners and the wider HPC community. We also compare performance across three major software toolchains available for Arm: Cray's CCE, Arm's version of Clang/Flang/LLVM, and GNU.**

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

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 systems companies, and competition between vendors.

The FP7 Mont-Blanc project used Arm processors designed for the mobile space to investigate the feasibility of using the Arm architecture for workloads relevant to the HPC community. Mont-Blanc's early results were encouraging, but it was clear that chips designed for the mobile space could not compete with HPC-optimised CPUs without further architecture and implementation developments. As a result, Cavium will release its first generation of HPC-optimised, Arm-based CPUs in 2018.

In response to these developments, 'Isambard' has been designed to provide the first Arm-based Cray XC50 'Scout' system as part of the UK's national HPC service[1]. Cavium ThunderX2 processors will form the basis of the system; these processors use the Armv8 instruction set but have been designed specifically for HPC 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 in excess of 250 GB/s. The Isambard system represents a collaboration between the GW4 Alliance (formed from the universities of Bristol, Bath, Cardiff and Exeter) along with the UK's Met Office, Cray, Arm and Cavium, and funded by EPSRC. Although Isambard is due to arrive in July 2018, this paper will present results using the project's early-access nodes, delivered in October 2017 and upgraded to B0 beta silicon in March 2018. These results will be among the first published for the production silicon of the Arm-based Cavium ThunderX2 processor, and the first using Cray's CCE tools for Arm.

The results we shall present will focus on single node, dual-socket performance in comparison to other state-of-the-art processors found in the majority of supercomputers today. These results will also lay the foundations for a future study of at-scale, production style workloads running on Isambard, which will utilise the Cray Aries interconnect.

The top ten most heavily used codes that are run on the UK's national supercomputer, ARCHER[2] (a Cray XC30 system), along with a selection of mini-apps, proxy applications, and applications from project partners, provide good representation of the styles of codes used by today's HPC community [1]. As such, they provide an ideal vehicle for which to benchmark the new Cavium ThunderX2 architecture, and the Isambard system presents a unique opportunity for such comparative benchmarking. Its heterogeneous design enables direct comparison between Cavium ThunderX2 CPUs and the best of today's mainstream HPC hardware, including x86 Intel Xeon and Xeon Phi processors, and NVIDIA P100 GPUs.

II. Isambard: system overview

The most exciting aspect of the Isambard system will be the full cabinet of XC50 'Scout' with Cavium ThunderX2, delivering over 10,000 high-performance Armv8 cores. Each

This paper has been accepted and will be published as an article in a special issue of Concurrency and Computation Practice and Experience on the Cray User Group 2018.

[1]http://gw4.ac.uk/isambard/

[2]https://www.archer.ac.uk

node includes two 32-core ThunderX2 processors running at 2.1 GHz. The processors each have eight 2666 MHz DDR4 channels, yielding a STREAM triad result of over 250 GB/s per node. The XC50 Scout system packs 4 dual-socket nodes into each blade, and then up to 46 such blades in a single cabinet. 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 at two hackathons, using the Isambard early-access nodes. These early-access nodes use the same Cavium ThunderX2 CPUs as the full Isambard XC50 system, but in a Foxconn whitebox form-factor. These nodes were upgraded to B0 beta silicon in March 2018 and were run in dual-socket, 32-core, 2.2 GHz configuration for the purposes of gathering the data presented herein. Each node includes 256 GB of DDR4 DRAM and runs SLES 12 SP3, with a cut-down set of the Cray Programming Environment (CPE). This subset of CPE includes all the elements we needed for benchmarking the ThunderX2 CPUs: the Cray Compiler, CCE, performance libraries and analysis tools. In addition we used Arm's LLVM-based compiler, and GNU. It should be noted that all of these compilers are still relatively early in their support for HPC-optimised Arm CPUs, given that, at the time of this work, production hardware has yet to ship. 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 study. The mini-apps themselves are all performance proxies for larger production codes, encapsulating the important performance characteristics such as floating point intensity, memory access and communication patterns of their parent application, 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 [2]. 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.

CloverLeaf: The CloverLeaf hydrodynamics mini-app solves Euler's equations of compressible fluid dynamics, under a Lagrangian-Eulerian scheme, on a two-dimensional spatial regular structured grid [3]. These equations model the conservation of mass, energy and momentum. The mini-app is an example of a stencil code and is classed as memory bandwidth–bound. CloverLeaf is regularly used to study performance portability on many different architectures [4]. The `bm_16` test case consists of a grid of 3840×3840 cells.

TeaLeaf: The TeaLeaf heat diffusion mini-app solves the linear heat conduction equation on a spatially decomposed regular grid, utilising a five point finite difference stencil [5]. 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. For this study, the focus is on single-node performance, and so we used the `bm_5` input deck, which utilises a 4000×4000 spatial grid.

SNAP: SNAP is a proxy application for a modern deterministic discrete ordinates transport code [6]. 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. The kernel itself is limited by the performance of the memory hierarchy [7]. Benchmarking SNAP is somewhat challenging, as the spatial problem size must divide evenly by the number of MPI ranks (often core count). Therefore, a fixed problem size per core was used for this study with a spatial grid of $1024 \times 2 \times 2$ per MPI rank, 136 angles per octant and 32 groups, with one MPI rank per physical core. The footprint of the angular flux solution in this case is just over 1 GB, which is roughly in line with typical ARCHER usage [1]; note that this requires use of the cache mode on the Knights Landing processor due to the limited 16 GB capacity of MCDRAM on that architecture. The figure of merit for this application is the *grind time*, which is a normalised value taking into account differing problem sizes.

Neutral: The Monte Carlo neutral particle transport mini-app, Netural, was written to explore the challenges of such algorithms on advanced architectures, disproving the old adage that Monte Carlo codes are 'embarrassingly parallel' [8]. The mini-app takes a mesh-based approach, and so there is a tension with regard to memory access of mesh and particle data, resulting in memory latency becoming the performance-limiting factor. The `csp` input included with the mini-app was used for this study, and the benchmark was run using OpenMP.

B. Applications

The Isambard system has been designed to explore the feasibility of an Arm-based system for real HPC workloads. As such, it is important to ensure that the most heavily used codes are tested and evaluated. To that end, eight real applications have been selected for this study taken from the top ten most used code on ARCHER, the UK National Supercomputer [1]. These applications represent over 50%

(a) An XC50 Scout blade

(b) An XC50 cabinet

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

of the usage of the whole supercomputer. 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.

