D2.2
Report on the ExaNoDe architecture design guidelines

Workpackage:	2	Co-Design for Exa-scale HPC System	
Author(s):	Milan Radulovic		BSC
	Kazi Asifuzzaman		BSC
	Darko Zivanovic		BSC
	Nikola Rajovic		BSC
	Petar Radojkovic		BSC
	Guillaume Colin de Verdiere		CEA
	Manolis Marazakis		FORTH
	Nikolaos Kallimanis		FORTH
	Dirk Pleiter		Juelich
Authorized by	Petar Radojkovic		BSC
Reviewer	Denis Dutoit		CEA
Reviewer	Guy Lonsdale		Scapos
Reviewer	Francis Wray		Scapos
Dissemination Level	Public		

Date	Author	Comments	Version	Status
2016-10-31	Petar Radojkovic	All the results are included. Solid first version of the text. Ready for the review by the D2.2 contributors.	1.0	Draft
2016-11-07	Petar Radojkovic	Included the comments of the T2.2 contributors. Document ready for the internal review.	1.1	Draft
2016-11-28	Petar Radojkovic	The comments of the reviewers are addressed.	2.0	Revised Draft
2016-12-13	Dirk Pleiter	Updates partially based on comments by reviewers. The document is ready to be delivered to EU.	2.1	Revised draft

Executive Summary

As various servers with different characteristics are hitting the market, it is essential to define and use a systematic methodology to compare them in terms of features that are important for HPC. In the context of the ExaNoDe Task 2.2, we defined the methodology for comparison of HPC compute nodes, and compared various main-stream and emerging HPC platforms.

First of all, we define the methodology for the comparison of existing and emerging HPC compute nodes restricting ourselves to designs without accelerators. We suggest the comparison based on the floating-point performance, memory bandwidth, latency and capacity. We also analyse the performance of different compute nodes when running important HPC benchmarks, HPL, HPCG and FFT, and consider server power consumption and RAS features. The investigated single compute nodes have 1-2 processors mounted with all of the considered processors comprising multiple cores. The presented methodology provides a good overall picture of an architecture targeting the HPC domain, while it is still simple, easy to understand and based on the benchmarks that are widely accepted by the HPC community.
In the Task 2.6, the same methodology will be used to do the preliminary evaluation of the ExaNoDe prototype, and determine its performance boundaries and potential bottlenecks.

Our analysis showed that, before the design of the HPC node, it is essential to decide which HPC approach should be followed: nodes based on server-level processors with strong cores or weaker processors, which requires a larger number of nodes to achieve similar performance. The main-stream HPC follows the former approach by using strong cores (e.g. Intel Haswell). The cores that will be used in the ExaNoDe prototype show performance significantly below the main-stream HPC ones; still they could be used to build massively parallel systems, using higher number of cores, sockets and servers. Therefore, in the context of the project it is important to deploy integration technologies (3D integration, Multi-Chip-Modules) which could help to improve scalability.

Our work also provides the recommendations for the design of HPC server CPUs, memory system and resiliency features.

Regarding the floating-point performance, we would recommend building systems that can provide high utilization of the execution units (e.g. above 80%), at least when running compute intensive benchmarks. Most of the emerging platforms under study, however, show a large gap between the theoretical and sustained floating point performance.

On the memory side, it is important to consider three design aspects – latency, bandwidth and capacity. Main memory latency of main-steam HPC servers under study is below 100ns, while for some of the emerging platforms it exceeds 250ns. Since main memory latency has a direct performance impact, HPC servers should try to minimize it, and any increment above the current 100ns should be analysed and justified. Our memory bandwidth analysis clearly showed that ratio between the memory bandwidth and the floating-point performance of the main-stream HPC platforms was decreasing in last few generations. Paradoxically, the community is recognizing that, for a large class of HPC applications, memory bandwidth is the primary performance bottleneck. Therefore, our recommendation would be to maximize the utilization of the available ExaNoDe prototype memory bandwidth, and look toward high-bandwidth memories in future ExaNoDe architectures. Regarding the prototype memory capacity, we recommend provisioning of at least 2GB of main memory per core (application process), following the rule of the thumb used in the main-stream HPC systems.

Finally, we illustrate the importance of the reliability, availability and serviceability (RAS) features of the HPC platforms. We conclude that it is essential that the ExaNoDe prototype design includes RAS features at compute node level such as ECC in memory controller, support for patrol scrubbing, ECC or parity protected cache memories. Ideally, these features should be considered from the early design stage; otherwise, the prototype may not be a viable building block for the Exascale HPC systems.

Table of Contents

1 Introduction 4
2 Experimental platforms and methodology 5
3 FLOPs and memory bandwidth........ 8
  3.1 Experimental........ 8
  3.2 Roofline model: Performance........ 9
    3.2.1 Core measurements 9
    3.2.2 Node measurements 11
  3.3 Roofline model: Performance per Watt........ 13
  3.4 Sustained vs. theoretical FLOPs and memory bandwidth........ 14
    3.4.1 Node measurements 15
  3.5 Summary........ 15
4 Memory hierarchy: Caches and main memory 17
  4.1 Experimental........ 17
  4.2 Results........ 17
    4.2.1 Memory access latency: Caches and Main memory 17
    4.2.2 Main memory size........ 19
  4.3 Summary........ 20
5 Important HPC benchmarks: HPL, HPCG, FFT........ 21
  5.1 Experimental........ 21
  5.2 Results........ 21
    5.2.1 Performance........ 22
    5.2.2 Performance per watt........ 23
  5.3 Summary........ 25
6 Reliability, availability and serviceability........ 27
7 Conclusions 29

1 Introduction

As various servers with different characteristics and architectures are hitting the market, it is essential to define and use a systematic methodology to compare them in terms of features that are important for HPC. In the context of the ExaNoDe Task 2.2, we defined the methodology for comparison of HPC compute nodes, and compared various main-stream and emerging HPC platforms in terms of performance (for important HPC applications), power consumption and resiliency. To allow for a focus on different processor architectures, we do not consider node designs including accelerators.

The rest of the document is organized as follows:

Section 2 lists the platforms under study with their most important features and summarizes the overall methodology used.

In Section 3, we compare the floating point performance and memory bandwidth of the platforms under study. First, we introduce *Roofline*, the performance model that we use for this comparison, followed by the performance and performance-per-watt measurements on the main-stream platforms. An important decision that has to be made when various platforms are compared in terms of floating point performance and memory bandwidth is whether to use theoretical or sustained (measured) values. In Section 3.4 we compare these two approaches.

Section 4 is focused on the memory hierarchy. In this section, we compare the access latency of all the memory levels, from L1 cache to the main memory, and we analyse the trends in the main memory sizing in the state-of-the-art HPC systems.

