Evaluating the Marvell ThunderX2 Server Processor for HPC Workloads

S.D. Hammond, C. Hughes, M.J. Levenhagen, C.T. Vaughan, A.J. Younge,
B. Schwaller, M.J. Aguilar, K.T. Pedretti and J.H. Laros

Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, USA

{sdhammo, chughes, mjleven, ctvaugh, ajyoung, bschwal, mjaguil, ktpedre, jhlaros}@sandia.gov

***Abstract*—The high performance computing industry is undergoing a period of substantial change. Not least because of fabrication and lithographic challenges in the manufacturing of next-generation processors. As such challenges mount, the industry is looking to generate higher performance from additional functionality in the micro-architecture space as well as a greater emphasis on efficiency in the design of network-on-chip resources and memory subsystems. Such variation in design opens opportunities for new entrants in the data center and server markets where varying compute-to-memory ratios can present end users with more efficient node designs for particular workloads.**

In this paper we compare the recently released Marvell ThunderX2 Arm processor - arguably the first high-performance computing capable Arm design available in the marketplace. We perform a set of micro-benchmarking and mini-application evaluation on the ThunderX2 comparing it with Intel's Haswell and Skylake Xeon server parts commonly used in contemporary HPC designs. Our findings show that no one processor performs the best across all benchmarks, but that the ThunderX2 excels in areas demanding high memory bandwidth due to the provisioning of more memory channels in its design. We conclude that the ThunderX2 is a serious contender in the HPC server segment and has the potential to offer supercomputing sites with a viable high-performance alternative to existing designs from established industry players.

***Index Terms*—High-Performance Computing, Server, Processor, Performance, Evaluation**

I. Introduction

Processor and computing architecture designs are going through a period of unprecedented change. There are multiple drivers placing pressure on historical approaches that include the, now well documented, challenges with advanced lithographic nodes and silicon manufacture, as well as changing dynamics in the computing marketplace towards a greater use of machine learning and data analytics. In the consumer market, there are also notable shifts away from the use of larger powerful desktops towards lightweight tablet and mobile devices, which help to place pressure on amortizing expensive fabrication facilities over more expensive units.

The effect of these factors on high-performance computing (HPC) has the potential to be dramatic. For the past two decades, supercomputers have typically utilized high-end commodity processor parts in large numbers to derive high aggregate system performance, with a near certainty that advances in lithographic processes correlated to Moore's Law, would continue to drive economical production of significantly faster designs year-upon-year. The situation today is different. Performance is no longer guaranteed to improve as the rate of lithographic improvements slow down, and the ability of high-performance computers to drive new features in processor cores becomes more challenging. Instead, hardware designers are marking out large parts of the silicon real estate to drive sales in alternative markets, shifting toward the use of very low-level micro-architecture optimizations as a way to deliver additional performance. Some of these recent designs include the use of wider vectors for compute throughput, redesigned cache/memory subsystems and changes in the design of system-on-chip or multi-processor packages to boost performance.

While these changes create an interesting period in computing for the research community, the much greater variation in processor and micro-architecture designs is creating higher-levels of complexity for large-scale supercomputing sites, which must weigh the more varied, and often more specialized novel features, against existing workloads that may not yet be ready to run on new hardware, or many not have the fundamental capability to exploit the novel hardware components being offered. The much greater number of options, which has been called the HPC "Cambrian" explosion, is therefore placing real pressure on system procurement and benchmarking specialists to navigate the much broader range of components being offered in order to select the best combination for each sites' HPC needs.

Recognizing the challenges in these areas and the potential of the changes to add significant opportunity and technical risk, the United States Department of Energy (DOE) National Nuclear Security Administration (NNSA) launched the Vanguard program [1] in 2018 to field moderate-scale, next-generation, prototype computing platforms. The intended users of these machines are application developers and systems engineers, giving them the ability to evaluate, typically, off-roadmap system components for performance and scalability, ensuring that any future system would meet the robust requirements that are placed on the systems that are used within the national security mission space.

The first system selected for deployment in the Vanguard program was Astra [2][3] - the first petascale supercomputer based on Arm processors. While [4] and [5] previously investigated the potential for Arm processors, Astra was intended to conduct these activities at a much larger scale. Specifically, Astra utilizes the ThunderX2 processor developed by Marvell to be a fully featured, server-class Arm design that can provide exceptional memory bandwidth while attempting to strike a more optimized balance between compute and data throughput requirements.

In this paper, we detail our early benchmarking of Astra's Arm-based processor compute nodes using kernels and mini-applications, comparing our results to those from Intel's Skylake [6] and Haswell [7][8] server/HPC offerings, extending the first comparison of Skylake in [9]. The specific contributions of this paper are as follows:

- **Performance Benchmarking of the ThunderX2 Processor** - we report on our initial experiences benchmarking the production silicon variants of Marvell's ThunderX2 processor used in the first petascale Arm supercomputer. In particular, this paper is the first to report on a broad suite of micro-benchmarking results of typical processor performance bottlenecks, which have been identified as critical aspects to important classes of DOE and NNSA/ASC workloads.
- **Mini-Application Benchmarking of the ThunderX2 Processor** - we include three important DOE mini-applications – LULESH, XSBench and HPCG – in the evaluation, showing how relevant high-performance kernels perform on the platform. While the micro-benchmarking results are indicative of the mini-application performance trends, the composition of low-level operations in the mini-application kernels present a more complex performance picture.
- **Comparison to Intel Skylake and Haswell Server Processors** - finally, we run the micro-benchmarks and mini-applications on Intel's server-class Skylake and Haswell Xeon processors, comparing them to the performance achieved on Marvell's ThunderX2. We describe the significantly different design optimizations that have been employed in each processor and highlight how these may change benchmark performance. We show that for some kernels, ThunderX2 provides the highest performance while for others, the Intel Skylake is the highest performer.

