Evaluating the Intel Skylake Xeon Processor for HPC Workloads

Simon D. Hammond
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, USA
sdhammo@sandia.gov

Courtenay T. Vaughan
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, USA
ctvaugh@sandia.gov

Clay Hughes
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, USA
chughes@sandia.gov

***Abstract*—Despite significant advances in the porting of scientific applications to novel architectures such as compute-optimized graphics processors, many-core processor/accelerators and, even special-purpose function units, the vast majority of scientific calculations are still performed on high-performance, commodity server processors. Even in the cases of applications which have been ported to new architectures, frequent serial sections still require strong server-class processor cores to compute as fast as possible.**

In this paper we report on a set of benchmark studies which evaluate Intel's latest Skylake Xeon server processor. Skylake represents a significant change in the Xeon product line with wider SIMD vector units, a redesigned cache architecture, and, an increased number of memory channels. The wider vector units provide 2x improvement for some compute-intensive applications and the combined memory changes can provide close to 2x the memory bandwidth. We evaluate these new hardware features on several HPC-relevant mini-applications and benchmarks, including, STREAM, LULESH, XSBench, HPCG and SW4Lite. Together, the new hardware functions provide up to 1.8x speedup on HPC benchmark codes when compared with the previous generation Haswell processor core, providing much greater utility to a broader range of HPC applications that rely on this class of compute node.

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

I. Introduction

The Department of Energy's National Nuclear Security Administration (NNSA) is one of the largest users of supercomputing machines in the world. In 2013, the NNSA issued an updated Computing Strategy [1] for it's Advanced Simulation and Computing program that underpins the complex qualification workload for the United States nuclear security program. The Computing Strategy outlined two classes of computing platform: (1) Advanced Technology Systems ("ATS") – systems that blend large-scale capability computing with novel hardware features to target higher application performance, and, (2) Capacity Technology Systems ("CTS") which are cost/performance optimized, mid-size installations (typically around a petaflop) for modest parallel-scale applications.

Traditionally, Capacity Systems within the NNSA have been based on the leading server-class processors of the day with previous systems having used processors from AMD [2] and Intel [3]. The current generation of systems (named Commodity Technology Systems-1 (CTS-1)) provide multiple petaflops of computing power from dual-socket 18-core Intel Broadwell processors [4] interconnected with Intel's OmniPath Generation-1 fabric [5].

The most recent installation of the Advanced Technology System family is found in the NNSA *Trinity* platform [6]–[8] installed at Los Alamos National Laboratory. Trinity is a split machine design, with roughly 9,500 nodes of dual-socket Haswell processors [9] and around 9,800 nodes of single-socket Intel Knights Landing Xeon Phi. Given the considerable computing power provided by both classes of system, the NNSA has significant interest in the benchmarking and evaluation of next-generation server-class hardware designs for future machine procurements.

In this paper, we describe results from a recent benchmark study which evaluates the performance of a range of microbenchmarks, benchmarks and small applications on Intel's latest Skylake server processor [10]. The contributions of this paper are:

- Microbenchmarking of the significant hardware changes found in the Skylake server platform including Level-2 cache bandwidth, additional memory bandwidth available from the increased number of memory channels, and, compute performance from a doubling of the vector width found in the AVX512 instruction set;
- Mini-application benchmarking to evaluate the potential benefit to applications of these hardware changes. We select a range of mini-applications that exhibit behaviors found in larger HPC codes so that we can evaluate each in a more direct context. Specifically, we use a range of mini-applications that are known to experience: bottlenecks from either: (1) memory bandwidth; (2) indirect memory accesses from cache; (3) indirect memory accesses with operands usually retrieved from main memory, and, finally, (4) throughput of computation.
- Evaluation of each mini-application with respect to vectorization and hardware hyperthreading being enabled or disabled in order to analyze the potential configuration options (either at compilation or runtime) that users may wish to select when porting codes to the new Skylake Xeon processor family.

The remainder of this paper is laid out as follows: Section II

discusses the pertinent hardware changes that can be found in the Skylake Xeon processor in greater depth. We describe the benchmark systems used for our experimentation in Section III. Microbenchmarking results for memory bandwidth, cache bandwidth and compute are presented in Section IV. We present mini-application studies in Section V. Finally, we conclude the paper with a summary of our findings in Section VI.

