Mainstream vs. Emerging HPC: Metrics, Trade-offs and Lessons Learned

1st Milan Radulovic
Barcelona Supercomputing Center (BSC)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
milan.radulovic@bsc.es

2nd Kazi Asifuzzaman
BSC & UPC
Barcelona, Spain
kazi.asifuzzaman@bsc.es

3rd Darko Zivanovic
BSC & UPC
Barcelona, Spain
darko.zivanovic@bsc.es

4th Nikola Rajovic
BSC
Barcelona, Spain
nikola.rajovic@bsc.es

5th Guillaume Colin de Verdière
CEA, DAM, DIF
Arpajon, France
guillaume.colin-de-verdiere@cea.fr

6th Dirk Pleiter
Jülich Supercomputer Centre
Forschungszentrum Jülich
Jülich, Germany
d.pleiter@fz-juelich.de

7th Manolis Marazakis
Foundation For Research &
Technology - Hellas (FORTH)
Heraklion, Greece
maraz@ics.forth.gr

8th Nikolaos Kallimanis
FORTH
Heraklion, Greece
nkallima@ics.forth.gr

9th Paul Carpenter
BSC
Barcelona, Spain
paul.carpenter@bsc.es

10th Petar Radojković
BSC
Barcelona, Spain
petar.radojkovic@bsc.es

11th Eduard Ayguadé
BSC & UPC
Barcelona, Spain
eduard@ac.upc.edu

***Abstract*—Various servers with different characteristics and architectures are hitting the market, and their evaluation and comparison in terms of HPC features is complex and multi-dimensional. In this paper, we share our experience of evaluating a diverse set of HPC systems, consisting of three mainstream and five emerging architectures. We evaluate the performance and power efficiency using prominent HPC benchmarks, High-Performance Linpack (HPL) and High Performance Conjugate Gradients (HPCG), and expand our analysis using publicly available specialized kernel benchmarks, targeting specific system components. In addition to a large body of quantitative results, we emphasize six usually overlooked aspects of the HPC platforms evaluation, and share our conclusions and lessons learned. Overall, we believe that this paper will improve the evaluation and comparison of HPC platforms, making a first step towards a more reliable and uniform methodology.**

***Index Terms*—HPC, emerging and mainstream architectures, energy efficiency, weak vs. strong cores, memory latency and bandwidth, Byte/FLOP ratio**

I. Introduction

Each year we see a greater variety of HPC systems in the market. In addition to mainstream x86 architectures, emerging architectures based on POWER, ARM, SPARC and others are steadily appearing and catching attention [1]. Instead of hosting a single type of platform, supercomputing centers already provide a diverse set of systems. Making the right choice of architecture is critical, but evaluating and comparing HPC systems is hard.

Our study evaluates and compares three generations of mainstream x86 architectures: Intel Nehalem, Sandy Bridge and Haswell, and five emerging architectures: Intel Knights Landing (KNL), IBM Power8, Cavium ThunderX, and Applied Micro (APM) X-Gene 1 and X-Gene 2. In addition to presenting a large body of quantitative results, we emphasize six usually overlooked aspects of HPC platform evaluation, and share our conclusions and lessons learnt.

First, we show that a platform's **performance and energy-efficiency depend significantly (*n*-fold) on the characteristics of the target applications**. For example, the ThunderX platform has 50% better energy efficiency than Haswell when running memory-bound HPCG, but Haswell shows 3.6× better efficiency for the compute-intensive HPL. We strongly advocate that any comparison between the platforms should start with the performance and energy-efficiency of HPL and HPCG as the boundaries of the compute-intensive and memory-intensive HPC applications. However, none of the previous studies [2]–[4] that compare emerging and mainstream HPC platforms includes these results.

Second, our performance-per-watt results show that emerging **ARM-based platforms generally show much lower energy efficiency** than the KNL and mainstream Intel platforms. This finding opposes the conclusions of previous studies [2], [4] that report or estimate high energy-efficiency of the ARM platforms (reasons for this are discussed in detail in Section IV).

Third, it is important to understand whether HPC systems should be based on strong or weak CPU cores. **ARM-based platforms typically use weak cores** that are slower than x86, KNL and POWER8 servers in terms of floating-point performance and memory bandwidth. The ARM approach can still be used to build massively parallel systems that reach the target performance by using a larger number of cores. However this scale-out approach may cause significant performance and energy-consumption penalties [5], and it is important to confirm that it is indeed a viable alternative.

Forth, we detect a significant range in the **main memory access latency**, with a factor of three difference between the fastest and slowest platforms under study ($90\,\mathrm{ns}$–$285\,\mathrm{ns}$). Since memory latency has a direct performance impact for many applications [6], it should be minimized in HPC servers, and any increment above about $100\,\mathrm{ns}$ should be analysed and justified.

Fifth, we also analyse the **Byte/FLOP ratio** and detect a huge difference of up to **21×** among the platforms under study. The Byte/FLOP ratio is one of the most important design decisions, and we hope that our results will resurface a discussion on its desired range.