Section 5 shows the performance and performance-per-watt of the main-stream platforms when running important HPC benchmarks: High-Performance Linpack (HPL), High Performance Conjugate Gradients (HPCG), and Fast Fourier Transform (FFT) benchmarks.

In Section 6, we analyse the reliability, availability and serviceability features of the state-of-the-at HPC platforms, and provide the guidelines for the ExaNoDe prototype.

Finally, Section 7 summarizes the main conclusions of our analysis.

2 Experimental platforms and methodology

This deliverable summarizes the comparison between various past, present and emerging HPC architectures. Some of the systems are fully-fledged servers, while some of the emerging ones are still at the level of the developer kit. The studied architectures are listed below:

- Juno r2 platform (released in 2015) is a developer kit forerunner of the ExaNoDe prototype
    - It is based on ARMv8-A 64bit architecture, with four Cortex-A53 and two Cortex-A72 cores
    - Represents single processor socket development kit

- Main stream (past and state-of-the-art) servers based on:
    - Intel Nehalem X5560 (released in 2009)
    - Intel SandyBridge E5-2670 (released in 2012)
    - Intel Haswell E5-2698v3 (released in 2014)

- Emerging servers based on:
    - IBM Power8 (released in 2014)
    - Intel Xeon Phi code named Knights Landing (released in 2016)
    - Dual-socket Cavium ThunderX server, based on ARMv8-A architecture (released in 2014)
    - Single-socket Applied Micro XGene2 server, based on ARMv8-A architecture (released in 2015)
    - Single-socket server of Applied Micro XGene1, based on ARMv8-A architecture (released in 2013)[1]

- Low-end platforms:
    - ARM: Raspberry Pi3, based on ARMv8-A 64bit architecture (released in 2016)
    - x86: Single-socket Intel Atom C2750 server(released in 2013)[2]

The most important features of all the architectures under study are summarized in Table 1. Our experiments characterize the platforms (fully-fledged servers and developer kits) on two levels:

- Core level: Refers to performance delivered by a single CPU core of the platforms under study.
- Node level: Refers to performance delivered by a platform, i.e. server or developer kit. By the term *node*, in this document we refer to a developer kit or a server (in case that a fully-fledged server for a given architecture exists).

Fair comparison of the platforms under study is a challenging task:

- The platforms are based on the CPUs with different Instruction Set Architectures (ISAs) and therefore different system software such as compilers and libraries for scientific computing. In this task we tried to identify for each platform, the systems software that provided the best performance.[3] This corresponds to the production use

[1]HPE ProLiant m400 Server Cartridge for HPE Moonshot System
[2]HPE ProLiant m300 Server Cartridge for HPE Moonshot System
[3] On POWER8 the ESSL library was not used as it was not available on the tested system.

of these platforms. Therefore, the conclusions of our work should not be understood as a comparison between different hardware (CPUs and memory), but a comparison of the platforms (systems) that also include the corresponding system software.

- The software was used as is and has not been tuned for each of the platforms by the ExaNoDe partners. The software tuning would require excessive effort, and it would be very difficult to quantify the effort needed to tune each of the benchmarks to each of the platforms.
- Some of the platforms under study are based on the developer kits while others are fully-fledged servers. Power comparison between the two groups of platforms has been avoided, because they are fundamentally different, so we argue that any power comparison could be misleading. It remains, however, a project task to assess the energy efficiency of the ExaNoDe architecture for the application portfolio defined in deliverable D1.1.

	ExaNoDe prototype forerunner developer kit	Mainstream servers			Emerging servers					Low-end platforms	
Platforms	Juno r2	Nehalem X5560	SandyBridge E5-2670	Haswell E5-2698v3	KnightsLanding	Power8	ThunderX	XGene2	XGene1	Atom C2750	Raspberry Pi3
Manufacturer	ARM	Intel	Intel	Intel	Intel	IBM	Cavium	AppliedMicro	AppliedMicro	Intel	Broadcom
Architecture	ARMv8-A	Nehalem	SandyBridge	Haswell	2nd gen. MIC	POWER8	ARMv8-A	ARMv8-A	ARMv8-A	Atom	ARMv8
Type	Dev. Kit	Server	Server	Server	Server	Server	Server	Server	Server	Server	Dev. Kit
Sockets	1	2	2	2	1	2	2	1	1	1	1
Cores per Socket	4xA53, 2xA72	4	8	16	68	10	48	8	8	8	4
CPU freq.	A53: 950 MHz, A72:1.2 GHz	2.8 GHz	2.6 GHz	2.3 GHz	1.4 GHz	3.69 GHz	1.8 GHz	2.4 GHz	2.4 GHz	2.4 GHz	1.2 GHz
Out-of-order	A53:No, A72:Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
DP Flops per cycle, per core	2 (A53) 2 (A72)	4	8	16	32	8	2	2	2	1.5	2
Caches	A53: L1: 32kB L2: 2MB (shared by 4xA53) A72: L1i: 48kB L1d: 32kB L2: 2MB (shared by 2xA72)	L1i: 32 kB L1d: 32 kB L2: 256 kB L3: 8 MB (shared by all cores in socket)	L1i: 32 kB L1d: 32 kB L2: 512 kB L3: 20 MB (shared by all cores in socket)	L1i: 32 kB L1d: 32 kB L2: 256 kB L3: 40 MB (shared by all cores in socket)	L1i: 32 kB L1d: 32 kB L2: 1MB (per each core)	L1i: 64 kB L1d: 64 kB L2: 512 kB L3: 6 MB (per each core)	L1i: 48 kB L1d: 32 kB L2: 16 MB (shared by all cores in socket)	L1i: 32kB L1d: 32kB L2: 256kB (shared by 2 cores) L3: 8MB (shared by all cores in socket)	L1i: 32kB L1d: 32kB L2: 256kB (shared by 2 cores) L3: 8MB (shared by all cores in socket)	L1i: 32 kB L1d: 24kB L2: 4 MB (shared by 2 cores)	L1: 32kB L2: 512kB (shared by all cores in socket)
Memory conf. Per socket	2x32bit chann. DDR3-1600	3x64bit chann. DDR3-1333	4x64bit chann. DDR3-1600	4x64bit chann. DDR4-2133	8 chann. MCDRAM 6x64bit chann. DDR4 2400 [4]	8 DMI chann. (16 rd, 8 wr lanes per chann, 8 Gb/s)	4x64bit chann. DDR3-1600	4x64bit chann. DDR3-1600	2x64bit chann. DDR3L-1600	2x64bit chann. DDR3L-1600	1x32bit chann. LPDDR2-900
Memory capacity per node	8 GB	24 GB	32 GB	128 GB	16 GB MCDRAM 192 GB DDR4	256 GB	128 GB	128 GB	64 GB	32 GB	1 GB
Main memory ECC	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

Table 1: List of the platforms used in the study with the summary of their most important features.