The remainder of this paper is laid out as follows: Section II provides an overview of the systems and processors used for our benchmarking studies, in particular, we comment on the different clock frequencies, core counts and cache structures in each design. Section III presents results from low-level microbenchmarking of the three processors. In Section IV we provide results of benchmarking three important DOE mini-applications and, finally, we conclude our discussion in Section V.

II. Benchmark Systems

A. *Marvell ThunderX2 (Astra)*

Astra, the first medium-scale prototype in the Vanguard program, is a 2,592 compute-node supercomputer deployed at Sandia National Laboratories. Each compute node is comprised of dual socket 28-core Marvell ThunderX2 processors; the block diagram for the processor is shown in Figure 1. We note that the 28-core design is the same as the 32-core design but with one ring stop disabled by processor yield. Processor cores are clocked at 2.0GHz and have the ability to execute in either 1, 2 or 4-way hardware multi-threading. For our benchmarking activities, the nodes were configured for SMT-1 execution as this was the mode provided during early system benchmarking and evaluation. Each core has private 32kB L1 data and instruction caches and a private 256kB 8-way associated L2 cache. 1MB of distributed L3 cache per core (for a total of 28MB) is provided across the inter-core ring network-on-chip. Each socket provides 64GB of system memory (for a total of 128GB per node) that is arranged as 8GB DIMMs for each of the 8 channels of 2666MT/s DDR4 memory. Although beyond the scope of this initial on-node performance evaluation paper, Astra compute nodes utilize a Mellanox ConnectX-5 OCP EDR InfiniBand network interface (100Gb/s) that is organized to provide a direct link to each socket for more uniform network access characteristics. For the ThunderX2 compute nodes of Astra, we utilize RedHat Enterprise Linux 7.6 and, unless noted, the Arm 19.1 production compiler. For compile flags we compile each benchmark using the `-mcpu=thunderx2t99` and `-mtune=thunderx2t99` architecture tuning flags (as recommended by Arm and Marvell [10]).

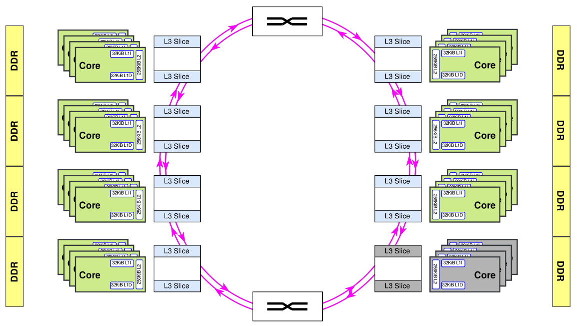


Fig. 1: Cavium ThunderX2 CN99XX Block Diagram

B. *Intel Skylake Platinum 8160 (Blake Testbed)*

The Skylake Xeon processors used in this study are are provided by Sandia's *Blake* system. Each node provides dual-socket Xeon Platinum 8160 processors at 2.1GHz. Each socket contains 24 cores with SMT-2 enabled per core, thus, 96 hardware threads can be used per node. 192GB of 2666MT/s DDR4 system memory is provided for both sockets (96GB per socket, spread across 6 memory channels). The processor cores each have a private L1 and L2 cache of 32kB and 1MB respectively (note that this is a 4x increase in L2 cache size per core over Haswell). The L1 caches are 8-way set associative, while the L2 cache is 16-way set associative. A 33MB L3 distributed cache is provided across the 2D processor mesh. Unless otherwise noted, we use the Intel 18.1 compiler toolchain with GCC 4.9.3 compatibility mode enabled, running on RedHat Enterprise Linux 7.4. For architecture optimization flags, we use `-xcore-avx512` to generate efficient vectorized code sequences.

C. *Intel Haswell E5-2697v3 (Mutrino Testbed)*

The Sandia Haswell development system, *Mutrino*, provides dual-socket Xeon E5-2697v3 processors that run at 2.3GHz. Each processor is comprised of 16-cores with SMT-2 enabled per core for a total of 64 hardware threads per node. A total of 128GB of 2133MT/s DDR4 system memory is available per node (64GB spread across 4 channels per socket). Each processor core has a private 32kB L1 cache and a private 256kB L2 cache, each with 8-way associativity. Each socket has a large 40MB L3 cache that is distributed across the ring bus. Unless otherwise noted, we use the Intel 18.1 compiler toolchain running on Cray Compute Node Linux. To produce architecture-optimized code, we use the `-xcore-avx2` flag.

III. Microbenchmarking of ThunderX2

In this section of the paper we report on a collection of basic micro-benchmark results taken from each of the systems described in Section II. The micro-benchmarks selected have historically been used as assessments of processor performance boundaries. Typical HPC codes do not typically perform operations at the same level of intensity as these micro-benchmarks but nonetheless, our micro-benchmarking efforts help in our understanding of potential bottlenecks and expected performance. In order to maintain a uniform benchmarking environment, our benchmark values are reported as either an arithmetic or harmonic mean, depending on the benchmark figure of merit (rates utilize harmonic mean), over the course of at least ten runs. We randomly select one node per run to aggregate performance data over a set of compute nodes, allowing for individual processor variation. For multi-threaded and MPI-based runs, we pin either a thread or process to an individual core, as appropriate, to reduce noise and interference.