Sixth, our measurements show significant, **up to 70% differences between theoretical and sustained performance**, especially for emerging platforms. Therefore, we note the importance of measuring performance using specialized kernel benchmarks rather than relying on theoretical numbers from datasheets, even for the first order evaluation of the system. Also, hopefully these results will motivate further development of the emerging HPC compilers and scientific libraries.

In summary, given the substantial investment of time and money to deploy an HPC system, it is important to carefully evaluate and compare the available mainstream and emerging architectures. The conclusions of such an analysis depend significantly on the applied methodology, and the previous studies report the findings based on different experimental set-up, statistics of interest and benchmarks. Overall, we believe that this paper will improve the evaluation and comparison of HPC platforms, making first steps towards more reliable and uniform methodology.

II. Experimental environment

In this section, we explain efforts in evaluation of HPC systems, together with workloads and experimental platforms we used in our analysis.

A. HPC benchmarks

HPC benchmarks are important for bounding the sustainable performance of different components in a system.

High-performance Linpack (HPL) [7] has been the only metric for ranking HPC systems for a long time. It measures the sustained floating-point rate (GFLOPs/s) for solving a dense system of linear equations using double-precision floating-point arithmetic.

High-performance Conjugate Gradients (HPCG) [8] is based on an iterative sparse-matrix conjugate gradient kernel. The performance of HPCG largely depends on the available memory bandwidth [9].

HPC Challenge (HPCC) [10] is a benchmark suite that is designed to approximately bound computations of high and low spatial and temporal locality. We used DGEMM and STREAM benchmarks from HPCC suite. **DGEMM** is a floating-point intensive benchmark that represents the corresponding Level 3 Basic Linear Algebra Subprograms (BLAS) routine. The benchmark calculates the product of dense double precision matrices: $C \leftarrow \alpha A \times B + \beta$. It is used for measuring the sustainable FLOP performance, at the *per-core* or *per-node* level. The **STREAM** benchmark performs operations on arrays that are several times larger than the last level cache, effectively measuring the system's sustained memory bandwidth [11]. It comprises four kernels: Copy, Add, Scale and Triad. In our analysis, we report the results of the Triad operation, since it is the most similar to kernels used in HPC applications.

LMbench suite [12] contains several benchmarks which measure performance of different hardware and software components in a system. We used the memory read latency benchmark in order to measure access latencies of different levels in memory hierarchy. The benchmark reads the input dataset in a random order to mitigate the impact of the data prefetching. By varying the input load size, we measure access latency to all memory hierarchy levels.[1]

B. HPC platforms

For the last decade, the dominant HPC architectures have been Intel architectures such as Nehalem, Sandy Bridge and Haswell. Apart from these, many-core systems, of which Intel's KNL is an example, are becoming popular, while other vendors are also emerging architectures that are promising for HPC. For our study, we included mainstream HPC architectures which have been predominantly used in HPC systems so far, as well as emerging architectures which have been recently introduced to the market and are set to be used in future HPC systems. The architectures under study with their most important features and used system software are summarized in Table I.

Comparing different HPC architectures under study is challenging. Architectures developed by different vendors essentially have different Instruction Set Architectures (ISAs) and therefore different system software such as compilers and scientific libraries. For each platform, we identified system software that provided the best performance. It has been used as is, and has not been tuned for each of the platforms. Hence, our conclusions should not be understood as a comparison between different hardware (CPUs and memory only), but a comparison of the platforms (systems) that also include the corresponding system software.

To our knowledge, there are no studies which analyze this many platforms, three mainstream and five emerging ones. Unlike some of the previous studies [2], [13], [14] which performed first-order evaluation of the emerging platforms by using their developer kits, all the platforms under study are fully-fledged production servers that could be used in an HPC system. We argue that it is important to compare fully-fledged servers since their performance features and power consumption differ significantly from the corresponding developers kits.

C. Power measurements

For all platforms under study we measure the power consumption at the server level, which may comprise multiple

[1] The measured latency comprises not only the latency of the hardware components (caches, memory controller, main memory), but also the latency of the system software such as virtual-to-physical memory translation.

TABLE I: Summary of the most important features and used system software of the platforms under study