[4] The Knights Landing (KNL) platform has two different types of main memory, DDR4 and 3d-stacked MCDRAM. Since the DDR4 and MCDRAM have significantly different characteristics, we performed two sets of KNL measurements, one for each memory type. In quadrant mode, the KNL exposes the DDR4 as belonging to NUMA node `0` and the MCDRAM to NUMA node `1`. Therefore choosing one DDR4 or MCDRAM can be done by setting the *m* option of *numactl* command. For example to run the stream benchmark, the command "`numactl -m 0 ./stream.exe`" will force the memory allocation to the DDR4, whereas "`numactl -m 1 ./stream.exe`" will force using the MCDRAM.

3 FLOPs and memory bandwidth

3.1 Experimental

We compare the platform FLOPs and memory bandwidth by using a roofline model. We use the roofline model because is a simple, easy-to-make, and it visually presents the most important aspects of the HPC nodes – floating point performance, memory bandwidth, and power consumption. The roofline model visually correlates FLOPS performance of the machine with memory bandwidth and application operational intensity.[5] Operational intensity represents the number of executed FLOPs per byte of data transferred between the main memory and the CPU. The applications with *low* operational intensity perform *small* number of floating point operations per amount of data transferred from the memory and vice versa. The roofline curve for a given platform is comprised of two performance ceilings – sloped and the horizontal one, see Figure 1. The sloped line represents the performance ceiling limited by the memory bandwidth; this determines the upper performance bound for applications with low operational intensity. The horizontal line represents performance ceiling limited by the platform's floating point performance.

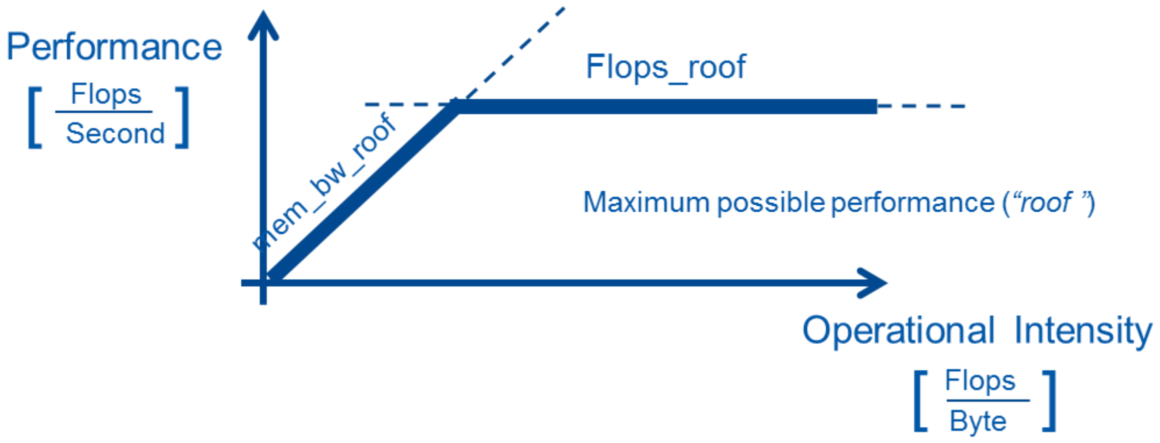

Figure 1: Representation of the roofline model. The sloped line represents the performance ceiling limited by the memory bandwidth; the horizontal line represents performance ceiling limited by the platform floating point performance.

The roofline model can be plotted based on the maximum theoretical or measured sustainable memory bandwidth and the floating point performance. Theoretical performance can be obtained from the product datasheets. Maximum sustainable values of the floating point performance and memory bandwidth can be measured by DGEMM and STREAM benchmarks, respectively. DGEMM[6] is a floating point intensive routine that calculates the product of double precision matrices: $C \leftarrow \alpha A \times B + \beta$. It represents Level 3 Basic Linear Algebra Subprograms (BLAS) routine. STREAM benchmark[7] performs operations on arrays

[5] S. W. Williams, A. Waterman, and D. A. Patterson. Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures. EECS Technical Report UCB/EECS-2008-134, Oct. 2008.

[6]J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff. ASet of Level 3 Basic Linear Algebra Subprograms. ACM Transactions on Mathematical Software, 16(1):1–17, Mar. 1990.

[7] J. D. McCalpin. STREAM: Sustainable Memory Bandwidth in High Performance Computers. https://www.cs.virginia.edu/stream/ref.html, 1997.

that are several times larger than the last level cache, and therefore puts high pressure to the memory bandwidth. It comprises of four kernels: Copy, Add, Scale and Triad. In this deliverable, we report the results of the Triad operation, since it is the most similar to kernels used in HPC applications. First we consider charts based on the sustained (measured) floating point and memory bandwidth values. In Section 3.4 we also compare the measured results with the theoretical ones.

3.2 *Roofline model: Performance*

We plot the roofline models at two levels:

- Compute node level (server or developer kit)
  - FLOPs roof: We execute multi-threaded DGEMM implementation, using all cores in the system.
  - Memory bandwidth roof: We execute multi-threaded STREAM implementation, using all cores in the system.

- Core level, i.e. performance delivered by a single CPU core
  - FLOPs roof: We execute DGEMM on a single CPU core
  - Memory bandwidth roof: We compute per-core ratio of the sustainable bandwidth as the node STREAM results divided by the number of cores. This would correspond to a fair share of the sustainable memory bandwidth delivered to each application process in case that the system is fully utilized.

3.2.1 Core measurements

Figure 2 shows the roofline model at the core level for the platforms under study. We plotted all measured data sets, but will in the following focus on the analysis and the conclusions that are relevant in the context of ExaNoDe project.

HPC systems that dominate the market are based on processors with high per-core (per application process) processing power and memory bandwidth. The KNL architecture is following this trend. Also, we notice that for each new generation of the x86 HPC CPUs (Nehalem → Sandy Bridge →Haswell) the increment in the processing power is much larger than the increment in the memory bandwidth. From example, moving from Nehalem to Sandy Bridge and Haswell increased per-core processing power from 10.6 to 24.9 GFLOPs (1.4x increment) and 47.3 GFLOPs (3.5xincrement), respectively. The sustainable memory bandwidth per core changed from 4.3 GB/s for Nehalem to 4.9 GB/s (13% increment) for SandyBridge and 3.7 GB/s (13.9% decrement) for Haswell.

Emerging ARM server processors XGene1 and XGene2 show significantly lower per-core processing power w.r.t. HPC processors used in systems currently listed on the TOP500 list (e.g., 44.9 GFLOPs (94.8%) and 43.4 GFLOPs (91.6%) lower w.r.t. Haswell), but the sustained memory bandwidth per core is still comparable, only 33.4% and 14.8% lower. For ThunderX we observed a noticeable floating-point performance drop of 1.96 GFLOPs (49.5%) and even higher sustainable memory bandwidth drop of 90.8% w.r.t. the XGene2.