II. Intel Skylake Server Processor

Skylake is the 6th-generation of Intel's Core microarchitecture and is widely seen as a significant redesign of its traditional Xeon server roadmap to provide increased performance and improved performance per unit of power. General availability of the processor came in 2017 and at the time of writing, Skylake represents the most up-to-date server platform available from Intel. The principal changes of the Skylake Xeon processor over its predecessors are:

- **Mesh-based Interconnect** - the core-to-core interconnection fabric used to move coherence and data traffic between caches the memory subsystem is changed from a high-performance bus architecture to a two-dimensional mesh building on Intel's successful demonstration of the technology in its Knights Landing Xeon Phi processor. The use of a two-dimensional mesh has the potential to reduce latency and increase bandwidth as the additional path diversity between each processor core/private cache and other components is increased.
- **Cache Subsystem Redesign** - the L3 cache available per core on the Skylake server Xeon is *decreased* from greater than 2MB per core to around 1.3MB. The silicon area that is saved by the decrease is used, instead, to increase the L2 size from 256kB per core to 1MB (a 4X increase). Thus, each core now has an increased *private* cache size that benefits applications with locality and can help to reduce contention for L3 cache accesses.
- **512-bit Wide Vector Extensions** - the server-grade variant of the Skylake processor offers support for Intel's AVX512 ISA extensions that debuted in the Knights Landing processor [11], [12]. While the increased width of the vector units (doubled from 256-bits in the previous Haswell/Broadwell generation) provides significant improvement for intensively vectorized codes. One of the more significant gains from the use of the AVX512 ISA is the availability of masking registers which permit lanes to be selectively disabled in the case that operands should not be worked on. As such, the compiler is afforded much greater flexibility in selecting how application code is vectorized in the presence of complex control flow or conditionals. The issue of vector instructions in Skylake is also significantly overhauled from previous generations. Each core now has three vector ports. Ports 0 and 1 each have a 256bit wide vector unit that can be selectively fused together to form a larger 512-bit path as required by the instruction stream. A third port (Port 5) in the core allows for simple (typically add, multiply, *etc.* instructions), full 512-bit width instructions to be issued. The effect is that the processor can consume less power when shorter vectors are being processed but still transition into dual 512-bit processing when required.
- **Increased Memory Channels** - the Skylake processor provides an increase in memory channels moving from 4-channels in the previous generation to 6-channels in the latest design. For memory bandwidth bound codes (which are frequently found in the HPC setting), the availability of 50% more memory channels is expected to provide a significant boost in performance.

III. Systems Used for Benchmarking

For benchmarking, we use two systems provided by the NNSA/ASC Advanced Architecture Test Beds project which is run by Sandia National Laboratories for the purposes of providing access to a broad cross-section of future high-performance computing nodes [13].

The Haswell system, *Shepard*, provides dual-socket Xeon E5-2697v3 processors which run at 2.3GHz. Each processor comprises 16-cores with dual-SMT 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 and 256kB L2 cache with associativity configured to 8-way for both. Each socket has a large 40MB L3 cache which is distributed across the ring bus.

Skylake processors are provided in Sandia's *Blake* system. Each node provides dual-socket Xeon Platinum 8160 processors at 2.1GHz. The socket of an 8160 Platinum Skylake contains 24 cores with dual-way SMT 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 maintain 8-way associativity, while the L2 cache is changed to 16-way set associative. A 33MB L3 distributed cache is provided across the 2D processor mesh.

For all experiments reported in this paper we use the Intel 18.1.0 compiler (with GCC 4.9.3 compatibility enabled) with architecture specific flags set for Skylake (`-xCORE-AVX512`) and Haswell (`-xCORE-AVX2`) as appropriate. Where applicable, Intel's 18.1 Math Kernel Library is used for BLAS and LAPACK calls. The configuration is set to use the OpenMP threaded variant of the library and to permit automatic platform optimized dispatch. All codes are compiled with a processor ISA optimized build of OpenMPI 2.1.2. Both systems run RedHat 7.4 Enterprise Linux with the most up to date security patches applied at the time of writing (note, that these include patches to address the recently exposed Spectre and Meltdown security vulnerabilities).

To encourage more accurate analysis, we average each result presented over a minimum of ten runs and select different nodes for each run from our cluster at random to perform

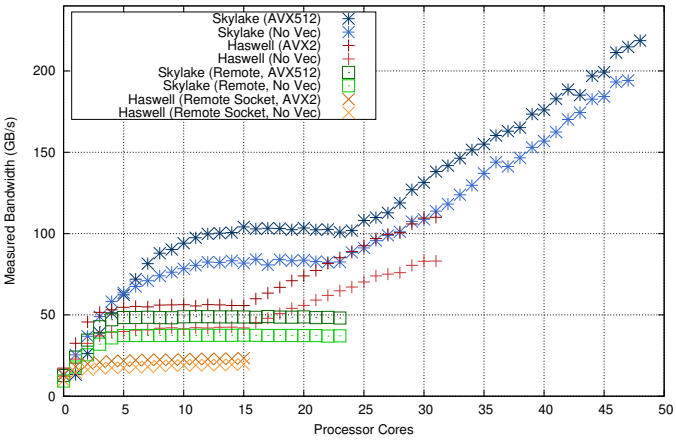

Fig. 1. Benchmarked STREAM Triad Memory Bandwidth on Dual-Socket Haswell and Skylake Nodes. Higher is Better. Problem Size is approx 2.2GB