A. *Memory Bandwidth (STREAM)*

One of the most attractive hardware features of the ThunderX2 processor is the greater number of memory channels available on the processor (eight per socket as opposed to six on Skylake and four on Haswell). For applications or kernels, which heavily utilize memory bandwidth, the TX2-based nodes would therefore provide a significant performance advantage over the Intel offerings. However, a trend in recent processor designs has seen peak memory bandwidth outpace the on-die transport capabilities of the network-on-chip. This allows

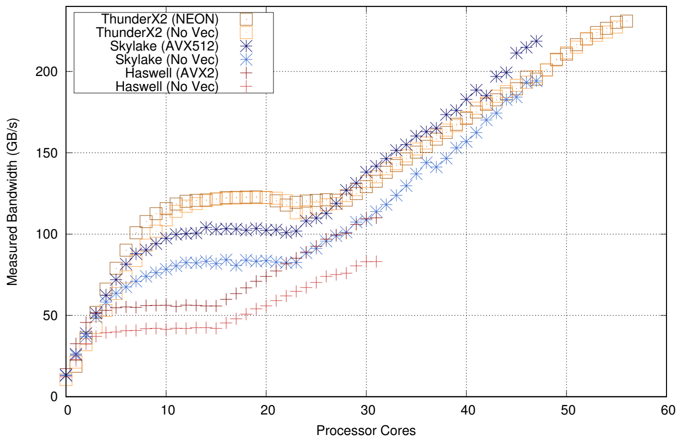


Fig. 2: Benchmarked STREAM Triad memory bandwidth on dual-socket ThunderX2, Haswell and Skylake nodes in GB/s; higher is better. All processors operate in SMT-1 mode; problem size is approx 2.2GB.

for memory DIMMs to operate at lower-load levels, providing more responsive performance, but negating the use of peak memory bandwidth as a guidance in system acquisitions.

In Figure 2, we present benchmarked memory bandwidth on the systems using the ubiquitous STREAM benchmark by McAlpin [11]. Two results are provided for each system - execution with vectorization enabled and with vectorization disabled. For the reader's reference, theoretical peak values for the ThunderX2, Skylake and Haswell processors are 317GB/s, 238GB/s and 143GB/s respectively.

The first observation of note is that the extra memory channels on the ThunderX2 provide the highest memory bandwidth of all three systems at 230.98GB/s. Although not shown here due to inclusion of intrinsics, a more aggressive hand-written version of Triad has exceeded 247GB/s in more recent tests. The Skylake node provides 218.65GB/s of Triad bandwidth and the Haswell 110.01GB/s. Skylake is comfortably the most efficient processor in terms of peak bandwidth at close to 92%, while the ThunderX2 and Haswell nodes are at around 77%.

Our results show that on Intel systems, the ability of application codes to vectorize plays a significant role in achieved bandwidth with un-vectorized code delivering a maximum of 194GB/s on the Skylake and 83GB/s on the Haswell, reducing the peak efficiencies to 81% and 58%, respectively. On the ThunderX2, the difference between vectorized code (using the Arm NEON instruction set) is approximately 25-27GB/s at small thread counts and less than 5GB/s at the largest thread counts. The minor impact that vectorization plays in Triad on the ThunderX2 and, as will be seen in Section IV, other applications, could be due to the fact that the NEON SIMD units are only 128b; half of Haswell and a fourth of Skylake. However, this is not necessarily a detriment to system design.

From these results we are able to show that the ThunderX2 is a strong choice when workloads demand high memory bandwidth, as is often the case with HPC-class applications. Moreover, for applications where vectorization is challenging,

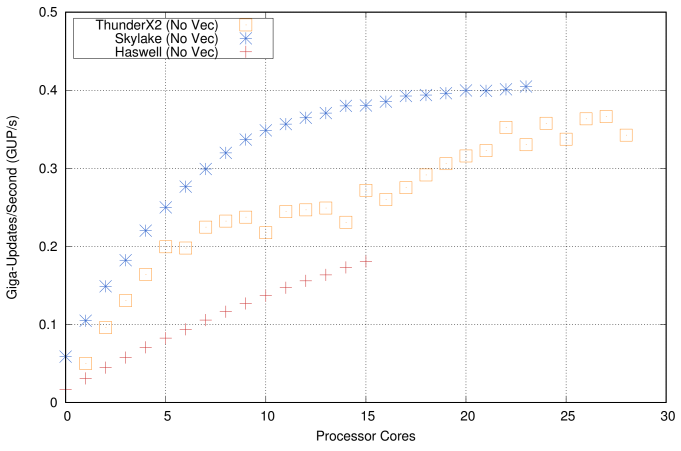


Fig. 3: Random Memory Access for a Single Socket using the Multi-Threaded GUP/s Benchmark. Results are in Giga-Updates per Second, Higher is Better. Problem Size is 32GB, 32,768 iteration updates performed in each benchmark set, single iteration updates a random selection of 64kB of problem data. Standard system page sizes are used to replicate production HPC environment.

either because of algorithm design or because of the limits of automatic vectorization in production compilers, the ThunderX2 is likely to provide an even greater level of performance.