Atom, a low-end x86 CPU shows lower performance, but comparable with XGene platforms –0.7 GFLOPs and 2.2 GFLOPs lower CPU performance and 0.3 GB/s and 0.9 GB/s lower per-core memory bandwidth w.r.t. XGene1 and XGene2, respectively.

Raspberry Pi3 as an example of a low-end ARM architecture, delivers 0.16 GFLOPs, which is, e.g., 47.2 GFLOPs lower performance than Haswell and 3.8 GFLOPs lower performance than XGene2. Raspberry Pi3 also delivers memory bandwidth of 0.55 GB/s, which is, e.g., 3.16 GB/s lower than Haswell and 2.61 GB/s lower than XGene2.

Juno r2 development kit, the ExaNoDe prototype forerunner is positioned very close to Raspberry Pi3 in terms of CPU core performance – it delivers 0.18 GFLOPs when using the A53 core and 0.65 GFLOPs when using the A72 core. In terms of sustained per-process memory bandwidth, Juno r2 is significantly better than Raspberry Pi3 (1.15 GB/s), and significantly below state-of-the-art x86 and emerging ARM HPC platforms (e.g. 2 GB/s and 2.6 GB/s lower per-core memory bandwidth w.r.t. XGene2 and Haswell, respectively).

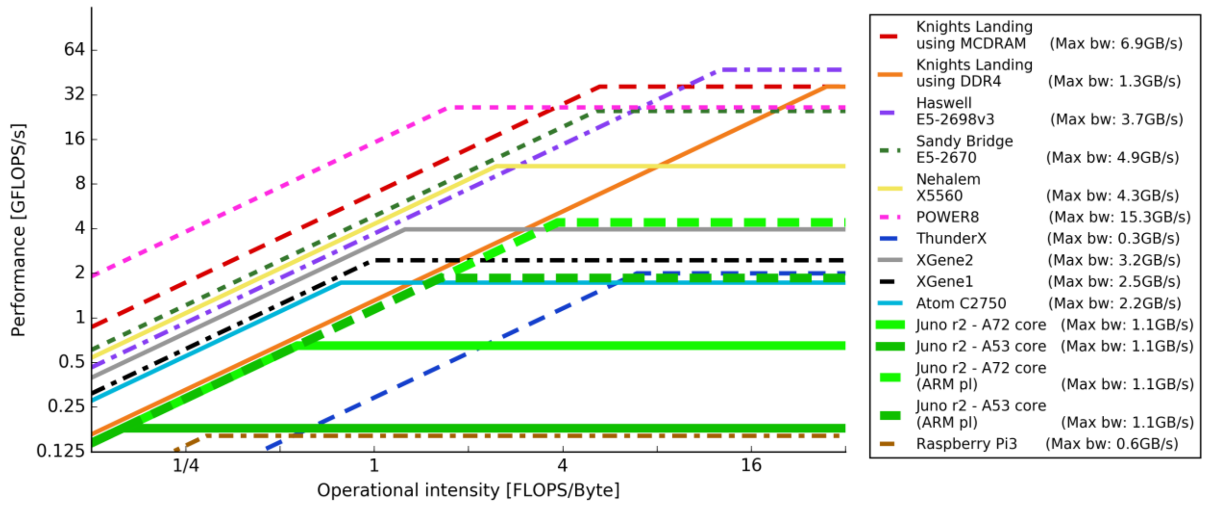

Figure 2: Roofline model of platforms under study, per-core granularity. The legend also includes the values of the sustainable memory bandwidth for all the platforms.

Regarding the sustained memory bandwidth, see Figure 3, POWER8 provides the best performance per core (15.3 GB/sec), followed by KNL using MCDRAM[8], Sandy Bridge, Nehalem and Haswell cores. XGene1 and XGene2 are the best ranked ARM platforms with the sustained per-core bandwidth of 3.2 GB/s and 2.5 GB/s.

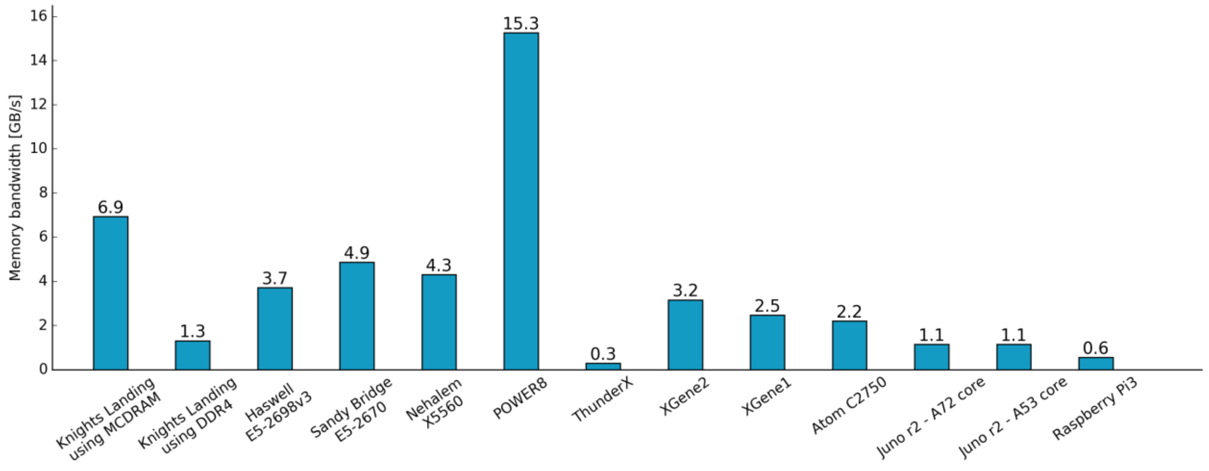

Figure 3: Sustainable memory bandwidth of the platforms under study, per-core granularity.

[8] KNL DDR4 sustained memory bandwidth per core if fairly low due to a large number of cores that the socket comprises.

In Figure 4, we also show a ratio between sustained FLOPs and memory bandwidth for the platforms under study (per-core granularity). Platforms with high a FLOP/Byte ratio are well suited for the applications that require high computation and have a small stress to the memory system, such as High-Performance Linpack (HPL). In these platforms, memory bandwidth may easily become a performance bottleneck. The platforms with lower FLOPs/Byte ratio perform well with applications that put a high pressure to the memory bandwidth, such as High Performance Conjugate Gradients (HPCG). In these platforms, performance may be easily limited by the floating-point processing power of the cores.