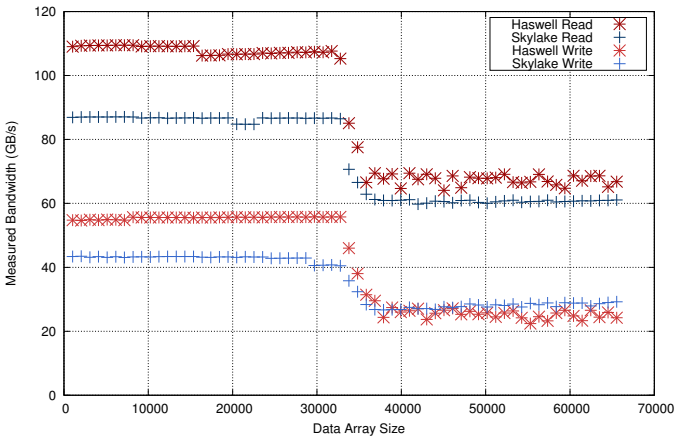

Fig. 2. Benchmarked Cache Bandwidth for a Single Processor Core of Skylake and Haswell (Small Data Array Sizes, Problem Size configured per X-Axis). Higher is Better

experiments. Our anecdotial observation is that we typically see small (1-5%) variations in the results presented for well configured runs where thread and process affinity are fully specified.

IV. Microbenchmarks

A. Memory Bandwidth

As described above, one of the most significant changes that can be found in the server (Xeon) variant of the Skylake processor is the increased number of memory channels, moving from 4 to 6 channels. Figure 1 shows results of executing an OpenMP compiled version of the community accepted STREAM benchmark [14]. In this figure, we make several benchmark studies: (1) execution with and with-out vectorization, denoted by the use of the vector ISA or "No-Vec" in the legend respectively and, (2), for accesses that are between local-socket memory and remote-socket memory, denoted as "Remote" when memory from the remote socket NUMA domain is being benchmarked. For local accesses, the figure shows the familiar plateauing from 1 thread to use of all cores on the first socket (recall this is 16 cores on the Haswell processor and 24 on Skylake), the curve then continues to increase as cores on the second socket are used.

We note that for both Haswell and Skylake processors, the use of vectorization improves memory bandwidth whether accesses are local or remote. For the local case both processors gain by as much as 28GB/s. However, the impact on Haswell (as a percentage of memory bandwidth lost) is larger at between 23 - 30% versus 3 - 20% on Skylake. While the reader might expect the memory bandwidth figures to be approximately identical whether vectorization is or is not used, the general trend in processor design is that more efficient load/store operations (*e.g.* when vector loads are used) continue to be the key determinant of memory subsystem performance. For Skylake this observation continues to hold.

The figure also shows another observation about the difference in socket-to-socket performance between the two

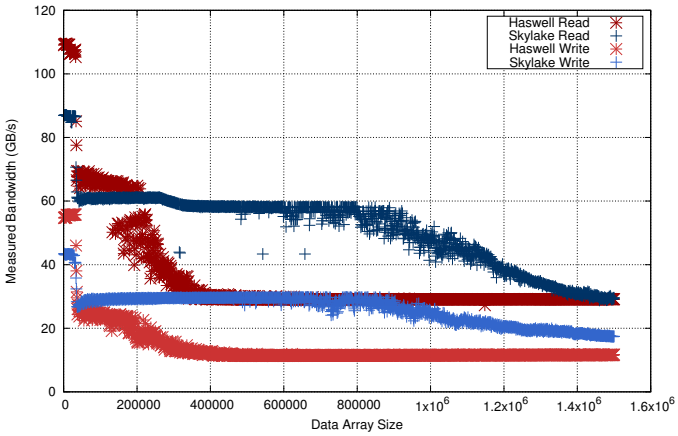

Fig. 3. Benchmarked Cache Bandwidth for a Single Processor Core of Skylake and Haswell (Larger Data Array Sizes, Problem Size configured per X-Axis). Higher is Better

processors. The bandwidth of accessing remote socket memory on Skylake is up to 2.2X higher than in the Haswell design, achieving close to 48.75GB/s versus 23.5GB/s. For future node designs with a single PCIe attached network interface (as in our existing commodity server systems), the socket-to-socket connectivity is an important factor in determining MPI bandwidth into the interconnect.