CP2K: The CP2K code[3] simulates the Ab-initio electronic structure and molecular dynamics of different systems such as molecular, solids, liquids, and so on. Fast Fourier Transforms (FFTs) form part of the solution step, but it is not straightforward to attribute these as the performance-limiting factor of this code. The memory bandwidth of the processor and the core count both have an impact. The code already shows sub-linear scaling up to tens of cores. Additionally, the performance of the code on a single node does not necessarily have the same performance-limiting factors as when running the code at scale. We will need the full Isambard XC50 system to test these effects, but in the meantime, we have used the `H2O-64` benchmark, which simulates 64 water molecules (consisting of 192 atoms and 512 electrons) in a 12.4 $\mathring{A}^3$ cell for 10 time-steps. This is an often-studied benchmark for CP2K, and therefore provides sufficient information to explore the performance across the different architectures in this paper.

GROMACS: The molecular dynamics package GROMACS[4] is used to solve Newton's equations of motion. Systems of interest such as proteins, contain up to millions of particles. It is thought that GROMACS is bound by the floating-point performance of the processor architecture. This has motivated the developers to handwrite vectorised code in order to ensure an optimal sequence of such arithmetic [9]. The hand-optimised code is written using compiler intrinsics, which results in GROMACS not supporting some compilers — such as the Cray Compiler — because they do not implement all 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 the `ion_channel_vsites` benchmark[5]. This consists of the membrane protein GluCl, containing around 150,000 atoms (which for GROMACS is typically small), and uses 'vsites' and a five femtosecond time-step. On the ThunderX2 processor, we used the `ARM_NEON_ASIMD` vector implementation, which is the closest match for the Armv8.1 architecture. However, this implementation is not as mature as some of those targeting x86.

NAMD: NAMD is a molecular dynamics simulation program designed to scale up to millions of atoms [10]. It is able to achieve scaling to more than half a million processor cores using the Charm++ parallel programming framework, a high-level library that abstracts the mapping of processors to work items — or *chares* — away from the programmer [11]. The test case used is the STMV benchmark, which simulates one of the smallest viruses in existence, and which is a common set of inputs for measuring scaling capabilities. This benchmark includes PME calculations, which use FFTs, and so its performance is heavily influenced by that of the FFT library used. Due to the complex structure of atomic simulation computation and the reliance of distributed FFTs, it is hard to define a single bounding factor for NAMD's performance — compute performance, memory bandwidth, and communication capabilities all affect the overall performance of the application.

NEMO: The Nucleus for European Modelling of the Ocean[6] (NEMO) code is one ocean modelling framework

[3]https://www.cp2k.org
[4]http://www.gromacs.org
[5]https://bitbucket.org/pszilard/isambard-bench-pack.git
[6]https://www.nemo-ocean.eu

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 highly likely to be memory bandwidth. The benchmark we used was derived from the `GYRE_PISCES` reference configuration, with a $1/12^\circ$ resolution and 31 model levels, resulting in 2.72M points, running for 720 time-steps.

OpenFOAM: Originally developed as an alternative to early simulation engines written in Fortran, OpenFOAM is a modular C++ framework aiming to simplify writing custom computation fluid dynamics (CFD) solvers. The code makes heavy use of object-oriented features in C++, such as class derivation, templating and generic programming, and operator overloading, which it uses to allow for powerful extensive design [12]. OpenFOAM's flexibility comes at the cost of additional code complexity, of which a good example is how every component is compiled into a separate dynamic library, and generic executables load the required modules at runtime, based on the user's input. The two features mentioned above grant OpenFOAM a unique position in our benchmark suite. In this paper, we use the `simpleFoam` solver for incompressible, turbulent flow from version 1712 of OpenFOAM[7], the latest released version at the time of writing. The input case is based on the RANS DrivAer generic car model, which is a representative case of real aerodynamics simulation and thus should provide meaningful insight of the benchmarked platforms' performance [13]. OpenFOAM is almost entirely memory bandwidth–bound.

OpenSBLI: OpenSLBI is a grid-based finite difference solver[8] 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[9], which solves a Taylor-Green vortex on a grid of $1024 \times 1024 \times 1024$ cells (around a billion cells).

Unified Model: The UK's Met Office code, the Unified Model[10] (UM), is an atmosphere simulation code used for weather and climate applications. It is often coupled with the NEMO code. The UM is used for weather prediction, seasonal forecasting, and for climate modelling, with timescales ranging from days to hundreds of years. At its core, the code solves the compressible non-hydrostatic motion equations on the domain of the Earth discretised into a latitude-longitude grid. As a structured grid code, the performance limiting factor is highly likely to be memory bandwidth. We used an `AMIP` benchmark [14] provided by the UK Met Office for version 10.8 of the UM.

VASP: The Vienna Ab initio Simulation Package[11] (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 [15]. The benchmark utilised is known as `PdO`, because it simulates a slab of palladium oxide. It consists of 174 atoms, and it was originally designed by one of VASP's developers, who also found that (on a single node) the benchmark is mostly compute-bound; however, there exist a few methods that benefit from increased memory bandwidth [16].

IV. Results

A. Platforms

The early-access part of the Isambard system was used to produce the Arm results presented in this paper. Each of the Arm nodes houses two 32-core Cavium ThunderX2 processors running at 2.2 GHz alongside 256 GB of DDR4 DRAM clocked at 2500 MHz. Note that this is slightly below the 2666 MHz memory speed of Isambard's XC50 nodes. On May 7th 2018, Cavium 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 the Intel Broadwell, Skylake and Knights Landing processors, with an additional internal Cray system 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.

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

[8] https://opensbli.github.io

[9] http://www.archer.ac.uk/community/benchmarks/archer/

[10] https://www.metoffice.gov.uk/research/modelling-systems/unified-model

[11] http://www.vasp.at

Processor	Cores	Clock speed GHz	FP64 TFLOP/s	Bandwidth GB/s
Broadwell	2×22	2.2	1.55	154
Skylake (Gold)	2×20	2.4	3.07	256
Skylake (Platinum)	2×28	2.1	3.76	256
Knights Landing	64	1.3	2.66	$\sim$490
ThunderX2	2×32	2.2	1.13	320

Table I
HARDWARE INFORMATION (PEAK FIGURES)

- 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.
- Intel Xeon Phi 7210 (Knights Landing) 64-core @ 1.3 GHz, with 96 GB of DDR4-2400 DRAM and 16 GB of MCDRAM, configured in quad/flat mode (or else quad/cache mode where the problem size exceeded MCDRAM capacity, such as for the SNAP proxy application). RRP $1,881 each.