B. *Random Memory Access (GUPs)*

While many HPC applications are optimized to reduce the number of random accesses (since these typically result in poor performance), they cannot be entirely eliminated in algorithms that require complex table-value lookups or have complex control flow. In Figure 3, we present results from running the GUPs (Gigaupdates per-second) RandomAccess benchmark from the HPCC benchmark collection [12]. Historically, this value has been used to evaluate the performance of randomly reading a datum, performing a logical exclusive-or operation with itself and writing it back to the memory system, which replicates randomly-selected read-modify operations over a large data set. The benchmark is typically bottlenecked by inefficient memory subsystem design, such as memory latency and poor TLB implementation.

Although previous studies have shown performance gains by using additional hardware threads to hide operation pipeline latency, the results in Figure 3 are gathered using SMT-1 mode to enable a fair comparison across all of the processors. In addition, we only present the results from the non-vectorized execution since compilation of random update kernels does not typically result in good vector instruction generation and our benchmarking shows no discernible difference in performance.

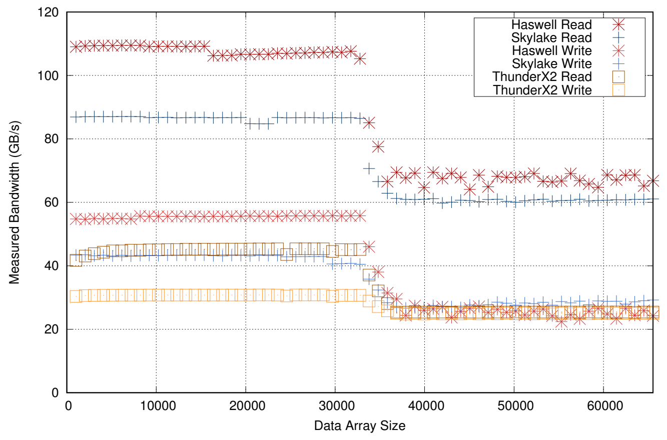
The highest performing processor is the Intel Skylake with the ability to perform almost 0.40GUp/s at peak versus around 0.36GUp/s on ThunderX2 and 0.18GUp/s on Haswell. Neither Skylake nor ThunderX2 demonstrate linear scaling, instead each shows significant performance growth at low thread counts; from 1-8 threads on the TX2 and 1-10 threads on the Skylake. Haswell has a more consistent performance delivery with near linear scaling across all thread counts. We attribute the Skylake's high performance to its higher frequency and design of its on chip mesh-based network-on-chip, which provides much higher path diversity than the ring-based architectures used on the TX2 and Haswell processors.

C. *Per-Core Cache Bandwidth (LMBench [13])*

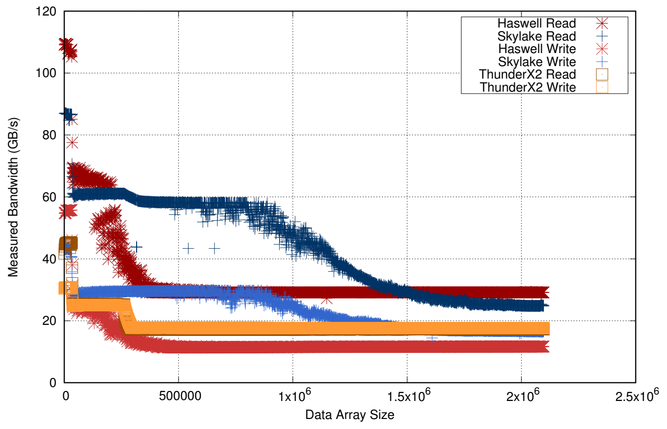
One of the most commonly attempted optimizations for codes that can utilize small working sets is to implement cache blocking or data structure optimizations so that accesses can be performed nearer to the processor, allowing them to benefit from the higher bandwidth and lower latencies of caches. It is worth noting that the design of the ThunderX2 memory subsystem closely resembles that of Intel's Haswell design, which can reduce the need to rearchitect cache-blocked code when porting from one architecture to another. In their Skylake design, Intel substantially changed the memory subsystem design of the processor to move capacity from the L3 distributed cache to the per-core L2 caches. By increasing capacity, Intel is offering a larger collection of applications the ability to benefit from cache performance since working set sizes can now increase from 256kB (as in Haswell and TX2) to 1MB.

Figure 4 shows benchmarked cache bandwidths for all three processors at small problem sizes (4a) that typically fit within the L1 cache up to larger (4b), L3-sized, data sets. We present read and write bandwidths separately as these vary considerably and can significantly change performance if one type of operation dominates the data access pattern. Haswell has the fastest per-core read and write bandwidths at around 111GB/s and 57GB/s - note that Haswell also has the highest clock frequency allowing it to improve data throughput. Skylake provides approximately 88GB/s of read-bandwidth from the L1 as well as 44GB/s of write bandwidth, in keeping with the near 2:1 ratio also found in Haswell. ThunderX2 provides the lowest L1 read and write bandwidths at around 46GB/s and 31GB/s, respectively (note the lower disparity between read and write operations on the TX2 processor).