B. Cache Bandwidth

Intel's significant redesign of the caching structures and policies in its Skylake processor represent a potentially fundamental change for applications that have been written to utilize cache blocking and well known cache sizing. The increase in L2 cache size, which is now four times larger than Haswell, does however, have the potential to provide significant gain in performance for codes with good to moderate locality.

Figures 2 and 3 show the cache bandwidths achieved for a single processor core performing read and write operations using the lmbench memory bandwidth benchmarking utility [15].

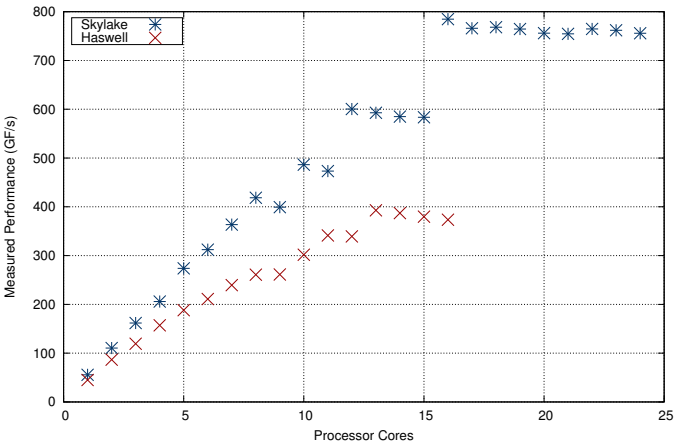

Fig. 4. Benchmarked DGEMM Matrix-Matrix Multiple Performance on Single-Socket Haswell and Skylake Nodes (Intel OpenMP (Threaded) Math Kernel Library used for DGEMM Calls). Higher is Better. Problem size is approximately 384MB

Two figures are provided to improve clarity – the first, Figure 2 provides a zoomed in view of smaller data array sizes, the second, Figure 3 provides a macro view of data array sizes out to 2MB which exceed the L2 private cache size of the Skylake core. The results show that the bandwidth available for the Skylake is lower within the range of the L1 cache. Some of this performance gap is accounted for by the lower clock of the Skylake core (2.1GHz vs. 2.3GHz) however, the results do appear to show that the L1 is slower despite this explanation. The sharp drop at approximately 32kB, represents the change to the L2 cache which provides lower bandwidth. In this case, Skylake initially provides slightly lower read bandwidth and slightly higher write bandwidth. The increased size of the L2 cache on Skylake can also be seen as the reader follows the line towards the right – Skylake continues to provide its performance for a much greater range than its opposition.

One observation present from the graph is that despite our setting the affinity of the benchmark to a single processor core, the results from the Haswell runs show a much larger variation in benchmark performance see by the frequent oscillations in the results. The benchmark results from Skylake are considerably smoother which may be attributed to the increased associativity of the L2.

C. Floating Point Intensive Computation (DGEMM)

The doubling of the vector units in Skylake offers the potential for the newest processor design to provide a significant boost to compute-dense applications. However, simple increasing of processor cores and vector widths comes at the cost of higher power draw and thermal output. Figure 4 shows results taken from repeated runs of a DGEMM (double-precision dense matrix-dense matrix multiplication) benchmark with the number of OpenMP threads being varied. We repeated this benchmark result five times across ten different compute nodes noticing only small (typically less than 2% variation in results).

TABLE I
BENCHMARKED KERNEL PERFORMANCE (REPORTED IN GFLOP/S) FOR SKYLAKE AND HASWELL SERVER PROCESSORS, 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 (NoVec)**	**Haswell (AVX2)**	**Haswell (NoVec)**
DDOT	20.05	30.50	9.87	11.41
WAXPBY	16.70	16.88	9.53	9.35
SpMV	18.56	17.95	10.22	10.20
Multi-Grid	18.29	17.94	10.01	9.89
Solve (Total)	18.33	18.04	10.03	9.95

The first result of note is that the doubled vector width performs as advertised with a doubling of the maximum FLOP/s rate of the Skylake. A benchmarked maximum value of 784GF/s is achieved at 16 cores versus a maximum of 392GF/s using 13 cores of the Haswell. We note however, the shape of the Skylake curve differs from Haswell, in that additional cores provided beyond 16 do not add additional performance to the benchmark result. We attribute these results to thermal throttling of the processor socket when there is extensive of use of power-hungry wide-vector instructions in the pipeline.