	Mainstream architectures			Emerging architectures				
Platforms	**Nehalem X5560**	**Sandy Bridge E5-2670**	**Haswell E5-2698v3**	**Knights Landing Xeon Phi 7250**	**Power8**	**ThunderX**	**X-Gene 2**	**X-Gene 1**
Manufacturer	Intel	Intel	Intel	Intel	IBM	Cavium	APM	APM
Architecture	Nehalem	Sandy Bridge	Haswell	2nd gen. MIC	POWER8	ARMv8-A	ARMv8-A	ARMv8-A
Released	2009	2012	2014	2016	2014	2014	2015	2013
Sockets	2	2	2	1	2	2	1	1
Cores per Socket	4	8	16	68	10	48	8	8
CPU freq. [GHz]	2.8	2.6	2.3	1.4	3.49	1.8	2.4	2.4
Out-of-order	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
DP Flops per cycle, per core	4	8	16	32	8	2	2	2
L1i	32kB	32kB	32kB	32kB	32kB	48kB	32kB	32kB
L1d	32kB	32kB	32kB	32kB	64kB	32kB	32kB	32kB
L2	256kB	512kB	256kB	1MB	512kB	16MB	256kB	256kB
L3	8MB	20MB	40MB	/	80MB	/	8MB	8MB
Memory conf. per socket	3 chann. DDR3-1333	4 chann. DDR3-1600	4 chann. DDR4-2133	8 chann. MCDRAM[a] + 6 chann. DDR4-2400	4 chann. DMI 28.8GBps	4 chann. DDR3-1600	4 chann. DDR3-1600	4 chann. DDR3-1600
Memory capacity per node	24GB	32GB	128GB	16GB MCDRAM + 192GB DDR4	256GB	128GB	128GB	64GB
Operating system (OS)	RHEL[b]	SUSE Linux Enterprise Server 11	RHEL	CentOS 7.2	Ubuntu 16.04	Ubuntu 14.04	Ubuntu 14.04	RHEL
Compiler	Intel compiler 17.0	Intel compiler 13.0.1	Intel compiler 17.0	Intel compiler 17.0	IBM XL 13.01	GCC 6.1.0	GCC 6.1.0	GCC 4.8.5
MPI implementation	Intel MPI	Intel MPI	Intel MPI	Intel MPI	Open MPI	Open MPI	Open MPI	Open MPI
Scientific libraries	Intel MKL	Intel MKL	Intel MKL	Intel MKL	ESSL[c]	ARM Performance Libraries	ARM Performance Libraries	OpenBLAS

[a] KNL system has been set to *flat mode*, therefore both memories, MCDRAM and DDR4, are exposed as separate NUMA nodes, and the user can choose in which memory the workload executes.
[b] RHEL stands for Red Hat Enterprise Linux distribution.
[c] ESSL stands for Engineering and Scientific Subroutine Library.

sockets, as detailed in Table I. The power measurements are performed with on-board or external power meters, that account for the overall server consumption including CPUs, memory, power supply, and so on. We also used power measurements to calculate the power efficiency of the platforms under study for HPL and HPCG benchmarks.

III. Results

In this section, we present results from the evaluation of multiple HPC architectures with different benchmarks. We start with the most prominent HPC benchmarks, HPL and HPCG, and later expand the analysis to include other benchmarks, which together give a more complete picture of a system's performance.

A. HPL and HPCG benchmarks

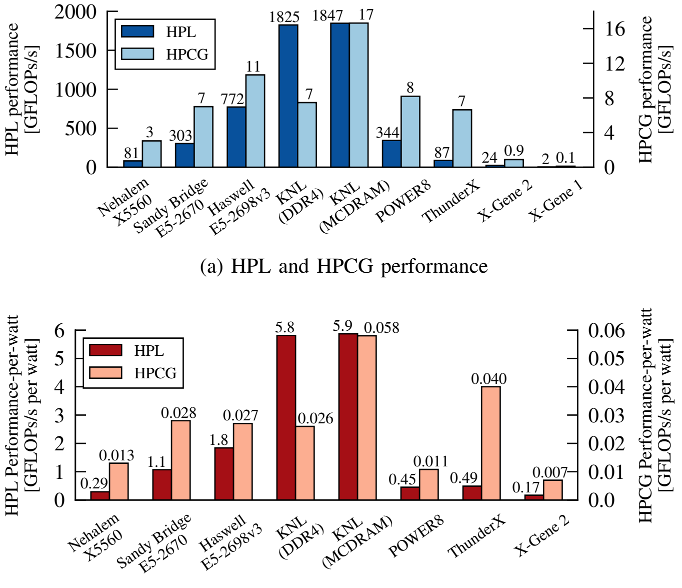
This section gives insights on the performance and power efficiency of platforms under study, while executing the HPL and HPCG benchmarks.[2] Figures 1a and 1b show the HPL and HPCG performance and performance-per-watt of platforms under study. These measurements are obtained with multi-threaded version of the benchmarks that use all the available physical cores.

Figure 1a lists the performance measurements. The KNL platform shows by far the best performance for HPL, followed by the Haswell and POWER8 servers, which reach 42% and 18% of the KNL's HPL performance. Emerging ARM platforms show significantly lower performance. ThunderX, X-Gene 2 and X-Gene 1 deliver 5%, 1.3% and 0.1% of the KNL HPL scores, respectively. The results also show the notable improvement in the HPL performance over the various generations of mainstream platforms, from Nehalem to Sandy Bridge and Haswell.

The HPCG results show a slightly different trend. KNL using MCDRAM is still the highest-ranked platform, followed by Haswell, POWER8, Sandy Bridge and ThunderX. However, the gap in performance between these platforms is much lower for HPCG than for HPL.

Figure 1b lists the performance-per-watt results for the studied platforms. These measurements are important because one of the main drivers for research on the feasibility of the HPC on emerging ARM-based platforms is improved energy efficiency over mainstream HPC servers. Mainstream platforms show increasing power efficiency for HPL, with

[2] For X-Gene 1 platform, we could not obtain power measurements.