The longer performance plateau show in Figure 4b for Skylake reflects the larger capacity L2 cache for each core (the Haswell and TX2 data drops much earlier). At large problem sizes that fit into the L3 distributed caches, all three processors provide similar bandwidths - the Haswell core provides the highest bandwidth at 29.8GB/s for reads and 12GB/s for writes; Skylake provides 25.8GB/s for reads and 16.9GB/s for writes; and ThunderX2 provides 17.6GB/s for reads and 18GB/s for writes, again, showing lower disparity between operation types. The result is therefore mixed and heavily dependent on specific application design and access patterns. For cache-read dominated workloads, a Haswell processor will provide the highest performance of a single core. For write dominated workloads, the data set size is critical to processor selection. At small input sizes that fit within the L1, Haswell would provide the highest performance. However, if the application working set size fits into the Skylake's larger L2 but spills from either Haswell or ThunderX2, then the



(a) Small Problem Sizes



(b) Medium Problem Sizes

Fig. 4: Per-Core Cache Bandwidth Measured Using the lmbench Suite (Higher is Better)

Skylake processor core will likely execute fastest. At even larger input sizes that only fit within the L3, the ThunderX2 may provide the best performance.

D. *Double Precision Floating-Point Throughput (DGEMM)*

Double precision compute rates have historically been used as the sizing metric to determine the world's largest supercomputers. In particular, the bi-annual Top-500 supercomputer list uses the High-Performance LINPACK benchmark [14] to determine machine rankings. The emphasis on dense floating-point matrix-matrix operations, particularly DGEMM, has driven the development of wider SIMD vector architectures and the inclusion of more complex floating-point operations in ISAs, such as fused multiply-accumulate (FMA), to increase throughput. Intel has been at the forefront of the widening of SIMD units as a mechanism to substantially increase the throughput of their Xeon server processors. Haswell contains dual FMA-capable vector units that are 256-bit wide. In Skylake, Intel doubled the vector width to 512-bits, thereby doubling the peak vector unit throughput. ThunderX2 provides dual 128-bit vector units that execute Arm's NEON instruction set. The NEON instructions are much less capable that the AVX512 instructions found in the Skylake processor, making it more challenging to generate optimized instruction sequences.

In Figure 5, we present benchmarked performance of large-scale, threaded DGEMM calls using the double-precision throughput benchmark from the NERSC8/Crossroads benchmark suite [15]. The purpose of this benchmark is to generate large DGEMM routines and repeatedly time these operations to obtain accurate floating-point throughput rates. In order to accurately reflect the maximum performance we utilize vendor optimized math libraries for the DGEMM routines.

The first observation from the benchmark data is that the routines from the MKL are a more tiered result curve, which we typically find is attributed to the matrix size and decomposition over threads. We believe that the causes of these tiers come from optimized blocking routines that have non-linear decision paths causing slightly different optimizations/algorithms to be used when certain matrix sizes are executed. The Arm performance libraries on the ThunderX2 have a much smoother and more linear performance curve, although this is lower than the Intel equivalent. The principal reason for the differing levels in performance is attributed to the different vector widths and clock rates on each of the processors. ThunderX2 has the shortest vector width (0.5 of Haswell and 0.25 of Skylake) but has a larger core count at 28 versus 16 on Haswell and 24 on Skylake. As such, ThunderX2 is able to achieve roughly the same performance (396GF/s) as Haswell (379GF/s). The Skylake provides the highest throughput with a maximum of 784GF/s occurring at 16 threads. ThunderX2 is the most efficient socket at 88.4% of theoretical peak. Skylake provides 72.9% of its theoretical peak when calculated using an AVX512 frequency of 1.4GHz and Haswell provides 66.7% of theoretical peak per socket.

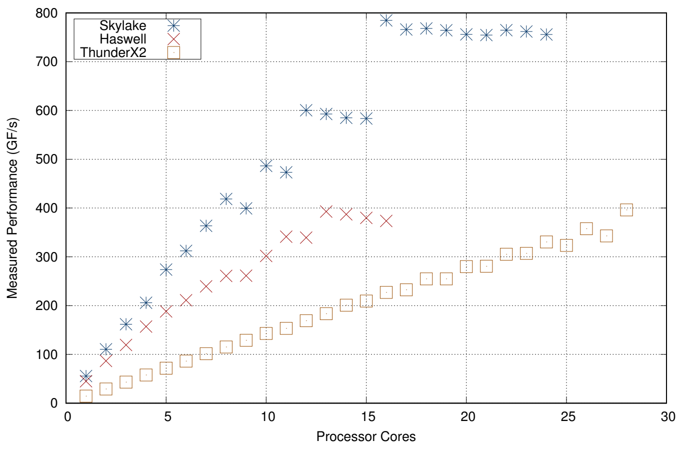


Fig. 5: Benchmarked DGEMM Matrix-Matrix Multiple Performance on Single-Socket Nodes (Intel OpenMP Math Kernel Library used for DGEMM Calls or Skylake and Haswell, Arm Performance Libraries (OpenMP Threaded) used on ThunderX2). Higher is Better. Problem size is approximately 384MB