The recommended retail prices (RRP) are correct at the time of writing (May 2018), and taken from Intel's website[12]. Both the Skylake and Knights Landing processors provide an AVX-512 vector instruction set, meaning that each FP64 vector operation operates on eight elements; by comparison, Broadwell utilises AVX2, which is 256-bits wide, operating on four FP64 elements. The Xeon processors both have three levels of on-chip (data) cache, with an L1 and L2 cache per core, along with a shared L3. The Xeon Phi processor, on the other hand, has two levels of on-chip cache: an L1 cache per core and an L2 cache shared between pairs of cores (*tiles*); the MCDRAM can operate as a large, direct-mapped L3 cache for the off-processor DRAM. 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 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.

We evaluated three different compilers for ThunderX2 in this study: GCC 7, the LLVM-based Arm HPC Compiler 18.2-18.3, and Cray's CCE 8.6–8.7. We believe this is the first study to date that has compared all three of these compilers targeting Arm. The compiler that achieved the highest performance in each case was used in the results graphs displayed below. Likewise for the Intel processors, we used GCC 7, Intel 2018, and Cray CCE 8.5–8.7. Table II lists the compiler that achieved the highest performance for each benchmark in this study.

[12]https://ark.intel.com/

Benchmark	ThunderX2	Broadwell	Skylake	Xeon Phi
STREAM	GCC 7	Intel 18	Intel 18	Intel 18
CloverLeaf	Arm 18.2	Intel 18	Intel 18	Intel 18
TeaLeaf	GCC 7	Intel 18	Intel 18	Intel 18
SNAP	CCE 8.6	Intel 18	Intel 18	Intel 18
Neutral	GCC 7	Intel 18	Intel 18	Intel 18
CP2K	GCC 7	GCC 7	GCC 7	—
GROMACS	GCC 7	GCC 7	GCC 7	—
NAMD	Arm 18.2	GCC 7	Intel 18	—
NEMO	CCE 8.7	CCE 8.7	CCE 8.7	—
OpenFOAM	GCC 7	GCC 7	GCC 7	—
OpenSBLI	CCE 8.7	Intel 18	Intel 18	—
UM	CCE 8.6	CCE 8.5	CCE 8.6	—
VASP	CCE 8.7	CCE 8.6	CCE 8.6	—

Table II
COMPILERS USED FOR BENCHMARKING

B. Mini-apps

Figure 2 compares the performance of our target platforms over a range of representative mini-applications.

STREAM: The STREAM benchmark measures the sustained memory bandwidth from the main memory. For processors using traditional DRAM technology, the available memory bandwidth is essentially determined by the number of memory controllers. Intel Xeon Broadwell and Skylake processors have four and six memory controllers per socket, respectively. The Cavium ThunderX2 processor has eight memory controllers per socket. The results in Figure 2 show a clear trend that Skylake achieves a $1.6\times$ improvement over Broadwell, which is to be expected, given Skylake's faster memory speed (2666 MHz DDR4 vs. Broadwell's 2400 MHz). The eight memory controllers per socket on the Cavium ThunderX2 achieve a $1.92\times$ speedup over Broadwell.

Broadwell and Skylake are achieving 84.4% and 80.4% of theoretical peak memory bandwidth, respectively. At 249 GB/s, ThunderX2 is achieving 77.8% of theoretical peak memory bandwidth. On ThunderX2, we found that the best STREAM Triad performance was achieved with running 16 OpenMP threads per socket (32 in total), rather than one thread per core (64 total). The CrayPAT tool reports rates of 69.4% D1 hit and 66.9% D2 hit for 32 threads, whereas for 64 threads rates of 66.6% D1 hit and 38.5% D2 hit are reported; notice that the D2 hit rate has dropped sharply. On the 18-core Broadwell CPUs in the Isambard Phase 1 system, using the Cray compiler, CrayPAT reports 66.7% D1 and 11.3% D2 hit rates with 36 threads; the numbers are similar with only half the number of threads. The low