(a) HPL and HPCG performance

(b) HPL and HPCG performance-per-watt

Fig. 1: Except for KNL, emerging platforms are generally behind the mainstream ones.

KNL as the best. POWER8, ThunderX and X-Gene 2 show significantly lower energy efficiencies, at just 7.6%, 8.3% and 2.9% of the KNL's performance-per-watt. HPCG power efficiency increases from Nehalem to Sandy Bridge and then stagnates for Haswell and KNL using DDR4. On the other hand, KNL using MCDRAM achieves the highest power efficiency. Emerging platforms show a much lower power efficiency, except for the ThunderX platform, which is the second best, only 31% lower than KNL using MCDRAM.

The results show that, regarding power efficiency, it is very important to identify the target application. When targeting floating-point intensive applications (such as HPL), using low-power/low-performance cores seems not to be the best approach for overall energy efficiency. However, when targeting applications with lower processing requirements (and higher stress to other resources such as main memory) the ThunderX approach may deliver the energy efficiency, which significantly exceeds the x86 and POWER8 platforms.

B. Sustainable FLOP performance and memory bandwidth

In addition to the HPL and HPCG measurements, we also plot the raw measurements, sustained FLOPS and memory bandwidths of the studied platforms, at two levels of granularity, *per-node* and *per-core*. For node-level measurements, we execute the multi-threaded implementations of DGEMM and STREAM, using all the cores in the system. For core-level measurements, we execute the DGEMM benchmark on a single CPU core, and compute the per-core ratio of the sustainable bandwidth as the node STREAM result divided by the number of cores per node. This would correspond to a fair share of the sustainable memory bandwidth delivered to each application process assuming that the system is fully utilized.

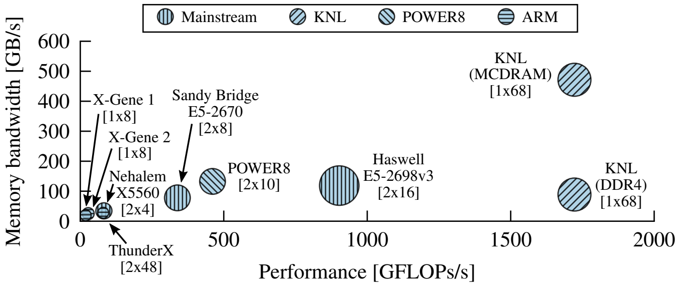
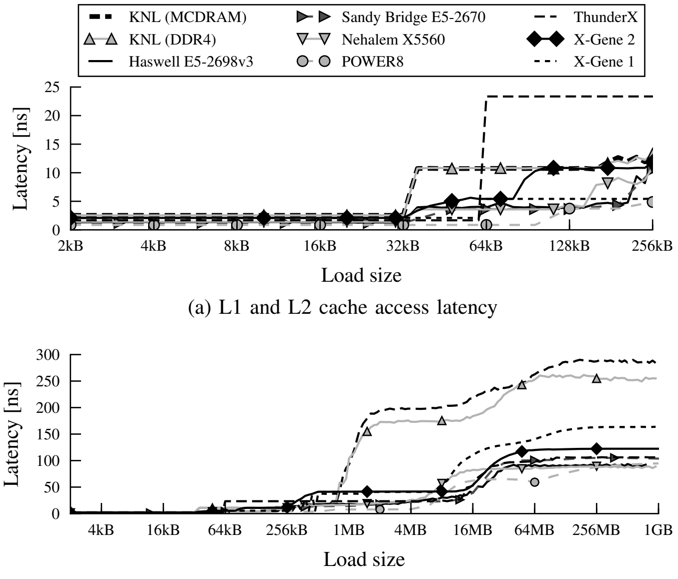


Fig. 2: Sustainable FLOPS and memory bandwidth: The position of each platform in the chart shows the per-node sustained GFLOPs/s (x-axis) and memory bandwidth (y-axis). The marker size is proportional to the per-core performance.

The results are summarized in Figure 2. The position of each platform in the chart shows the per-node sustained GFLOPs/s (x-axis) and memory bandwidth (y-axis), while the size of the marker is proportional to the per-core performance. We also use different hatch patterns to distinguish between mainstream, ARM-based, POWER8 and KNL platforms. The results correlate with the HPL and HPCG analysis. Again, we see that the relative difference in the FLOPs performance (x-axis) is much higher than the memory bandwidth differences (y-axis). The Figure 2 also indicates that POWER8, Haswell and KNL platforms with DDRx memory interfaces reached plateau in terms of the sustained memory bandwidth, while KNL with MCDRAM provided a huge leap forward. Finally, the figure clearly shows that the emerging ARM platforms have lower performance on the server level (they are clustered in the left bottom corner of the chart) and their per-core capabilities are n-fold below the capabilities of the x86, POWER8 and KNL platforms (the size of the marker is much smaller).