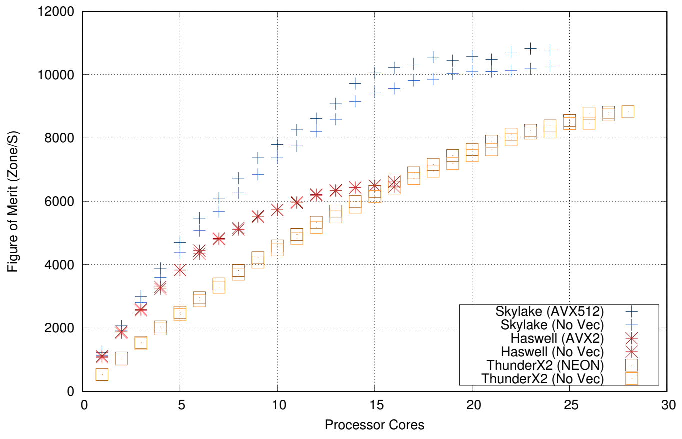


Fig. 6: Benchmarked LULESH Performance using OpenMP multi-threading. Results at for the LULESH Zones-per-second FOM. Higher is Better. Problem size is approximately 570MB

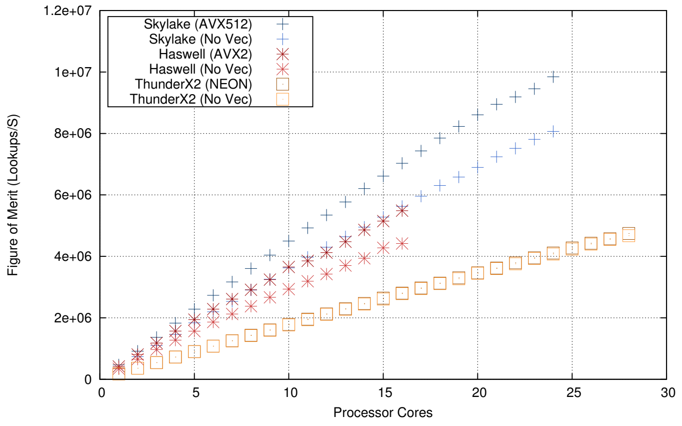


Fig. 7: Benchmarked XSBench Figure-of-Merit. Higher is Better. Problem size is approximately 5.6GB

IV. Benchmarking of Mini-Applications

In this section of the paper we report on three miniapps. Miniapps are, typically, orders of magnitude smaller than a real application but act as proxies for a key application performance issue, providing an easy to understand context to reason about performance issues.

A. *LULESH*

LULESH is one of the most widely used mini-applications developed by the US Department of Energy. The code was originally developed by Lawrence Livermore National Laboratory to represent challenging hydrodynamics algorithms that are performed over unstructured meshes [16][17]. Such algorithms are common in many high-performance computing centers and are particularly prevalent within the NNSA laboratories. In the original LULESH specification, the authors state that such algorithms routinely count in the top ten application codes in terms of CPU hours utilized [16].

The unstructured nature of LULESH presents challenges for the design of memory subsystems, not least because operands are gathered from a fairly limited locale but are done so sparsely. This makes efficient streaming and vectorization of the data operations difficult and places additional pressure on the memory subsystem (typically the L2 caches) to provide operands quickly. However, due to the difficulty in predicting memory accesses, the memory system is likely unable to provide good prefetch support, which can quickly lead to bottlenecks. During compilation, our experience has been that many compilers see the indirection being used for data accesses and will attempt to generate gather/scatter vector instructions if the hardware is able to support them. For the purposes of this paper, Haswell and Skylake support gather instructions but only Skylake provides scatter capabilities in its instruction set. The NEON units of the ThunderX2 provide support for neither, making vectorization cost-models unlikely to select vector code sequences during compilation.

Figure 6 shows the performance of LULESH running on all three processors. We present results for all processors with and without vectorization enabled during compilation. In these results, Skylake provides the highest performance at around 10,800 zones/second with vectorization enabled; Haswell provides a maximum rate of 6,614 zones/second; and ThunderX2 a maximum of 8,119 zones/second. However, the differences with and without vectorization are underwhelming for all of the processors. For both Haswell and ThunderX2, vectorization makes little difference, between 2-3%; even for Skylake the difference is only 8-9%. Digging into the execution metrics, it can quickly be seen that only 3-5% of the run time is actually vectorized, likely due to the issues described above. Even then, the vectorized code is only using a fraction of the available vector width. As such, the performance is limited more by the memory subsystem than the architecture of the vector units, which benefits the ThunderX2. The ThunderX2 provides a 1.22x speedup over Haswell at 28 cores with both providing roughly equal performance at the 16 core mark. Skylake has steep performance growth from 1 to 16 cores and then a much more modest improvement of around 600 zones/second from 16 to 24 cores. The similarity between the shape of the Haswell and ThunderX2 performance curve is likely a reflection of the similar cache sizes and the use of ring topologies in the die.

B. *XSBench*

Monte Carlo transport algorithms are useful in a wide variety of scientific calculations including nuclear reactor design, medical dosimetry and the simulation of nuclear reactions. XSBench is a mini-application [18], developed by the Argonne CESAR Nuclear Reactor simulation codesign project. The code is designed to model one of the most computationally intensive macroscopic neutron cross-section calculations, which can account for up to 85% of a full simulation. In keeping with the philosophy of mini-application design, XSBench removes less relevant instruction paths and focuses on the essential performance-relevant aspects of the algorithm used by production code. The principle operation being benchmarked is the

TABLE I: Benchmarked Kernel Performance of HPCG (reported in GFLOP/s) with and without native vectorization enabled. Higher is Better. All cores used per socket in MPI only execution. Total Problem Size 86.5GB

Kernel	**Skylake (AVX512)**	**Skylake (No Vec)**	**Haswell (AVX2)**	**Haswell (No Vec)**	**ThunderX2 (NEON)**	**ThunderX2 (No Vec)**
DDOT	20.05	30.50	9.87	11.41	20.14	15.96
WAXPBY	16.70	16.88	9.53	9.35	23.51	23.52
SpMV	18.56	17.95	10.22	10.20	34.59	34.70
Multi-Grid	18.29	17.94	10.01	9.89	30.97	31.00
Solve (Total)	18.33	18.04	10.03	9.95	30.66	30.51

look up of cross-section data from nuclide grids – a memory latency intensive set of indirect accesses to a large collection of data configured when the application launches. The typical observed behavior for XSBench is that it places pressure on the memory subsystem using random memory accesses, which are heavily dominated by reads, to search through its structures.