V. MINI-APPLICATIONS AND BENCHMARKS

A. High Performance Conjugate Gradient Benchmark (HPCG)

The High Performance Conjugate Gradient benchmark (HPCG) [16], [17] is a recent augmentation to the more familiar High Performance LINPACK (HPL) benchmark suite that defines the ranking of the "Top 500" supercomputers in the world. In part, HPCG was created to address the criticism that HPL has lost relevance in modern supercomputing due to the adoption of newer algorithms and a growing acknowledgement that the floating point calculation capabilities of modern systems had far outpaced other bottlenecks leaving many practical scientific applications unable to harness the computational power reported in HPL runs.

Table I displays HPCG benchmark performance for the most significant/time consuming kernels when run on Skylake and Haswell processors. These runs are performed using MPI only execution. For these configurations we perform benchmarking with and without the native vector ISA enabled during compilation. The results concur with the statement that benchmarked performance is radically different to the HPL numbers that might be expected (as seen in our DGEMM (the predominant kernel for HPL) above).

Skylake delivers 80% higher performance than Haswell for virtually all of the kernels executed. While this can expected of kernels which have significant computation, the kernels of HPCG, in particular the Sparse-matrix vector product (SpMV) are considered heavily memory bandwidth bound and so would be expected to gain by only as much as 50% over the Haswell due to the extra two memory channels present in Skylake. Such a result demonstrates that such kernels are not just benefiting

from additional hardware resources but, are, in fact, benefiting from greater processor efficiency, such as higher aggregate cache throughput and the additional load/store slots which come from having extra processor cores and caches on the Skylake die.

The limited gain from vectorization will come as no surprise to frequent users of the HPCG reference code. The code does not make use of restrict pointers which for almost all compilers leads to the dependency analysis being unable to determine that the code is safe for vector instruction generation. Significantly more performance is possible with this code when more aggressive optimizations are employed [18]. We have presented the reference results to demonstrate what unoptimized codes may expect to achieve on their initial ports to new systems.

B. LULESH Hydrodynamics Mini-Application

LULESH is well exercised mini-application developed by Lawrence Livermore National Laboratory for the purposes of representing challenging hydrodynamics calculations over unstructured meshes [19], [20]. Such codes are common in many high-performance computing centers and are particularly prevalent within the NNSA, commonly ranking amount the top ten application codes in terms of CPU hours spent [19].

The unstructured nature of LULESH, like many of its parent codes, makes the generation of efficient memory accesses particularly challenging as the access pattern cannot be predetermined *a priori* as in the code of a structured algorithm. During compilation, the compiler typically sees some form of indirection and will need to generate operand gather and/or scatter instructions for reading/writing to the mesh respectively. Both Skylake and Haswell instruction sets have support for vectorized gather and scatter instructions but these have historically been known to be significantly less efficient than the equivalent standard packed vector load/store instructions.

While the unstructured nature of the algorithms present in LULESH pose a challenging problem for efficient instruction generation, the algorithm typically experiences a high degree of locality in accesses due to the fact that mesh points being computed on are often close in physical proximity and therefore accesses are close in the data structures used. We typically find high hit rates for the L1 and L2 cache access patterns for such an algorithm which is likely to benefit the redesigned cache structure of the Skylake core.

Figure 5 shows benchmarked LULESH Figure-of-Merit (FOM) measured in the number of zones computed per second. Note, for these benchmark runs we use the LULESH version 2.0.3 (the most current released code at the time of writing), executing in OpenMP threaded mode, as the base implementation has an overly rigid requirement to decompose problems when MPI ranks are employed, making scaling studies all but impossible with this degree of fidelity. The results show that for both processors the highest FOM is achieved using the vectorized variant of the code although the difference between the vectorized and non-vectorized variant is typically between

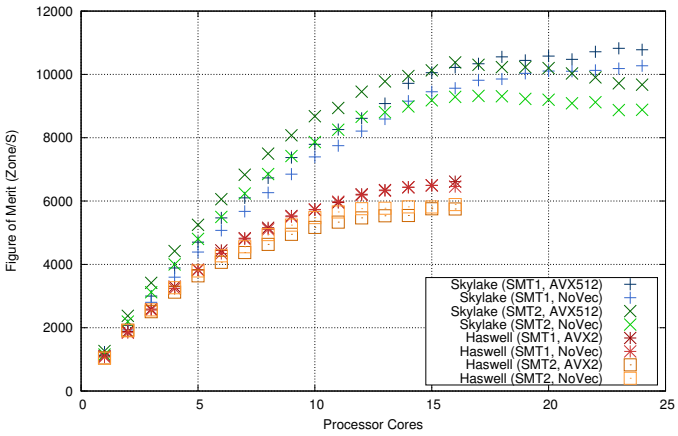

Fig. 5. Benchmarked LULESH Figure-of-Merit on Skylake and Haswell Processors. Higher is Better. Problem size is approximately 570MB