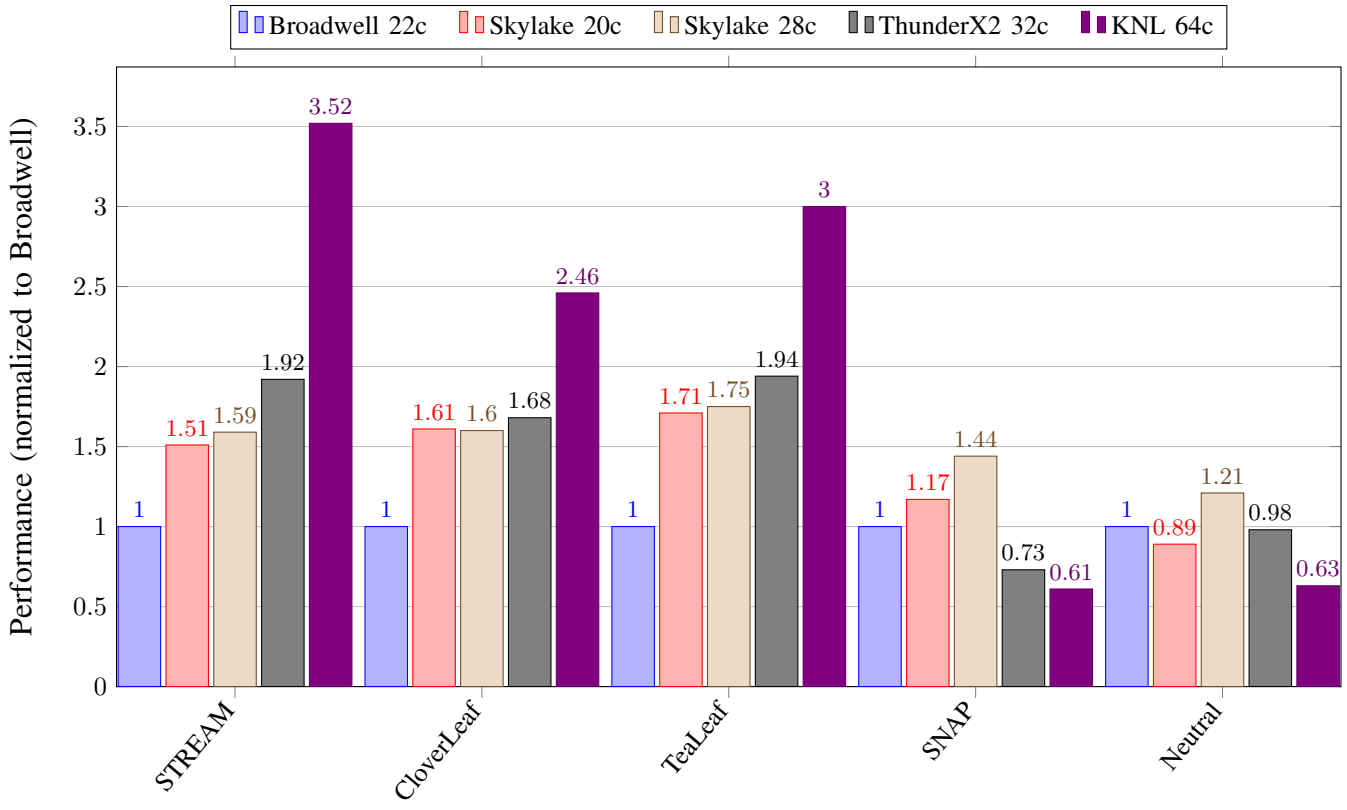

Figure 2. Comparison of Broadwell, Skylake, Knights Landing and ThunderX2 for a range of mini-apps. Results are normalized to Broadwell.

L2 hit rate is showing that there is a strong reliance on the main memory bandwidth for data traffic, as expected.

The MCDRAM technology used by the Intel Xeon Phi (Knights Landing) processor offers improved memory bandwidths over traditional DRAM technology and this is clearly seen in Figure 2, with speedups of 3.5× over the Broadwell baseline. A percentage of theoretical peak memory bandwidth is difficult to calculate as Intel does not publish this figure, but rather publishes an expected performance from benchmarks similar to STREAM.

The use of non-temporal store instructions is an important optimisation for the STREAM benchmark, as Raman et al showed, where, for the Triad kernel, the use of non-temporal store instructions resulted in a 37% performance improvement [17]. On the Intel architecture, using these instructions for the write operation in the STREAM kernels ensures that the cache is not polluted with the output values, which are not re-used; note that it is assumed that the arrays are larger than last-level cache. As such, if this data occupied space in the cache, it would reduce the capacity available for the other arrays which are being prefetched into cache. The construction of the STREAM benchmark with arrays allocated on the stack with the problem size known at compile time allows the Intel compiler to generate non-temporal store instructions for all the Intel architectures in this study. Although the GCC compiler does not generate non-temporal stores for the Cavium ThunderX2 architecture (in fact it cannot generate non-temporal store instructions for any architecture), the implementation of these instructions within the ThunderX2 architecture does not result in a bypass of cache. Instead, the stores still write to L1 cache, but in a way that exploits the write-back policy and *least recently used* eviction policy to limit the disruption on cache. As such, this may be limiting the achieved STREAM performance on ThunderX2, as memory bandwidth is being wasted evicting the output arrays of the kernels. Despite this lack of true streaming stores, it is clear that the additional memory controllers on the ThunderX2 processors provide a clear external memory bandwidth advantage over Broadwell and Skylake processors.

CloverLeaf: The normalised results for the CloverLeaf mini-app in Figure 2 are very consistent with those for STREAM on the Intel Xeon and Cavium ThunderX2 processors. CloverLeaf is a structured grid code and the vast 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 [4]. The same is true on the processors in this study, with the improvements on ThunderX2 coming from the additional memory controllers. 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 which use traditional DRAM technology. The High Bandwidth Memory technology in the form of MCDRAM is what gives the Knights Landing processor its further performance improvement.

It has been noted that the time per iteration increases as the simulation progresses on the ThunderX2 processor. This phenomenon is not noticeable on the x86 processors. We believe this is due to the data-dependent need for floating-point intrinsic functions (such as `abs` and `min` and `sqrt`); this can be seen in a `viscosity` kernel, for instance. As the time iteration progresses, the need for such functions is higher and, therefore, the kernel increases in runtime. Although these kernels are memory bandwidth–bound, the increased number of floating-point operations increases the computational intensity and is therefore slightly less bandwidth-bound (under the Roofline model).

TeaLeaf: The TeaLeaf mini-app again tracks the memory bandwidth performance of the processors. Indeed, this was shown previously on x86 and GPU architectures [5]. The additional memory bandwidth on the ThunderX2 processor clearly improves the performance over those processors with fewer memory controllers.

SNAP: Understanding the performance of the SNAP proxy application is difficult [7]. If truly memory bandwidth–bound, the MCDRAM memory available on the Knights Landing processor would increase the performance as with the other memory bandwidth–bound codes discussed in this study; however, the Knights Landing run results in an *increased* runtime over our Broadwell baseline. Similarly, the ThunderX2 processor does not provide an improvement over the baseline for the tested problem for this code. The Skylake processor does give an improvement, however, and whilst this does have memory bandwidth improvements over Broadwell, this is not the only significant architectural change. Skylake has 512-bit vectors, which creates a wide data path through the cache hierarchy—a whole cache line is loaded into registers for each load operation. In comparison, Broadwell has 256-bit vectors and the ThunderX2 has 128-bit vectors, moving half and a quarter of a 64 byte cache line per load operation, respectively. The wider vectors of the x86 processors improve the cache bandwidth, and for Knights Landing the cache bandwidth is limited by the clock speed, despite the use of 512-bit vectors [18].

The main sweep kernel in the SNAP code requires that a cache hierarchy support both reading (and writing) a very large data set and simultaneously keeping small working set data in low levels of cache. The CrayPAT profiler reports the cache hit rates for L1 and L2 caches when using the Cray compiler. On the ThunderX2, the hit rates for the caches are both at around 84%, which is much higher than the hit rate for the STREAM benchmark (where the latter is truly main memory bandwidth–bound). As such, this shows that the SNAP proxy application is heavily reusing data in cache, and so performance of the memory traffic to and from cache is a key performance factor. On the Broadwell processors in Isambard Phase 1, CrayPat reports D1 hit rates of 89.4% and D2 hit rates of 24.8%. Again, the L1 cache hit rate is much more than in the STREAM benchmark, indicating high reuse of data in the L1 cache, but that the L2 is not used as efficiently. It is the subject of future work to understand how the data access patterns of SNAP interact with the cache replacement policies in these processors. However, for this study, it is clear that main memory bandwidth is not necessarily the performance limiting factor, but rather it is the access to the cache where the x86 processors have an advantage.