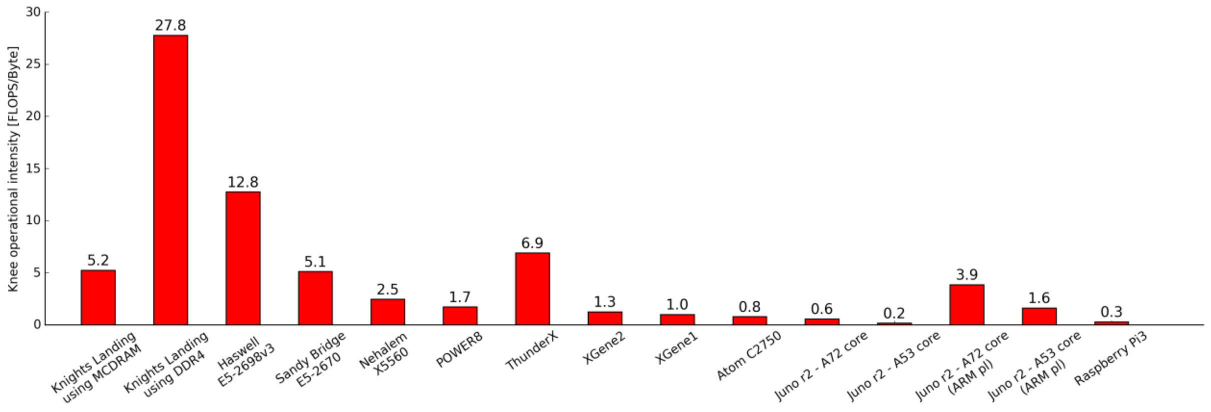

Figure 4: Ratio between sustained FLOPs and memory bandwidth of the platforms under study, per-core granularity.

3.2.2 Node measurements

Figure 5 shows the roofline model at the level of compute node, i.e. server or developer kit.

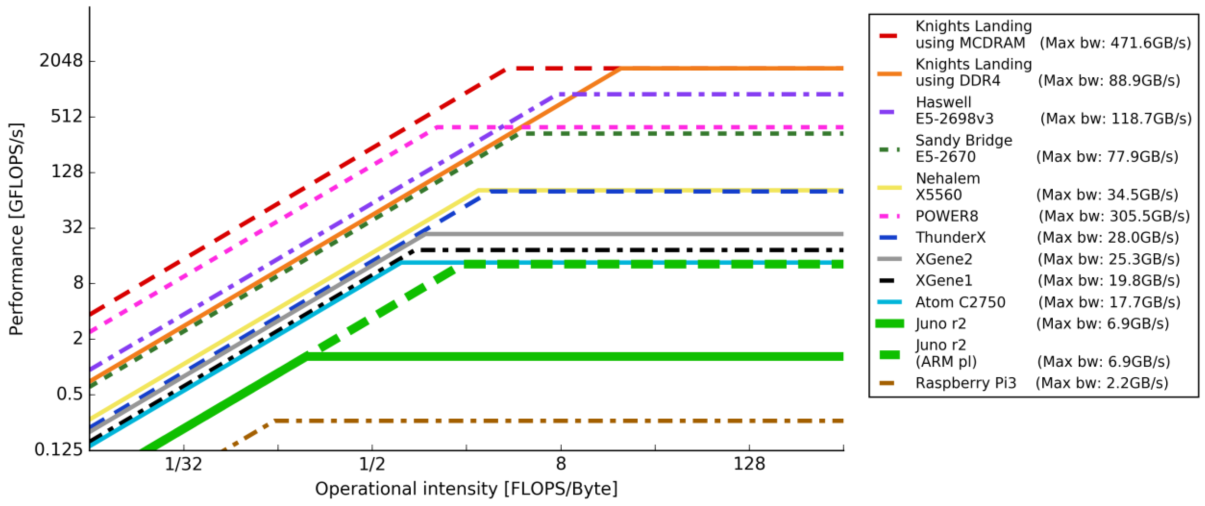

Figure 5: Roofline model of platforms under study, per-node granularity. The legend also includes the values of the sustainable memory bandwidth for all the platforms.

Although there are some differences w.r.t. to Figure 2,[9] we reach the same conclusions as from the per-core experiments.

Regarding the sustained memory bandwidth, KNL using MCDRAM provides the best performance (471.6 GB/sec), followed by the POWER8, Haswell and Sandy Bridge servers. The memory bandwidth of the KNL server when using DDR4 is still comparable with the state-of-the-art x86 servers. ThunderX is the best ranked ARM platform with a sustained bandwidth of 28 GB/s.

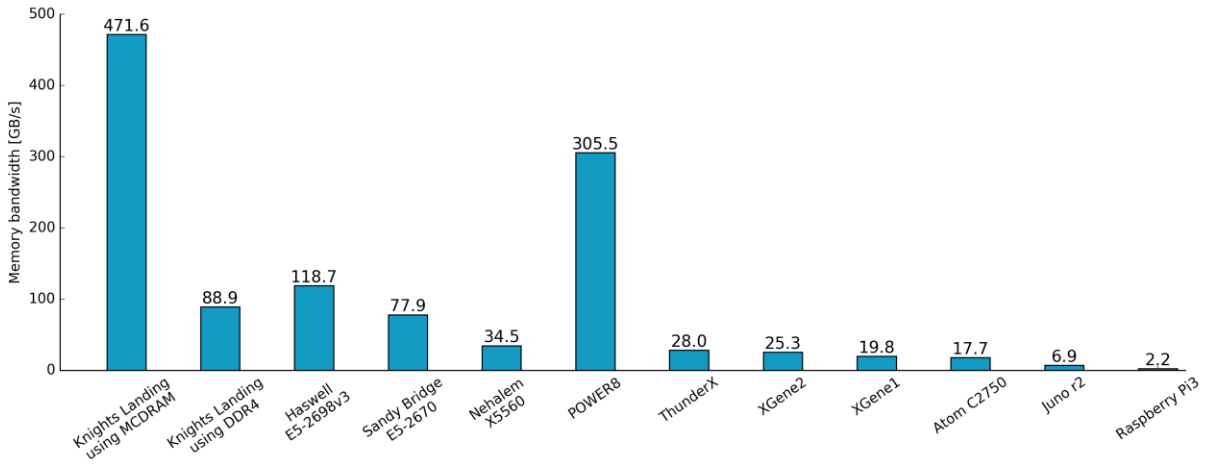

Figure 6: Sustainable memory bandwidth of the platforms under study, per-node granularity.

In Figure 7, we present a ratio between sustained FLOPs and memory bandwidth for the platforms under study, this time at the per-node granularity.