Figure 7 shows benchmarked performance of XSBench on the three selected systems. As in our previous results, we report with and without vectorization enabled during compilation. Skylake delivers the highest number of lookups per second at 9.84M; Haswell is second with 5.48M; and ThunderX2 is the poorest performer at 4.75M. For both the Skylake and Haswell, vectorization improves performance by almost 20%. On ThunderX2 the use of NEON results in an almost negligible performance improvement of 0.05%. Given the intense read-dominated operations from sparse locations in memory, the availability of gather instructions on both Skylake and Haswell provides significant performance gains. The lack of gather instructions in the much simpler design of the NEON vector units is a clear detriment to the TX2 performance, requiring all of its cores to match the non-vectorized performance of Haswell, demonstrating the much greater efficiency of the Intel processors for this type of workload.

C. High Performance Conjugate Gradient (HPCG)

The High Performance Conjugate Gradient benchmark (HPCG) [19][20] is a recently developed benchmark that is intended to augment the familiar High Performance LINPACK (HPL) benchmark that defines the ranking of the world's "Top 500" supercomputers. HPCG was created to address some of the criticism that HPL has lost its relevance as a predictor of application performance on supercomputing installations. As new algorithms have been developed over time, many users have seen a growing pressure on memory subsystem performance rather than raw compute performance, as evidenced by the two previous miniapps. HPCG therefore attempts to place a greater emphasis on memory speed as well as interconnect performance, particularly efficient system-wide reduction operations. As an on-node benchmark, HPCG places significant stress on memory bandwidth.

Results for the reference implementation of HPCG on all three systems are shown in Table I. We present results for HPCG as performance of each of the principal kernels used when running on the full system in MPI-only execution mode. For the sake of brevity, we dispense with scaling curves for presentation to compact our use of space. The reader will note the significantly higher performance of the ThunderX2 system, which we attribute to the much greater provision of memory bandwidth and the higher number of cores available to drive benchmark execution. For most kernels, the ThunderX2 provides more than 1.5x the performance of the Intel Skylake system and 3x the performance of the Intel Haswell system. As with the GUPS and LULESH results, vectorization provides only limited benefit for most kernels. The exception, of course, being the dot-products (DDOT) where vectorization provides almost 33% higher performance on the TX2 and actually negatively impacts the performance on the two Intel systems. The reference implementation does not make use of the restrict qualifier, which leads most compilers unable to determine if vectorization is safe. We note that high performance for all systems can be achieved by using vendor optimized benchmark implementations [21] as we refer the reader to the Top-500 list for these performance results. These are not compared in this study in order to ensure we utilize the same algorithms on all systems.

V. Conclusions

The processor design industry is undergoing a major period of change. Part of this change is a much greater degree of variation in processor and micro-architecture design. This is both a challenge and an opportunity for many large-scale supercomputing sites: the challenge comes from having to more intimately understand complex workloads, even while they themselves are changing. If there is to be a close relationship between the hardware purchased and the scientific software used on them, such a deep knowledge will be critical. For those sites with rapidly changing workloads, this will be particularly difficult. For others, there will be an opportunity to break from contemporary commodity-based designs and seek greater levels of performance through a closer mapping between hardware and software. In addition, the rapid pace of change will open doors in the future into greater levels of customization as well as more aggressive use of packaging techniques to improve performance levels.

During this period, the NNSA program must balance the use of new and novel hardware with an incredibly broad and diverse application portfolio, arguably one the largest collection of computing codes anywhere in the world. Change in a world of millions of lines of source code is without a doubt one of the most significant challenges in its future HPC strategy.

The Vanguard program, of which Astra is the first deployment, is an acknowledgment of the current situation. There is a risk of failing to identify future potential hardware opportunities if novel approaches are not explored. At the same time, the use of next-generation systems is itself a risk if codes cannot adequately utilize the performance they offer, and so careful analysis must be performed to show areas of strength and weakness.

In this paper we have described our initial benchmarking of the Marvell ThunderX2 server-class Arm processor used in Astra. At the time of writing, Astra was in its initial shake-out period, and so we argue that the results shown are a baseline of system performance and these results are likely to improve in the coming months and years. We have presented initial benchmarking of a suite of low-level micro-benchmarks as well as three important DOE mini-applications that represent important HPC kernels. Our benchmarking compares the ThunderX2 to Intel's Haswell and Skylake Xeon processors - leading contenders for HPC processor selection. We find, perhaps unsurprisingly, that no single processor provides the best performance across all codes and that each design has advantages for our complex workloads. Such an observation helps to justify our statements on the complex choices ahead with future hardware purchases. Nonetheless, the results demonstrate that the ThunderX2 processor, and by extension the Arm ecosystem, can deliver exceptional levels of performance for certain codes and are viable candidates for selection in the next-generation of supercomputing deployments.