Neutral: In previous work, it was shown that the Neutral mini-app has algorithmically little cache reuse, due to the random access to memory required for accessing the mesh data structures [8]. Additionally, the access pattern is data driven and this not predictable, and so any hardware prefetching of data into cache according to common access patterns is likely to be ineffective, resulting in high numbers of cache misses. Indeed, CrayPAT shows a low percentage (27.3%) of L2 cache hits on the ThunderX2 processor, and a similar percentage on Broadwell. The L1 cache hit rate is rather high on both architectures, with over 95% hits. As such, the extra memory bandwidth available on the ThunderX2 or the Intel Xeon Phi does not provide an advantage over the Intel Xeon processors. Note from Figure 2 that the ThunderX2 processor still achieves around parity performance with Broadwell, with Skylake 28c only offering a small performance improvement of about 21%.

Summary: Many of these results highlight the superior memory bandwidth offered by the ThunderX2's eight memory channels, which deliver 249 GB/s for the STREAM Triad benchmark [2]—twice that of Broadwell and 35% more than Skylake. This performance increase can be seen in the memory bandwidth–bound mini-apps such as CloverLeaf and TeaLeaf, with the ThunderX2 processor showing similar improvements to STREAM over the x86 processors. The MCDRAM technology present on the Knights Landing again shows further improvements, highlighting that the performance of these mini-apps is determined by the available memory bandwidth of the processor.

The SNAP and Neutral mini-apps, however, rely more on the on-chip memory architecture (the caches), and so they are unable to leverage the external memory bandwidth on all processors. As such, the additional memory controllers on the ThunderX2 processors do not seem to improve the performance of these mini-apps relative to processors with less memory bandwidth. Note, too, that although the MCDRAM technology provides greater memory bandwidth, it again does not improve the performance. The exact nature of

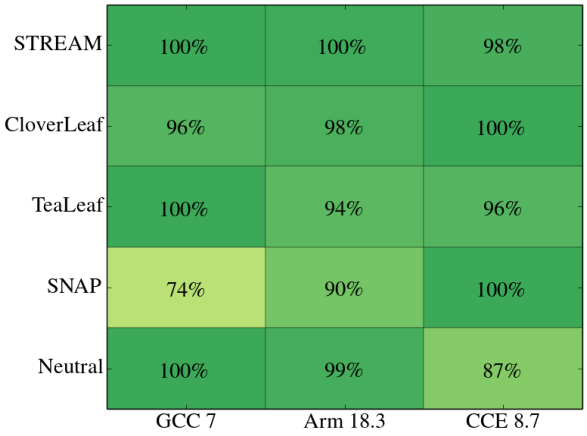

Figure 3. Efficiency of different compilers running on Cavium ThunderX2.

the interaction of these algorithms with the cache hierarchy is the subject of future study.

Figure 3 compares the latest versions of the three available compilers on the ThunderX2 platform, normalised to the best performance observed for each mini-app. The benefits of having multiple compilers for Arm processors is clear, as none of the compilers dominate performance. The Arm HPC compiler uses the LLVM-based Flang, a relatively new Fortran frontend, which produces a miscompilation for TeaLeaf. SNAP and Neutral have more complex computational kernels than the other mini-apps that draw out differences in the optimisations applied by the compilers.

C. Applications

Figure 4 compares the performance of dual-socket Broadwell, Skylake and ThunderX2 systems for the real application workloads described in Section III-B[13].

CP2K: CP2K comprises many different kernels that have varying performance characteristics, including floating-point-intensive routines and those that are affected by external memory bandwidth. While the floating-point routines run up to $3\times$ faster on Broadwell compared to ThunderX2, the improved memory bandwidth and higher core counts provided by ThunderX2 allow it to reach a $\sim 15\%$ speed-up over Broadwell for this benchmark. The 28c Skylake processor provides even higher floating point throughout and closes the gap in terms of memory bandwidth, yielding a further 20% improvement over ThunderX2. The H2O-64 benchmark has been shown to scale sublinearly when running on tens of cores [19], which impacts on the improvements that ThunderX2 can offer for its 64 cores.

GROMACS: The GROMACS performance results are influenced by two main factors. First, the application is heavily compute-bound, and the x86 platforms are able to exploit their wider vector units and wider datapaths to cache. Performance does not scale perfectly with vector width due to the influence of other parts of the simulation, in particular the distributed FFTs. Secondly, because GROMACS uses hand-optimised vector code for each platform, x86 benefits from having the more mature implementation, one that has evolved over many years. Since Arm HPC platforms are new, it is likely that the NEON implementation is not yet at peak efficiency for ThunderX2.

NAMD: As discussed in Section III-B, NAMD is not clearly bound by a single factor, and thus it is hard to underline a specific reason why one platform is slower (or faster) than another. It is likely that results are influenced by a combination of memory bandwidth, compute performance and other latency-bound operations. The results observed do correlate with memory bandwidth, making Broadwell the slowest platform of the three for this application. Running more than one thread per core in SMT increased NAMD performance, and this is the most pronounced on the ThunderX2, which can run 4 hardware threads on each physical core. Furthermore, due to NAMD's internal load balancing mechanism, the application is able to efficiently exploit a large number of threads, which confers yet another advantage to ThunderX2 for being able to run more threads (256) than the x86 platforms.

NEMO: For the NEMO benchmark, ThunderX2 is $1.4\times$ faster than Broadwell 22c, but 11% slower than Skylake 28c. While the benchmark should be mostly memory bandwidth bound, leading to significant improvements over Broadwell, the greater on-chip cache bandwidth of Skylake gives it a slight performance advantage over ThunderX2. Running with multiple threads per core to ensure that the memory controllers are saturated provides a small improvement for ThunderX2.

OpenFOAM: The OpenFOAM results follow the STREAM behaviour of the three platforms closely, confirming that memory bandwidth is the main factor that influences performance here. With its eight memory channels, ThunderX2 yields the fastest result, at $1.9\times$ the Broadwell performance. Skylake is able run $1.67\times$ faster than Broadwell, i.e. a bigger difference than in plain STREAM, because it is likely able to get additional benefit from its improved caching, which is not a factor in STREAM. This benchmark strongly highlights ThunderX2's strength in how performance can be improved significantly by higher memory bandwidth.

OpenSBLI: The OpenSBLI benchmark exhibits a similar performance profile to OpenFOAM, providing another workload that directly benefits from increases in external memory bandwidth. The ThunderX2 system produces speed-ups of $1.73\times$ and $1.13\times$ over Broadwell 22c and Skylake 28c, respectively.