4 and 8%. We note that performance improvement to the basic LULESH implementation is possible (as described in [21]). In this case, the level of vectorization can be improved and show larger gains. The use of SMT/hardware hyper-threading shows a difference between the processors – as small core counts (less than 16 cores) on the Skylake, using two hardware threads per core results in performance improvements of between 1 - 12%. Beyond the use of 16 cores, a single hardware thread provides the highest performance, achieving an FOM that is up to 10% faster. For Haswell, it is always beneficial to use only a single hardware thread. Some of this difference may be explained by thermal management on the Skylake die, when intensive floating point instruction streams are detected, the processor will respond to prevent overheating and thermal damage to the chip. At 16 cores operating with two threads per core, the instruction stream may generate sufficient compute intensity that the on-die management of the processor will scale back the execution performance. Overall, Skylake achieves a performance improvement of between 1.58 - 1.63X over the Haswell processor.

C. XSBench Monte Carlo Macroscopic Cross Section Lookup Benchmark

Monte Carlo transport algorithms are used in a variety of scientific calculations including nuclear reactor design, medical dosimetry and the simulation of nuclear reactions. The XSBench mini-application [22], developed for the DOE's CESAR Nuclear Reactor simulation codesign project, is designed to model one of the most computationally intensive macroscopic neutron cross-section calculations. Such calculations can account for up to 85% of a complex particle transport simulation. In keeping with the philosophy of mini-application design, XSBench removes some of the less relevant instruction paths and focuses on the essential performance-relevant aspects of the algorithm used by the CESAR center. The principle operation being benchmarked is the look up of cross-section data from nuclide grids – a memory latency intensive set of

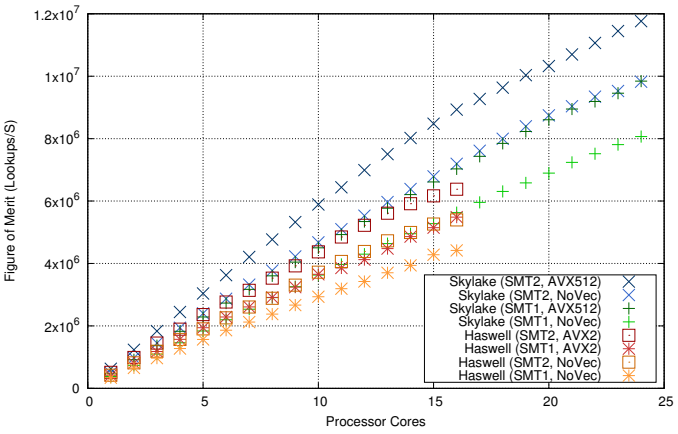

Fig. 6. Benchmarked XSBench Figure-of-Merit on Skylake and Haswell Processors. Higher is Better. Problem size is approximately 5.6GB

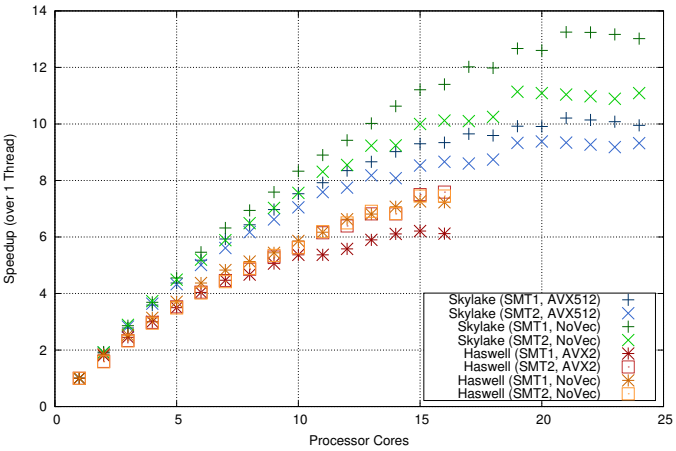

Fig. 7. Benchmarked SWLite Speedup over 1 Thread on Skylake and Haswell Processors. Higher is Better. Problem size is approximately 11GB

indirect accesses to a large collection of data configured at the application launch.

The random access, poor locality nature of the lookups in XSBench (a significant number of which require accesses reaching memory rather than being found in cache), means the mini-application tends to experience bottlenecks on memory subsystem latency. Figure 6 shows benchmarked lookups per second from XSBench on the Skylake and Haswell processors. Unlike our previous codes which benefit from greater locality and, as such, are bottlenecked less by cache and memory miss latencies, XSBench sees much greater performance gain from the use of latency-hiding hardware threading, with approximately 20% more lookups being able to be performed per second when SMT-2 mode is used. The effect is similar on Haswell although the gain from the use of hardware threads is lower with the performance increase limited to 16%.