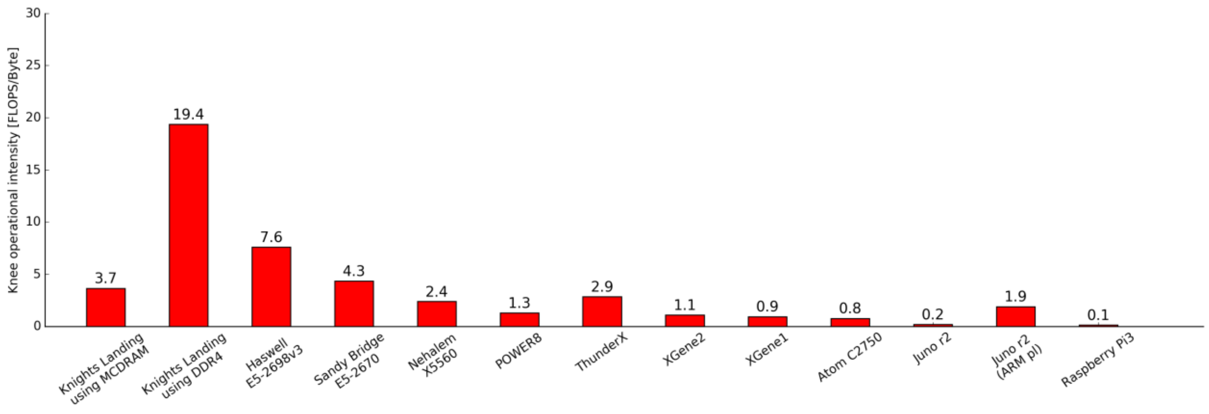

Figure 7: Ratio between sustained FLOPs and memory bandwidth of platforms under study, per-node granularity.

The presented results could open an important general question: Should one use powerful cores similar to Haswell as the building blocks for the large scale HPC systems[10] or would the use of weaker cores, such as Atom, XGene or ThunderX, be an alternative. Reaching the target performance would then require increasing the number of cores and compute nodes in

[9] For example, KNL platforms overcomes the Haswell performance; ThunderX shows higher performance that XGene1 and XGene2 systems, and it is approaching the Nehalem; Juno r shows significant improvement over the Raspberry Pi3.

[10] To clarify, we do not refer to using *a single* server as the first large vector supercomputers.

the system. The main-stream HPC platforms clearly follow the former approach. The performance of the A53 and A72 cores that will be used in the ExaNoDe prototype, is significantly below the, e.g. Haswell cores. As a part of the ExaNoDe approach towards Exascale HPC, it is essential to analyse the trade-offs between these approaches for production HPC applications. Our recent study is one of the first steps in this direction.[11] Looking towards future ExaNoDe architectures, it would be also interesting to characterize sustainable performance of ARM platforms with Scalable Vector Extension, the architecture that is planned to be the building block for Fujitsu's post K computer, Japan's next flagship supercomputer planned for the 2020 timeframe.[12]

Based on the Juno r2 developer kit result, we could estimate that the ExaNoDe prototype performance might not be competitive with the existing HPC platforms. However, it is important to put the results in the perspective and determine whether or not the technological improvements developed within the project (3D integration and Unimem for scale-out approach, HPC@FPGA for scale-up approach) applied to the state-of-the-art and future ARM server-class products would lead to a competitive HPC platform. Also, it would be interesting to see whether application developers could exploit the ExaNoDe prototype FPGAs, and what would be ratio between the FPGA theoretical and sustained performance when running important HPC benchmarks.

3.3 Roofline model: Performance per Watt

Besides the sustainable performance, we also compared performance relative to power consumption of the main-stream platforms under study. The ARM platforms considered in the context of this deliverable were development systems and thus could not be easily compared to fully fledged server systems. We restricted the comparison to a set of similar servers equipped with different generations of Intel Xeon processors. The comparison is displayed as a roofline model in Figure 8. Analogously to sustainable performance, there is a trend of increasing performance per watt moving from Nehalem to Sandy Bridge and Haswell, 0.86 GFLOPs/W, 1.48 GFLOPs/W and 1.96 GFLOPs/W, respectively.

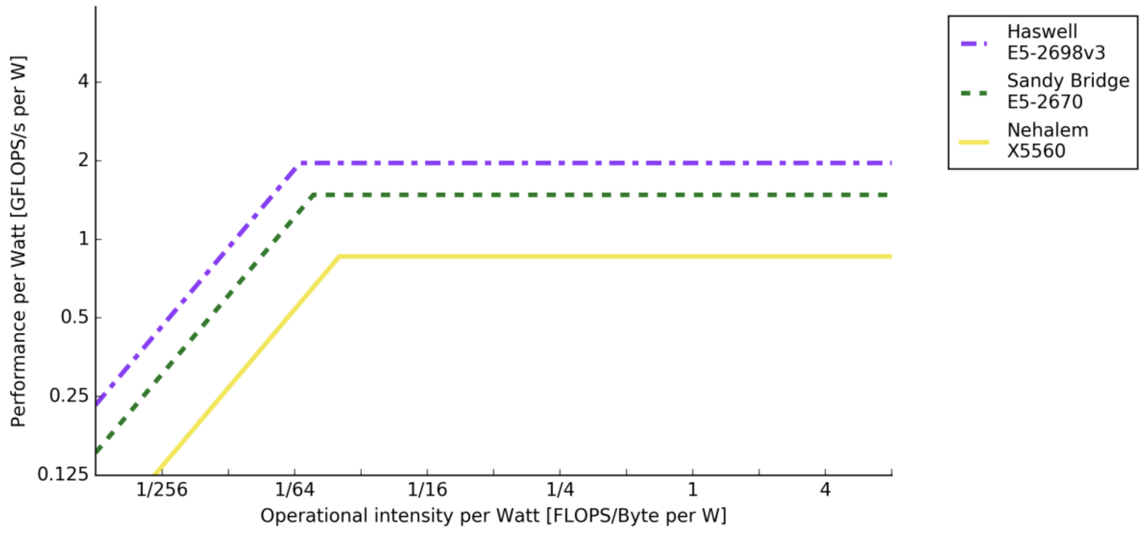

Figure 8: Roofline model showing performance per watt of main-stream platforms under study.

[11] D. Zivanovic et al., *"Large-Memory Nodes for Energy Efficient High-Performance Computing"*, In Proceedings of The International Symposium on Memory Systems (MEMSYS), 2016. (The best paper award.)

[12] ARM Unveils Scalable Vector Extension (SVE) for HPC. HPCWire, August 2016. https://www.hpcwire.com/2016/08/22/arm-unveils-scalable-vector-extension-hpc-hot-chips/

3.4 Sustained vs. theoretical FLOPs and memory bandwidth

In the previous charts, we plotted the sustained floating point performance and memory bandwidth – the values that are measured with DGEMM and STREAM benchmarks. As the next step of our analysis, we compare the measured sustained values with the theoretical ones, specified in the device datasheets. The results are plotted in Figure 9 and Figure 10.