This result launches an important discussion: should one use powerful cores similar to Haswell, POWER8 or KNL as the building blocks for large-scale HPC systems or use weaker cores, such as X-Gene or ThunderX? Reaching the target performance with weaker cores would require building of massively parallel systems, using higher number of cores, sockets and servers. This *scale-out* approach could indeed show good HPL and HPCG performance since these benchmarks have very good scalability and their performance is close to being proportional to the total system FLOPS or system memory bandwidth, respectively [15]. However, scale-out approach with production HPC applications may cause significant performance penalties. A recent study from Zivanovic et al. [5] analyzes the scale-out overhead of production HPC applications [16] running on a large HPC system [17]. Their results show that, even if the applications are substantially optimized, increasing the number of application processes to solve a fixed problem leads to significant increase in both energy and node-hours.

We therefore believe that before claiming that HPC systems based on weak cores are a viable alternative, it is essential to perform a profound analysis of the trade-offs between horizontal and vertical scaling in HPC.



(a) L1 and L2 cache access latency

(b) L1, L2, L3 and main memory access latency

Fig. 3: Cache and main memory latency can vary significantly among the studied platforms. KNL memory access latency exceeds 3× the latency on other platforms.

C. Caches and main memory access latency

For decades, the memory system has imposed a fundamental limitation on system performance. This is recognized by the HPC community: HPL scores are frequently complemented by HPCG performance; sustained memory bandwidth is one of the main HPC performance metrics [10] [18], and high-bandwidth memory solutions caused a lot of interest by the HPC users. However, although the community invests significant effort to understand the memory bandwidth, the cache and main memory latencies are usually overlooked. This is surprising because the memory latency has a direct performance impact, and the memory wall itself was defined in terms of latency, not bandwidth [6].

In this section, we compare the access time of the caches and main memory for platforms under study. The results are plotted in Figure 3. The x-axis of the figures shows the input dataset size. In Figure 3a the load size ranges from 2 KB to 256 KB, which focuses on the L1 and L2 caches. In Figure 3b dataset size reaches up to 1 GB, covering all levels of caches and main memory. Even for the L1 and L2 caches we detect a significant difference in the latencies. At the L1 cache (2–32 KB load) the latency varies from 1.25 ns (Sandy Bridge, Haswell, POWER8) to 2.5 ns for KNL. In the L2 cache (128 KB load), the difference is even more significant, from 3.6 ns (Nehalem, Sandy Bridge, Haswell, POWER8) to 23 ns (ThunderX). The main memory latency (256 MB load) ranges from 90 ns (Nehalem, Haswell and POWER8) and 105 ns (Sandy Bridge, ThunderX) up to 250 ns and 285 ns for KNL using DDR4 and MCDRAM, respectively.

Overall, our measurements show that the cache and main memory latencies can vary significantly among platforms. Mainstream platforms and POWER8 perform well on all memory hierarchy levels. Emerging ARM platforms, ThunderX and X-Gene, have somewhat higher latency. KNL has significantly higher latency especially for the datasets that exceed 1 MB. Since these latencies have a direct performance impact, especially for the workloads with a high rate of dependent memory accesses, they are an important parameter to consider. KNL platform is especially interesting since it incorporates high-bandwidth MCDRAM, based on 3D-integration. While KNL delivers memory bandwidth far superior to any other platform under study, it comes with the cost of the memory access latency that exceeds 3× the latency on mainstream platforms. Finally, it is also important to notice that most of the KNL memory access penalty does not come from the memory device itself. DDR3 and DDR4 modules timing parameters are standardized by JEDEC [19], and the variation between them (in nanoseconds) is negligible. Still, KNL DDR4 access is around 150 ns slower than other platforms. Therefore, the KNL memory access penalty originates mainly from handling the memory request between the last-level cache and the memory device, i.e. from the memory queues and memory controller. It is interesting to see whether future architectures will succeed in incorporating 3D-stacked memory without a significant latency overhead.

D. Byte/FLOP ratio

Using the node-level measurements of FLOPs and memory bandwidth, in Figure 4 we show the ratio between sustained memory bandwidth and FLOPS. This ratio presents the amount of data (in bytes) which has to be transfered from main memory, in order to perform one floating-point operation. Platforms with a low Byte/FLOP ratio are well suited for compute-intensive applications such as HPL. In these platforms, for real applications memory bandwidth may easily become a performance bottleneck. The platforms with a high Byte/FLOP ratio perform well with applications that put a high pressure on memory bandwidth, such as HPCG. In this case, floating-point processing power may limit the performance.

We detect a huge difference in the Byte/FLOP ratio among the platforms under study. The measured Byte/FLOP ratio ranges from 0.05 (KNL-DDR4) to 1.07 (X-Gene 1), a difference of more than 21×. For mainstream HPC systems (Nehalem, Sandy Bridge and Haswell), the Byte/FLOP ratio is significantly below 1, and it has the tendency of decreasing [20], which does not serve well for memory-bound HPC workloads. As seen in Section III-B, current DDRx technology cannot

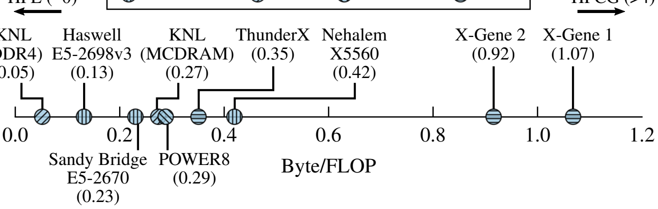


Fig. 4: Ratio between sustained memory bandwidth and FLOPS of the platforms under study can differ up to 21×.