The benefits of vectorization are also more pronounced in XSBench than our previous benchmarking. The code sees gains of up to 22% on Skylake and up to 24% on Haswell. We believe that the hardware gather instructions which are used for indirect lookups play a greater role here and this result may expose their optimization for indirection of operands in memory and not those who may be fetched from cache. This would indicate that unlike LULESH, where the operands are frequently accessed from cache due to the locality of the data, XSBench benefits from a pipeline which can retrieve sparse operands from memory with less overhead.

Overall, on a socket-to-socket comparison, the Skylake processor achieves a performance advantage of around 1.85X over Haswell for XSBench.

D. SW4Lite Numerical Intensive Geodynamics Mini-Application

SW4 is an experimental research code which provides complex 3-dimensional modeling of seismic activities [23]. The application has complex heterogeneous material models and free surace boundary condition modeling on a realistic topography. The code has gained a reputation for being a numerically intensive test of modern processors because of the optimized kernels used to implement its internal models as well as its calls to optimized BLAS and LAPACK routines. The size and complexity of the full SW4 code makes rapid exploration sufficiently difficult that a smaller representative mini-application, SW4Lite, has been implemented to capture many of the most important computational characteristics required of future hardware.

Figure 7 shows the thread scaling of the SW4Lite kernel when run on the Skylake and Haswell processors. For both processors the effect of vectorization is significant – SW4Lite is between 39% and 45% faster to execute on Skylake and between 45% and 47% on Haswell reflecting the compute intensive nature of the calculation and the higher level of optimization which has been applied to the code by the authors. For both processors the use of SMT-2 mode does not provide a benefit which correlates with our anecdotal evidence that the use of more threads in compute intensive codes does not benefit since the floating point units are already saturated and additional threads only serve to add overhead into the application. For Skylake, the use of hardware threads causes a slowdown of at least 11%, for Haswell the effect is larger with a slowdown of approximately 30%. Overall, Skylake is almost 83% faster to execute the SW4Lite kernel over Haswell when comparing the fast execution times on both processors.

VI. Conclusions

The Skylake Xeon server processor represents the very latest in Intel's offering for high-performance computing nodes. The redesigned core has a raft of new features and refinements that are intended to improve application performance including a completely redesigned cache subsystem, a 50% increase in the number of memory channels per socket providing a large boost in memory bandwidth, wider, and more capable vector processing units to increase computational throughput and a new two-dimensional network-on-chip to reduce core-to-core and core-to-memory latencies.

In this paper we have evaluated these new features with several micro-benchmarks and mini-applications, which, for the majority of the studies have been run with and without vectorization and hardware hyper-threading in use. The benchmarking coverage was designed specifically to provide insight into what larger HPC applications might expect to see once ported to a new Skylake platform based on the level of optimization and environment configuration may be employed.

Our results show that the Skylake server provides a significant performance improvement over the previous generation Haswell processor core. For intensive compute throughput, our DGEMM benchmark showed a 2X gain in performance which is largely expected due to the use of 512-bit wide vector units. For the SW4Lite kernel, which is itself compute intensive, the results are lower but still impressive with a reduction of up to 83% in benchmarked compute time.

Memory bandwidth intensive codes are expected to see significant gains on Skylake with the additional two memory channels per socket. The STREAM micro-benchmark used in this paper demonstrated a dual-socket node-to-node gain of nearly 2X to 223.8GB/s versus 112.6GB/s on the Haswell. For the HPCG mini-application, which has been shown to be heavily dependent on memory bandwidth, Skylake offers a single-socket to single-socket advantage of 80% in kernel performance for all four of the principal solve kernels studied.

The overhauled cache design – arguably one of the most aggressive in Intel's recent designs – found in Skylake has the potential to provide large performance gains for code with small to moderate locality. The increased L2 cache size will permit codes with slightly weaker locality to gain from the larger capacity when compared to previous generation processor cores. The microbenchmarked cache bandwidth demonstrated lower read and write bandwidths when compared to Haswell but, as expected, the cache bandwidths were available for much larger data arrays because of the increased capacity. This represents a trade off in the design of the processor – a larger but slightly slower cache versus a smaller but faster one. The LULESH mini-application was used to investigate the impact on application performance in this setting. The unstructured nature of the kernels found in LULESH results in the compiler generated indirection-based memory accesses. Due to the physical locality of points in the mesh, the code has been shown to benefit from both L1 and L2 caches. For LULESH, Skylake offers a gain of around 1.6X over Haswell on a socket-to-socket basis.