Figure 9 plots the ratio of the sustained and theoretical FLOPs for the platforms under study. In these experiments, we execute the DGEMM on a single core, and compare the results with the theoretical maximal FLOPs values. POWER8 and the x86 platforms, Nehalem, Sandy Bridge, Haswell and KNL have very good floating-point unit utilization with the DGEMM rates of above 80% of the theoretical maximum; Sandy Bridge actually reaches the utilization of 94%. XGene1, XGene2 and ThunderX platforms reach a core utilization of 51%, 83% and 56%, respectively. For Atom we detect a lower core utilization of 44%. Finally, Juno r A53 and A72 cores use only 9% and 27% of the theoretical FLOPs, while the Raspberry Pi3 platform reaches only 7%.

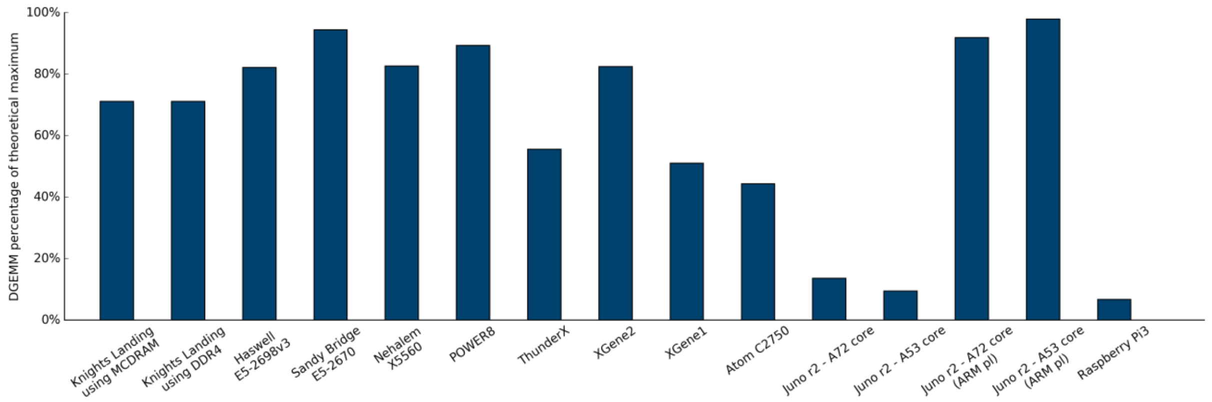

Figure 9: Theoretical vs. sustained floating point performance. Single-core analysis.

The results show that some architectures reach low floating-point unit utilization even when running the DGEMM floating-point computation intensive benchmark. An explanation for these results could be that the overall system cannot fully utilize the SIMD floating-point execution units. By the overall system we refer to both hardware and system software, including the pipeline, OoO engine, caches, compilers and scientific libraries. This result is surprising because traditionally HPC servers deliver sustained floating-point performance and memory bandwidth close to the theoretical maximums. The traditional HPC servers, however, have a strong system-software support for the HPC workloads, while the HPC system software for the emerging platforms is still under development; for example, the
math libraries for ARM-based HPC servers were released only a year ago.[13] Therefore, in a case of HPC system provisioning based on the emerging platforms, it is important to consider that sustained performance may be significantly lower than the theoretical ones. This low performance is, however, at least partially due to the low performance using commonly used open-source implementations of the benchmarks. For the Juno r A53 and A72 systems the benchmarks have also been executed using ARM's proprietary libraries.

---

[13] ARM Accelerates Mathematical Computation on 64-bit ARM-based HPC systems. ARM press release, November 2015. https://www.arm.com/about/newsroom/arm-accelerates-mathematical-computation-on-64-bit-arm-based-hpc-systems.php

3.4.1 Node measurements

Figure 10 shows similar results but on the node level. This time, in addition to the DGEMM benchmark (sustainable FLOPs) we executed the STREAM benchmark to measure the sustained memory bandwidth of the platform. Again, the overall conclusion is that, for some platforms (and most of the ARM platforms) we detect a large gap between theoretical and measured FLOPs and memory bandwidth.[14]Again, this has to be considered when doing even a first-order estimation of the performance of the systems based on these platforms.

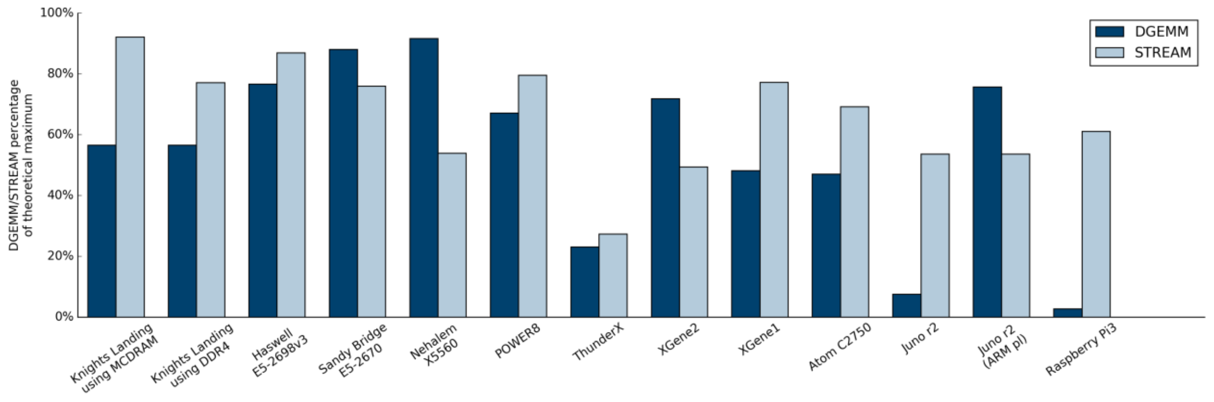

Figure 10. Theoretical vs. sustained floating point performance and memory bandwidth. Compute node analysis, server or developer kit.

3.5 Summary

The main outcomes of our analysis targeting the floating point performance and memory bandwidth are:

- The main-stream HPC platforms clearly follow the scale-up approach, with strong cores and servers. The performance of the A53 and A72 cores that will be used in the ExaNoDe prototype, is significantly below the, e.g. Haswell cores, so they could be used only in the scale-out approach, by targeting scalable, efficient system with large number of cores and sockets. As a part of the ExaNoDe approach towards Exascale HPC, it is essential to analyse the trade-offs between scale-up and scale-out HPC when running production HPC applications.
- Based on the Juno r2 developer kit result, we could estimate that the ExaNoDe prototype performance might not be competitive with the existing HPC platforms. However, it is important to put the results in the perspective and determine whether or not the technological improvements developed within the project (3D integration and Unimem for scale-out approach) applied to the state-of-the-art and future ARM server-class products would lead to a competitive HPC platform. Also, it would be interesting to see how this picture changes when integrating accelerators into the ExaNoDe node design, e.g. FPGAs.
- The results show that some architectures (especially the emerging ones) reach low floating-point unit utilization even when running DGEMM floating-point computation intensive benchmark. This could partially be attributed to lacking optimisation of software

[14]For some of the platforms, we performed a preliminary analysis about the sources of gap between the theoretical and sustained performance. For example, for the ThunderX platform we detected that one of the reasons of low STREAM scores is the low-performance of the interconnection between the sockets. When we adjusted the benchmark to use only the local-socket memory it showed close to 2x improvement w.r.t. the default execution.

stacks, which indicates that significant efforts are required at software level to fully exploit this new hardware.