keep up with aggressive FLOPs performance increases, so further progress in memory bandwidth relies on high-bandwidth memory solutions based on 3D-integration. In this respect, the KNL platform has a much higher Byte/FLOP ratio using MCDRAM than DDR4. Emerging systems, on the other hand, show a promising ratio, which is higher than mainstream platforms. This is mostly because the sustainable memory bandwidth is currently comparable between mainstream and emerging platforms, while the FLOPs performance of emerging systems is significantly below the mainstream ones. If this ratio keeps up with future developments of emerging platforms, we could see systems that cope better with memory-bandwidth intensive HPC workloads. Since HPC system performance strongly depends on the Byte/FLOP ratio, we advocate for this ratio to be constrained more precisely for HPC systems.

E. Theoretical vs. sustained FLOPs/s and memory bandwidth

As the final step of our analysis, we compare the maximum theoretical FLOPS performance and memory bandwidth from platform datasheets with the sustained values measured using DGEMM and STREAM. This comparison is important because sometimes theoretical numbers are used to compare platforms or estimate large-scale system performance before they are built. Our results, however, show that the differences between theoretical and measured numbers may be significant.

The results are displayed in Figure 5. Mainstream HPC systems based on Sandy Bridge and Haswell deliver sustained FLOPS performance and memory bandwidth close to theoretical maximums. Some emerging architectures, however, reach moderate FLOPS and memory bandwidth utilization even when running the DGEMM and STREAM benchmarks. For example, X-Gene 1 and KNL reach only 48% and 56% of the maximum theoretical FLOPS, while X-Gene 2 and POWER8 achieve similar rates for memory bandwidth. An explanation could be that the overall system cannot fully utilize SIMD floating-point execution units or data-transfer mechanisms. By the overall system we include both hardware and system software, including the pipeline, out-of-order (OoO) engine, caches, compilers and scientific libraries. The HPC system software for emerging platforms is still under development; for example, the first math libraries for ARM-based servers were released two years ago [21]. Similar studies confirm that system software stack on emerging platforms is relatively immature, which limits the achievable performance [4], [22], [23]. Finally, ThunderX shows very low FLOPS and memory utilization of 23% and 27%, respectively. In this case, additional problem is the simplicity of the in-order core and poor performance of inter-socket communication.

This analysis has two outcomes. Firstly, we would avoid using maximum theoretical performance even for first-order provisioning or an early evaluation of the HPC system, especially for the emerging platforms. Secondly, for some of the platforms under study, the results also show notable room for performance improvement, which will hopefully motivate further development of compilers and scientific libraries for emerging HPC platforms.

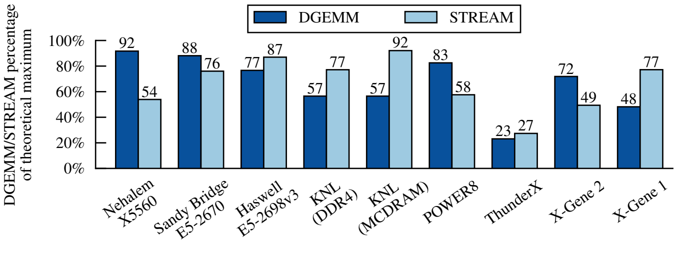


Fig. 5: Sustained FLOPS and memory bandwidth show significant difference to theoretical maximums, especially for emerging platforms.

IV. Related work

In addition to mainstream x86 architectures, emerging architectures based on POWER, ARM, and SPARC are steadily appearing and catching the attention of the HPC community. Although making the right choice of architecture is critical for the HPC infrastructure providers, only few studies evaluate and compare available emerging and mainstream HPC platforms.

The study of Rajovic et al. [2] is the first to analyze the suitability of mobile ARM processors for HPC. The study compares the performance and energy efficiency of development boards using mobile ARM 32-bit SoCs against a laptop with a Intel Sandy Bridge CPU. [3] Based on these measurements, the authors conclude that the performance and energy efficiency of mobile ARM platforms is competitive to the mainstream x86 HPC servers.

Abdurachmanov et al. [22] compare an X-Gene 1 development board with a dual-socket Intel Sandy Bridge server and Intel Xeon Phi PCIe add-on card. The authors compare only the CPU power consumption using on board sensors for X-Gene 1 development kit and Xeon Phi card, and RAPL interface [24] on the Sandy Bridge CPU. The study analyzes performance and energy efficiency of a single benchmark, ParFullCMS, and it concludes that the Sandy Bridge and Xeon Phi CPUs have similar performance that is 2.5× higher than X-Gene 1. Performance-per-watt results position Sandy Bridge as the most efficient platform, followed by X-Gene 1 (approximately 10% lower efficiency) and Xeon Phi (more than 35% lower efficiency w.r.t. Sandy Bridge).

Early evaluation of emerging platforms using developer kits is valuable and needed. However, we argue that energy-efficiency analysis requires measurements on the fully-fledged production servers, as performed in our study.