[13]Due to system stability issues on our prototype ThunderX2 Foxconn systems, we were unable to run the memory clock at the 2500 MHz speed for CP2K, Unified Model and VASP, instead running at 2200 MHz.

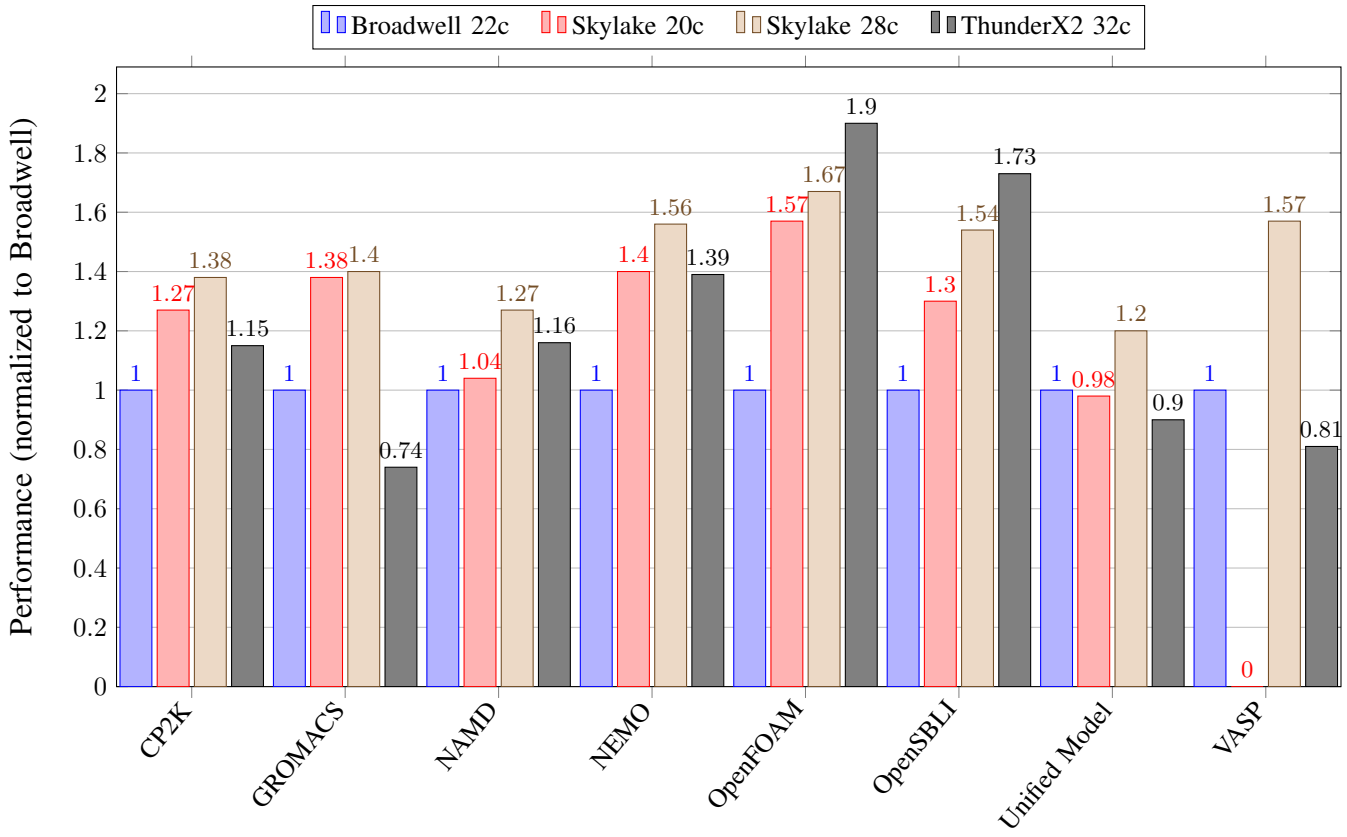

Figure 4. Comparison of Broadwell, Skylake and ThunderX2 for a set of real application codes. Results are normalised to Broadwell.

Unified Model: Comprising two million lines of Fortran, the Unified Model is arguably the most punishing of the benchmarks used in this study, stressing the maturity of the compilers as well as the processors themselves. Skylake 28c only yields a 20% improvement over Broadwell 22c, indicating that performance of this test case is not entirely correlated to memory and cache bandwidth or floating-point compute, and that the relatively low-resolution benchmark may struggle to scale efficiently to higher core counts. The ThunderX2 result is around 10% slower than Broadwell, but demonstrates the robustness of the Cray software stack on Arm systems by successfully building and running without requiring any modifications. Interestingly, when running on just a single socket, ThunderX2 provides a ∼15% improvement over Broadwell. We also observed performance regressions in the more recent versions of CCE on all three platforms; the Broadwell result was fastest using CCE 8.5, which could not be used for either Skylake or ThunderX2.

VASP: The calculations performed by the VASP benchmark are dominated by floating-point-intensive routines, which naturally favour the x86 processors with their wider vector units. While the higher core counts provided by ThunderX2 make up for some of the difference, the VASP benchmark exhibits a similar profile to GROMACS, with ThunderX2 around ∼20% slower than Broadwell 22c, and sitting around half the speed of Skylake 28c.

Performance per Dollar: So far we have focused purely on performance, but one of the main advantages of the Arm ecosystem for HPC should be the increased competition it brings. While performance per Dollar is hard to quantify rigorously, and price is always subject to negotiation, we can still reveal some illuminating information by using the published list prices of the CPUs to compare the respective performance per Dollar of the processors. In the results to follow, we only take the CPU list prices into account. This deliberately leaves every other factor out of the comparison; instead, factoring our results into a subjective whole system cost is left as an exercise for the expert reader. Figures 5 and 6 show the Performance per Dollar of our platforms of interest, normalised to Broadwell 22c. The numbers are calculated by taking the applications performance numbers shown in Figures 2 and 4 and simply dividing them by the published list prices described in Section IV-A, before renormalising against Broadwell 22c.