Acknowledgements

Benchmarking time during the early access phases of Astra was made possible by the Vanguard system prototype program which is sponsored by the NNSA/ASC. We are grateful to all of the administrator staff and the technical support team from HPE for enabling productive access to the Astra system.

Access to the benchmark resources used for this paper are provided by the NNSA/ASC Advanced Architectures Test Bed project, a sequence of hardware deployments by the NNSA/ASC program which provide early access systems to research staff across the NNSA Tri-Lab community. We are grateful to the Test Bed project administration team who configure and maintain the hardware systems.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] J. Laros, K. Alvin, K. Pedretti, and S. Hammond, "DOE NNSA Vanguard Program," Sandia National Laboratories, Albuquerque, NM, United States of America, Tech. Rep., 2017.

[2] J. H. Laros, K. T. Pedretti, S. D. Hammond, M. J. Aguilar, M. L. Curry, R. Grant, R. J. Hoekstra, R. A. Klundt, S. T. Monk, J. B. Ogden *et al.*, "NNSA/ASC FY18 L2 Milestone# 8759 Report: Vanguard Astra and ATSE - an ARM-based Advanced Architecture Prototype System and Software Environment," Sandia National Laboratories, Albuquerque, NM, United States of America, Tech. Rep., 2018.

[3] K. T. Pedretti, J. H. Laros, R. Grant, S. D. Hammond, K. S. Hemmert, D. J. Martinez, J. P. Noe, A. M. Patterson, and H. L. Ward, "Vanguard: Maturing the ARM Software Ecosystem for US DOE Supercomputing," Sandia National Laboratories, Albuquerque, NM, United States of America, Tech. Rep., 2017.

[4] 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*, p. e5110, 2019.

[5] 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 (CUG) Conference*, 2018.

[6] J. Doweck, W.-F. Kao, A. K.-y. Lu, J. Mandelblat, A. Rahatekar, L. Rappoport, E. Rotem, A. Yasin, and A. Yoaz, "Inside 6th-Generation Intel Core: New Microarchitecture Code-Named Skylake," *IEEE Micro*, vol. 37, no. 2, pp. 52–62, 2017.

[7] T. Jain and T. Agrawal, "The Haswell Microarchitecture - 4th Generation Processor," *International Journal of Computer Science and Information Technologies*, vol. 4, no. 3, pp. 477–480, 2013.

[8] P. Hammarlund, A. J. Martinez, A. A. Bajwa, D. L. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar *et al.*, "Haswell: The Fourth-Generation Intel Core Processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, 2014.

[9] S. D. Hammond, C. T. Vaughan, and C. Hughes, "Evaluating the Intel Skylake Xeon Processor for HPC Workloads," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2018, pp. 342–349.

[10] J. Linford. (2019) Compiler Flags Across Architectures: -march, -mtune, and -mcpu. [Online]. Available: https://community.arm.com/developer/tools-software/tools/b/tools-software-ides-blog/posts/compiler-flags-across-architectures-march-mtune-and-mcpu

[11] 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.

[12] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The HPC Challenge (HPCC) Benchmark Suite," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, vol. 213, 2006.

[13] L. W. McVoy, C. Staelin *et al.*, "lmbench: Portable Tools for Performance Analysis," in *USENIX annual technical conference*. San Diego, CA, USA, 1996, pp. 279–294.

[14] J. Dongarra, "Performance of Various Computers Using Standard Linear Equations Software," University of Tennessee, TN, USA, Tech. Rep. CS-89-85, 2019.

[15] (2019) Crossroads Benchmarks, Micro-Benchmarks, & ASC Code Suite. [Online]. Available: https://www.lanl.gov/projects/crossroads/benchmarks-performance-analysis.php

[16] R. Hornung, , J. Keasler, and M. Gokhale, "Hydrodynamics Challenge Problem, Lawrence Livermore National Laboratory," Lawrence Livermore National Laboratory, CA, United States, Tech. Rep. LLNL-TR-490254, 2011.

[17] I. Karlin, J. Keasler, and J. Neely, "LULESH 2.0 Updates and Changes," Lawrence Livermore National Laboratory, Livermore, CA, United States, Tech. Rep. LLNL-TR-641973, August 2013.

[18] J. R. Tramm, A. R. Siegel, T. Islam, and M. Schulz, "XSBench - The Development and Verification of a Performance Abstraction for Monte Carlo Reactor Analysis," in *PHYSOR 2014 - The Role of Reactor Physics toward a Sustainable Future*, Kyoto, Japan, 2014.

[19] J. Dongarra and M. Heroux, "Toward a New Metric for Ranking High Performance Computing Systems," Sandia National Laboratories, NM, USA, Tech. Rep. SAND2013-4744, 2013.

[20] J. Dongarra, P. Luszczek, and M. Heroux, "HPCG Technical Specification," Sandia National Laboratories, NM, USA, Tech. Rep. SAND2013-8752, 2013.

[21] J. Park, M. Smelyanskiy, K. Vaidyanathan, A. Heinecke, D. D. Kalamkar, M. M. A. Patwary, V. Pirogov, P. Dubey, X. Liu, C. Rosales *et al.*, "Optimizations in a High-Performance Conjugate Gradient Benchmark for IA-based Multi-and Many-core Processors," *The International Journal of High Performance Computing Applications*, vol. 30, no. 1, pp. 11–27, 2016.