Rajovic et al. [3] also deploy a prototype cluster with nodes based on mobile ARM 32-bit SoCs and compare it with a production HPC Sandy Bridge cluster. The study also estimates the performance of the potential successor mobile SoCs with advanced ARM cores and embedded GPUs. The authors conclude that emerging ARM-based systems would offer performance equivalent to mainstream x86 systems, while saving 40% energy, and achieving higher integration density.

[3] In order to reduce the non-essential power consumption the authors switch off the laptop's screen.

However, these conclusions are based on two non-trivial HPC application requirements. First, the HPC applications would have to fully utilize the GPUs embedded into emerging SoCs, which is not the case for most current production HPC codes. For the applications that can fully utilize the GPUs, the CPU+GPU emerging systems should be compared with similar (CPU+GPU) mainstream platforms, not with respect to the CPU-only systems. Second, the application should have perfect parallel efficiency and load balancing when scaling-out from strong x86 cores to an approximately 4× larger number of weaker ARM cores. However, scale-out of production HPC applications typically leads to significant performance penalties [5]. Finally, the authors do not consider the performance and energy impact of RAS features (RAS: Reliability, availability and serviceability), such as memory ECC, available in the contemporary HPC systems, and not available on the emerging system under study.

Laurenzano et al. [4] compare the performance, power and energy consumption, and bottlenecks of Sandy Bridge, Atom, Haswell and X-Gene 1 servers. This analysis is based on system measurements with a large number of benchmarks and statistical modeling. The authors conclude that on average, for all the benchmarks under study, the X-Gene 1 and Atom servers have comparable performance, which is significantly below the Haswell and Sandy Bridge systems. When focusing on the workloads with an intensive use of floating-point and vector units, the study *estimates* similar performance on Sandy Bridge and X-Gene 1 systems. Our measurements, however, show that DGEMM performance on Sandy Bridge CPUs exceed X-Gene 1 and X-Gene 2 by 9× and 6×, respectively (see Section III-B). Regarding the energy efficiency, Laurenzano et al. measure similar results for the X-Gene 1 and Sandy Bridge, somewhat below the Atom and Haswell servers. For all the platforms under study, the authors perform server-level measurements, but then extract the *power resulting from executing the application* as a subtraction between the server power executing the application and the idle server power. Our position is that using this metric to quantify and compare energy efficiency is misleading and unfavorable for servers with higher energy proportionality, in which power consumption is highly correlated to server performance.

The conclusions of the studies that evaluate and compare emerging and mainstream HPC platforms depend significantly on the methodology and benchmarks used. Still, the related work shows there is no unified approach for this analysis, and that the conclusions are sometimes based on a methodology and assumptions open to discussion. In addition to a large body of quantitative results, this paper emphasizes usually-overlooked and important aspects of the HPC platforms evaluation. We believe this will improve the evaluation and comparison of HPC platforms, making a first step towards a uniform and more reliable methodology.

V. Conclusions

In our study, we perform an extensive analysis of HPC architectures, three mainstream and five emerging ones. To the best of our knowledge this is the first study to include so many platforms. In addition to presenting a large body of quantitative results, we highlight six important features in HPC systems evaluation that require higher attention by the community.

First, we show a platform's performance and energy-efficiency depend significantly (n-fold) on the characteristics of the target applications, We strongly advocate that any comparison among platforms should start with measurements using HPL and HPCG, which form the boundaries of compute-intensive and memory-intensive HPC applications.

Second, contrary to the conclusions of previous studies [2], [4], our measurements on fully-fledged HPC servers show that emerging ARM platforms generally show much lower energy efficiency than the KNL and mainstream Intel platforms.

Third, we (re-)open a discussion on whether HPC systems should be based on strong or weak CPU cores. Using weak cores to deploy a scale-out approach may cause significant performance and energy-consumption penalties due to the imperfect scalability of production HPC applications. It is therefore important to confirm that this approach is indeed a viable alternative.

Forth, our results show a huge range of memory access latencies, from $90\,\mathrm{ns}$ to $285\,\mathrm{ns}$ for the studied platforms. While KNL with MCDRAM has the highest mem. bandwidth, it also has the highest mem. latency, due to complex memory controller and its handling of memory requests. Since memory latency has a direct performance impact any increment above about 100 ns should be analysed and justified.

Fifth, we detect that the Byte/FLOP ratio can differ by a factor of up to 21× between platforms. While mainstream platforms show a decreasing tendency, emerging platforms trend upwards in this metric. We propose for a community to properly define this ratio for HPC applications, since it has a direct impact on system performance.

Sixth, our results show that sustainable performance on the emerging platforms can deviate more than 70% from theoretical performance. Therefore, we strongly suggest not rely on theoretical performance, even in a first-order system provisioning. These results will hopefully motivate further development of the compilers and scientific libraries for emerging HPC platforms.

Overall, our study provides a significant body of useful information for HPC practitioners and infrastructure providers. Even more important, we believe it will considerably improve the future evaluations and comparisons of HPC platforms, making a first step towards a more reliable and uniform methodology.