There are a number of interesting points to highlight. First considering the mini-apps in Figure 5, we see that ThunderX2's RRP of just $1,795 gives it a compelling advantage compared to all the other platforms. ThunderX2

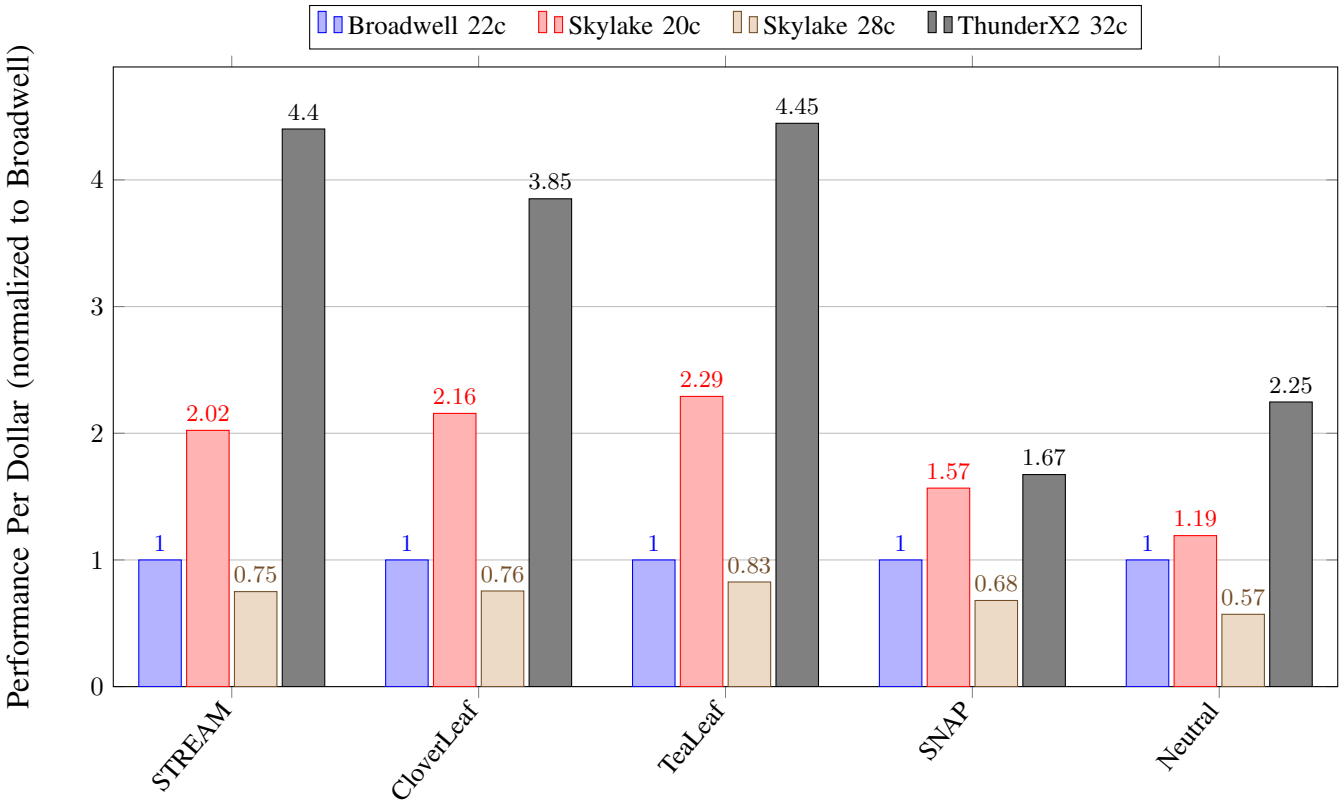

Figure 5. Performance Per Dollar Comparison of Broadwell, Skylake and ThunderX2 for a range of mini-apps. Results are normalised to Broadwell.

consistently comes ahead of not just the top-bin and therefore expensive Skylake 28c, but also the much more cost conscious, mainstream Skylake 20c SKU. The picture for real applications in Figure 6 is similar, where even in cases when ThunderX2 had a performance disadvantage, such as in the compute bound codes GROMACS and VASP, ThunderX2 becomes competitive once cost is considered. Where ThunderX2 was already competitive on performance, adding in cost makes ThunderX2 look even more competitive, often achieving a performance/price advantage over even the more cost oriented Skylake 20c SKUs of 2x or more.

Summary: Overall, the results presented in this section demonstrate that the Arm-based Cavium 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. 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. When processor cost is taken into account, ThunderX2's proposition is even more compelling. With multiple production quality compilers now available for 64-bit Arm processors, the software ecosystem has reached a point where developers can have confidence that real applications will build and run correctly, in the vast majority of cases with no modifications.

Some of the applications tested highlight the lower floating-point throughput and L1/L2 cache bandwidth of ThunderX2. Both of these characteristics stem from the narrower vector units relative to AVX-capable x86 processors. In 2016, Arm unveiled the Scalable Vector Extension ISA [20], which will enable hardware vendors to design processors with much wider vectors of up to 2,048-bits, compared to the 128-bits ot today's NEON. We therefore anticipate that the arrival of SVE-enabled Arm-based processors in the next couple of years will likely address most of the issues observed in this study, enabling Arm processors to deliver even greater performance for a wider range of workloads.