Finally, to investigate whether the cache subsystem changes and the use of a two-dimensional network-on-chip would affect applications that are bottlenecked on sparse memory accesses that actually reach the memory controllers (unlike LULESH which typically are resolved in cache), we benchmarked the Skylake and Haswell processors using the XSBench mini-application. XSBench uses pointer indirection to look up values from larger data structures replicating the behavior of larger, complex Monte Carlo algorithms. For XSBench, the use of hardware threading provided benefit on both processors being studied which we attribute to the latency-hiding benefits from SMT. Skylake again provided approximately 1.8X improvement over Haswell in a socket-to-socket comparison.

Overall, the results show that Skylake is a compelling processor with a number of benchmark speedups measured at around 1.8X over the previous generation Haswell core. Such gains are hard to achieve and the list of changes described in our Skylake overview show the extensive work which the designers at Intel have undertaken to deliver improved performance while requiring only limited increases in processor power. The results described in this paper show the level of performance that future algorithms and kernels might expect in the eventuality that Skylake or a follow-on processor were to be selected for a future hardware deployment.

ACKNOWLEDGEMENTS

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.

The document release number of this paper is: SAND2018-5244C.

REFERENCES

[1] J. Ang, P. Henning, T. Hoang, and J. Neely, "Advanced Simulation and Computing Platform Strategy," Office of Advanced Simulation and Computing, NA-114, Sandia National Laboratories, NM, United States, Tech. Rep. SAND 2013-3951P, 2013.

[2] M. Rajan, D. Doerfler, C. T. Vaughan, M. Epperson, and J. Ogden, "Application Performance on the Tri-Lab Linux Capacity Cluster-TLCC," in *Technology Integration Advancements in Distributed Systems and Computing*. IGI Global, 2012, pp. 144–160.

[3] M. Rajan, D. Doerfler, P. T. Lin, S. D. Hammond, R. F. Barrett, and C. T. Vaughan, "Unprecedented Scalability and Performance of the new NNSA Tri-lab Linux Capacity Cluster 2," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*. IEEE, 2012, pp. 417–425.

[4] A. Nalamalpu, N. Kurd, A. Deval, C. Mozak, J. Douglas, A. Khanna, F. Paillet, G. Schrom, and B. Phelps, "Broadwell: A family of IA 14nm processors," in *VLSI Circuits (VLSI Circuits), 2015 Symposium on*. IEEE, 2015, pp. C314–C315.

[5] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood, and R. C. Zak, "Intel Omni-Path Architecture: Enabling Scalable, High Performance Fabrics," in *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*. IEEE, 2015, pp. 1–9.

[6] C. T. Vaughan, D. Dinge, P. Lin, S. D. Hammond, J. Cook, C. R. Trott, A. M. Agelastos, D. M. Pase, R. E. Benner, M. Rajan *et al.*, "Early Experiences with Trinity-The First Advanced Technology Platform for the ASC Program," Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), Tech. Rep. SAND2016-3605C, 2016.

[7] J. H. Laros and D. W. Doerfler, "Trinity Advanced Technology System Overview," Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), Tech. Rep. SAND2015-0496C, 2015.

[8] D. W. Doerfler, "Trinity: Next-Generation Supercomputer for the ASC Program," Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), Tech. Rep. SAND2014-2739C, 2014.

[9] 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.

[10] 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.

[11] A. Sodani, "Knights Landing (KNL): 2nd Generation Intel Xeon Phi Processor," in *Hot Chips 27 Symposium (HCS), 2015 IEEE*. IEEE, 2015, pp. 1–24.

[12] A. Sodani, R. Gramunt, J. Corbal, H.-S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y.-C. Liu, "Knights Landing: Second-Generation Intel Xeon Phi Product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, 2016.

[13] J. H. Laros, J. A. Ang, S. D. Hammond, and J. M. Brandt, "Sandia's Advanced Architecture Test Beds," Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States); Sandia National Laboratories, Livermore, CA, Tech. Rep. SAND2015-4764C, 2015.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] R. Hornung, H. Jones, J. Keasler, R. Neely, O. Pearce, S. Hammond, C. Trott, P. Lin, C. Vaughan, J. Cook *et al.*, "ASC Tri-lab Co-design Level 2 Milestone Report 2015," Lawrence Livermore National Laboratory, Livermore, CA (United States), Tech. Rep. LLNL-TR-677453, 2015.

[22] 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.

[23] B. Sjögreen and N. A. Petersson, "A Fourth Order Accurate Finite Difference Scheme for the Elastic Wave Equation in Second Order Formulation," *Journal of Scientific Computing*, vol. 52, no. 1, pp. 17–48, Jul 2012. [Online]. Available: https://doi.org/10.1007/s10915-011-9531-1