References

[1] TOP500 List, "Supercomputing Centers Have Become Showcases for Competing HPC Technologies," http://www.top500.org/news/supercomputing-centers-have-become-showcases-for-competing-hpc-technologies/, Jun. 2017.

[2] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC?" in *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 40:1–40:12.

[3] N. Rajovic, A. Rico, F. Mantovani, D. Ruiz, J. O. Vilarrubi, C. Gomez, L. Backes, D. Nieto, H. Servat, X. Martorell, J. Labarta, E. Ayguade, C. Adeniyi-Jones, S. Derradji, H. Gloaguen, P. Lanucara, N. Sanna, J.-F. Mehaut, K. Pouget, B. Videau, E. Boyer, M. Allalen, A. Auweter,

D. Brayford, D. Tafani, V. Weinberg, D. Brömmel, R. Halver, J. H. Meinke, R. Beivide, M. Benito, E. Vallejo, M. Valero, and A. Ramirez, "The Mont-blanc Prototype: An Alternative Approach for HPC Systems," in *Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 38:1–38:12.

[4] M. A. Laurenzano, A. Tiwari, A. Cauble-Chantrenne, A. Jundt, W. A. Ward, R. Campbell, and L. Carrington, "Characterization and bottleneck analysis of a 64-bit ARMv8 platform," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2016, pp. 36–45.

[5] D. Zivanovic, M. Radulovic, G. Llort, D. Zaragoza, J. Strassburg, P. M. Carpenter, P. Radojković, and E. Ayguadé, "Large-Memory Nodes for Energy Efficient High-Performance Computing," in *Proc. of the International Symposium on Memory Systems*, 2016.

[6] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, Mar. 1995.

[7] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers," http://www.netlib.org/benchmark/hpl/, Sep. 2008.

[8] J. Dongarra, M. Heroux, and P. Luszczek, "The HPCG Benchmark," http://www.hpcg-benchmark.org, 2016.

[9] V. Marjanović, J. Gracia, and C. W. Glass, *Performance Modeling of the HPCG Benchmark.* Springer International Publishing, 2014, pp. 172–192.

[10] 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 *Proc. of the ACM/IEEE Conference on Supercomputing*, 2006.

[11] J. D. McCalpin, "STREAM: Sustainable Memory Bandwidth in High Performance Computers," University of Virginia, Tech. Rep., 1991-2007. [Online]. Available: http://www.cs.virginia.edu/stream/

[12] L. McVoy and C. Staelin, "Lmbench: Portable Tools for Performance Analysis," in *Proc. of the Annual Conference on USENIX Annual Technical Conference*, 1996.

[13] Z. Ou, B. Pang, Y. Deng, J. K. Nurminen, A. Yl-Jski, and P. Hui, "Energy- and Cost-Efficiency Analysis of ARM-Based Clusters," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2012, pp. 115–123.

[14] A. Selinger, K. Rupp, and S. Selberherr, "Evaluation of Mobile ARM-based SoCs for High Performance Computing," in *Proc. of the High Performance Computing Symposium*, 2016, pp. 21:1–21:7.

[15] D. Zivanovic, M. Pavlovic, M. Radulovic, H. Shin, J. Son, S. A. Mckee, P. M. Carpenter, P. Radojković, and E. Ayguadé, "Main Memory in HPC: Do We Need More or Could We Live with Less?" *ACM Transactions on Architecture and Code Optimization*, vol. 14, no. 1, pp. 3:1–3:26, Mar. 2017.

[16] Partnership for Advanced Computing in Europe (PRACE), "Unified european applications benchmark suite," www.prace-ri.eu/ueabs/, 2013.

[17] Barcelona Supercomputing Center, "MareNostrum III System Architecture," http://www.bsc.es/marenostrum-support-services/mn3, 2013.

[18] S. W. Williams, A. Waterman, and D. A. Patterson, "Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-134, Oct. 2008. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-134.html

[19] Joint Electron Device Engineering Council, http://www.jedec.org.

[20] J. D. McCalpin, "SC16 Invited Talk: Memory Bandwidth and System Balance in HPC Systems." [Online]. Available: http://sites.utexas.edu/jdm4372/2016/11/22/sc16-invited-talk-memory-bandwidth-and-system-balance-in-hpc-systems

[21] ARM Ltd, "ARM Accelerates Mathematical Computation on 64-bit ARM-based HPC Systems," http://www.arm.com/about/newsroom/arm-accelerates-mathematical-computation-on-64-bit-arm-based-hpc-systems.php, Nov. 2015.

[22] D. Abdurachmanov, B. Bockelman, P. Elmer, G. Eulisse, R. Knight, and S. Muzaffar, "Heterogeneous High Throughput Scientific Computing with APM X-Gene and Intel Xeon Phi," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1410.3441

[23] I. Z. Reguly, A.-K. Keita, and M. B. Giles, "Benchmarking the IBM Power8 Processor," in *Proc. of the International Conference on Computer Science and Software Engineering*, 2015.

[24] Intel Corporation, "Intel® 64 and IA-32 Architectures Software Developer's Manual," Tech. Rep., Jul. 2017.