V. 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. 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

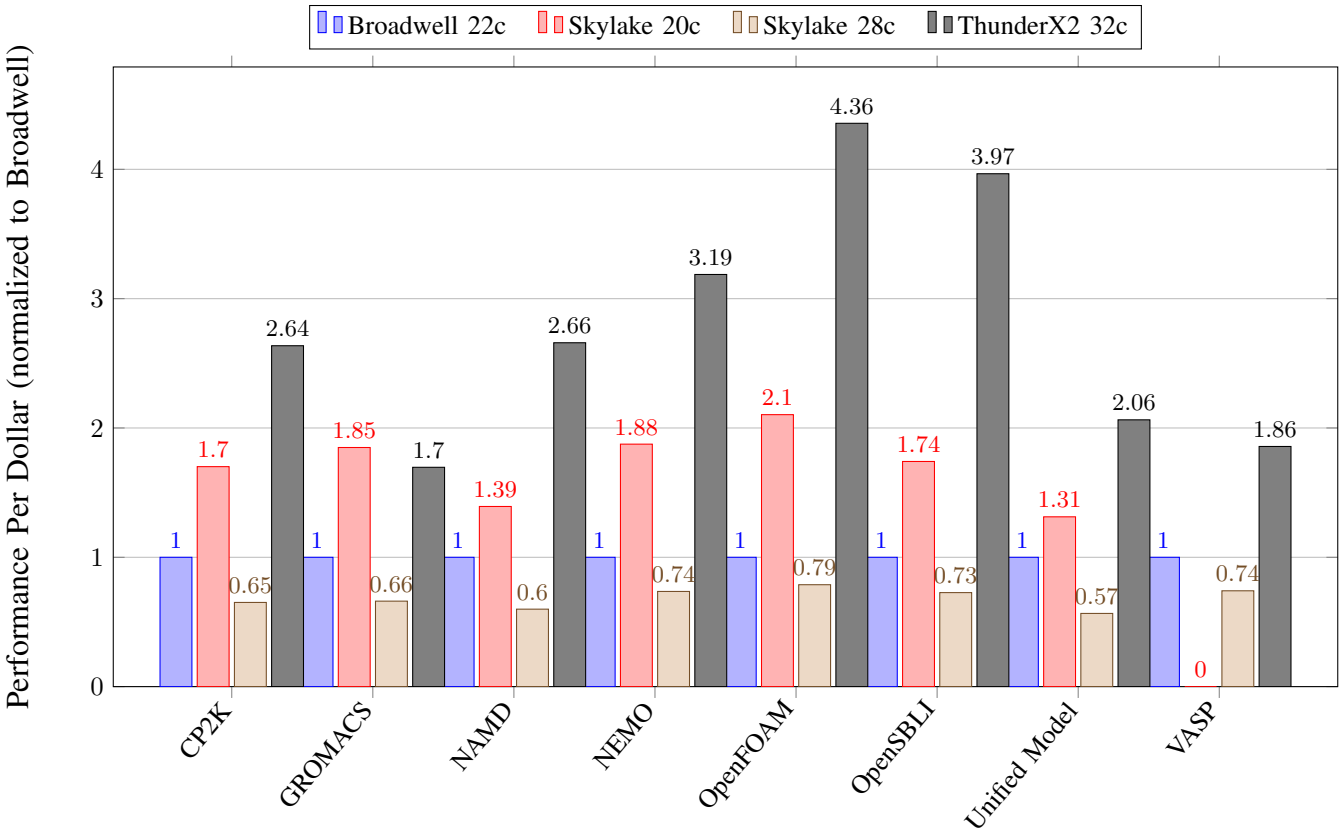

Figure 6. Performance Per Dollar Price Comparison of Broadwell, Skylake and ThunderX2 for our application codes. Results are normalised to Broadwell.

maturity of the Arm ecosystem for HPC, where these processors can now be considered as viable contenders for future procurements. Future work will use the full Isambard system to evaluate production applications running at scale on ThunderX2 processors.

We did not address energy efficiency in this paper. Our early observations suggest that the energy efficiency of ThunderX2 is in the same ballpark as the x86 CPUs we tested. This is not a surprise — for a given manufacturing technology, a FLOP will take a certain number of Joules, and moving a byte so many millimeters across a chip will also take a certain amount of energy. There is no magic, and the instruction set architecture has very little impact on the energy efficiency when most of the energy is being spent moving data and performing numerical operations.

Overall, these results suggest that Arm-based server CPUs that have been optimised for HPC are now genuine options for production systems, offering performance 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 (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 behind the results in this paper, and to Federica Pisani from Cray who organised the events.

Attendees of the hackathons in November 2017 and March 2018 included (listed alphabetically): David Acreman, Lucian Anton, Andy Bennett, Charles Bentham, Steven Chapman, Steven Daniels, Luiz DeRose, Christopher Edsall, Pawan Ghildiyal, Andreas Glöss, Matthew Glover, Indraneil Gokhale, James Grant, Thomas Green, Alistair Hart, Ian Harvey, Phil Hasnip, Mattijs Janssens, Hrvoje Jasak, Jess Jones, Merzuk Kaltak, Ilias Katsardis, JaeHyuk Kwack,

Alfio Lazzaro, Unai Lopez-Novoa, David Lusher, Simon McIntosh-Smith, Sachin Nanavati, Szilárd Páll, Pablo Ouro, Andrei Poenaru, Iakov Polyak, Heidi Poxon, James Price, Harvey Richardson, Kevin Roy, Jonathan Skelton, Craig Steffen, Adrian Tate, Adam Voysey and Christopher Woods.

Access to the Cray XC40 supercomputer 'Swan' 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.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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/

[6] 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.

[7] T. Deakin, W. Gaudin, and S. McIntosh-Smith, "On the Mitigation of Cache Hostile Memory Access Patterns on Many-Core CPU Architectures." Frankfurt: Springer International Publishing, 2017, pp. 348–362. [Online]. Available: http://link.springer.com/10.1007/978-3-319-67630-2{_}26

[8] M. Martineau and S. McIntosh-Smith, "Exploring On-Node Parallelism with Neutral, a Monte Carlo Neutral Particle Transport Mini-App," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, sep 2017, pp. 498–508. [Online]. Available: http://ieeexplore.ieee.org/document/8048962/

[9] S. Pll 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

[10] M. T. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. V. Kal, R. D. Skeel, and K. Schulten, "Namd: a parallel, object-oriented molecular dynamics program," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 10, no. 4, pp. 251–268, 1996. [Online]. Available: https://doi.org/10.1177/109434209601000401

[11] J. C. Phillips, Y. Sun, N. Jain, E. J. Bohm, and L. V. Kale, "Mapping to Irregular Torus Topologies and Other Techniques for Petascale Biomolecular Simulation," in *Proceedings of ACM/IEEE SC 2014*, New Orleans, Louisiana, November 2014.

[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*, vol. 1000. IUC Dubrovnik, Croatia, 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] C. Hall, "The UK Meteorological Office climate model: The AMIP run and recent changes to reduce the systematic errors," *WORLD METEOROLOGICAL ORGANIZATION-PUBLICATIONS-WMO TD*, pp. 301–306, 1995.

[15] 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/

[16] 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.

[17] K. Raman, T. Deakin, J. Price, and S. McIntosh-Smith, "Improving Achieved Memory Bandwidth from C++ Codes on Intel Xeon Phi Processor (Knights Landing)," Presentation at International Xeon Phi User Group Spring Meeting, Cambridge, UK, April 2017.

[18] T. Deakin, J. Price, and S. McIntosh-Smith, "Portable Methods for Measuring Cache Hierarchy Performance (poster)," in *Supercomputing*, Denver, Colorado, 2017.

[19] I. Bethune, F. Reid, and A. Lazzaro, "CP2K performance from Cray XT3 to XC30," 2014.

[20] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu *et al.*, "The ARM Scalable Vector Extension," *IEEE Micro*, vol. 37, no. 2, pp. 26–39, 2017.