4 Memory hierarchy: Caches and main memory

4.1 Experimental

Another important parameter that has to be considered in the HPC compute node design is the access latency to the caches and main memory.[15] More details about caches in all the platforms can be found in Table 1.

In order to quantify access latencies to different levels of memory hierarchy, we used one of the benchmarks from the LMbench benchmark suite.[16] It is a suite of simple, portable benchmarks, which compare different performance characteristics of Unix systems. It comprises both bandwidth benchmarks (cached file read, memory read/write, pipe, TCP etc.) and latency benchmarks (context switching, various networking latencies, memory read latency etc.). We used the memory read latency benchmark, with random-access reads in order to mitigate the impact of the data prefetching. By varying input dataset size, we could measure access latency to all memory hierarchy levels. The measured latency comprises the latency of the hardware components (caches, memory controller, main memory), but also the latency of the system software, and virtual to physical memory translation.

4.2 Results

4.2.1 Memory access latency: Caches and Main memory

In Figure 11 and Figure 12, we compare the latency of various levels of cache and main memory for the platforms under study. In Figure 11 we focus on the latencies for the LMbench input datasets ranging from 2KB to 256KB (X-axis). This covers L1 and L2 caches of the platforms under study. In Figure 12, we increase the array size up to 1GB, which covers the accesses to all levels of cache and main memory.

Even at the level of L1 and L2, we detect a significant difference in the latencies, see Figure 11.[17] For example, at the L1 cache (array size 2—16KB) the latency varies from 1.25ns (Atom, SandyBridge, Haswell, POWER8) to 3.3ns (Juno r2 A72 core). In the L2 cache (e.g. array size 128KB), the difference is even more significant, it ranges from 3.6ns (Nehalem, Sandy Bridge, Haswell, POWER8) to 23ns (ThunderX), while Juno r2 L2 cache latency is around 15ns.

As we increase the array size, see Figure 12, the difference in the access time between different platforms increases. For example, in the case of the 4MB array, the latency ranges from 23ns (Nehalem, Sandy Bridge, Haswell) to 175ns and 195ns for KNL using DDR4 and MCDRAM, respectively. The latency on the Juno r2 board is 150ns for A72 and 175ns for A53 core.

[15] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. ACM SIGARCH Computer Architecture News, 23(1):20–24, Mar. 1995.

[16]The LMbench benchmark suite. http:// http://www.bitmover.com/lmbench/lmbench.html.

[17]Measured latencies for L1 and L2 caches in cycles: KNL (MCDRAM): 4, 17.2; KNL (DDR4): 4, 17.2; Haswell: 4.8, 14.4; Sandy Bridge: 4, 12; Nehalem: 4.6, 11.4; POWER8: 3, 13; ThunderX: 3, 42; XGene2: 5, 13; XGene1: 5, 13; Atom: 3.3, 15.2; Juno r2 A72: 4, 17; Juno r2 A53: 3, 14.2; Raspberry Pi3: 3, 16.

The main memory latency measured by the LMbench (e.g. 256MB array) ranges from 90ns (Nehalem, Haswell and POWER8) and 105ns (Sandy Bridge, ThunderX) up to 250ns and 285ns for KNL using DDR4 and MCDRAM, respectively. The latency on the Juno r2 board is 195ns for A72 and 245ns for A53 core.

To summarize, cache and main memory latency is an important parameter to consider when designing or selecting HPC servers. Our measurements show that this latency can vary significantly among the platforms under study. Main-stream x86 platforms (Nehalem, Sandy Bridge and Haswell) and POWER8 perform well on all levels of memory hierarchy. Emerging ARM platforms, ThunderX and XGene2, have somewhat higher latency. Juno r2 and KNL (for both DRAM4 and MCDRAM) have significantly higher access latency than main-stream x86 or emerging ARM platforms especially for the datasets that exceed 1MB. In the context of the ExaNoDe project, the Unimem would enable sharing and borrowing of the main memory between the nodes. The usage of this approach to HPC applications however depends on the latency penalty of the remote memory access and its impact on the overall performance. Once that ExaNoDe prototype is deployed, this will be analysed in detail (ExaNoDe Task 2.6).

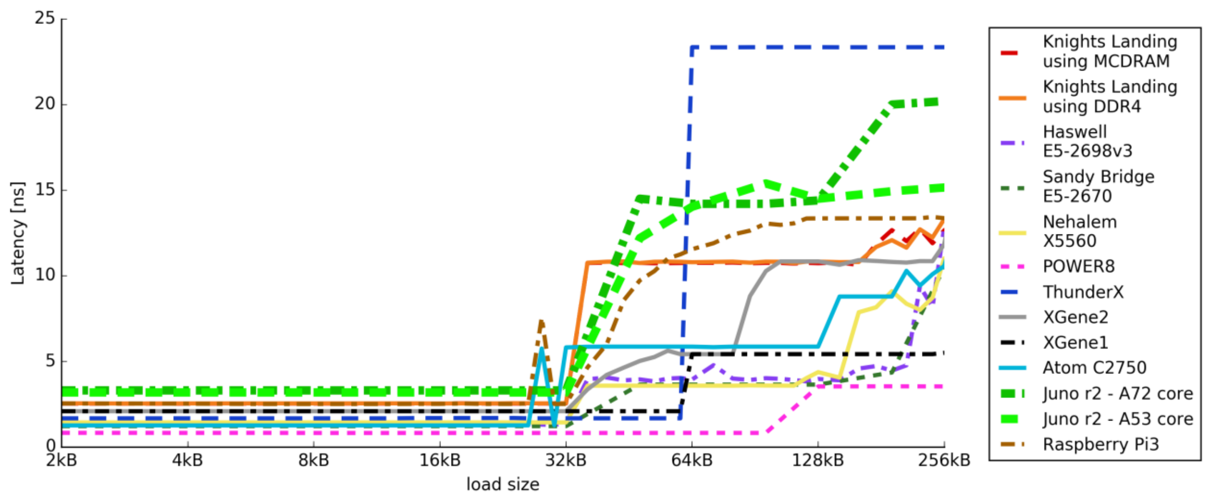

Figure 11: L1 and L2 cache access latency. X-axis shows the input dataset size.

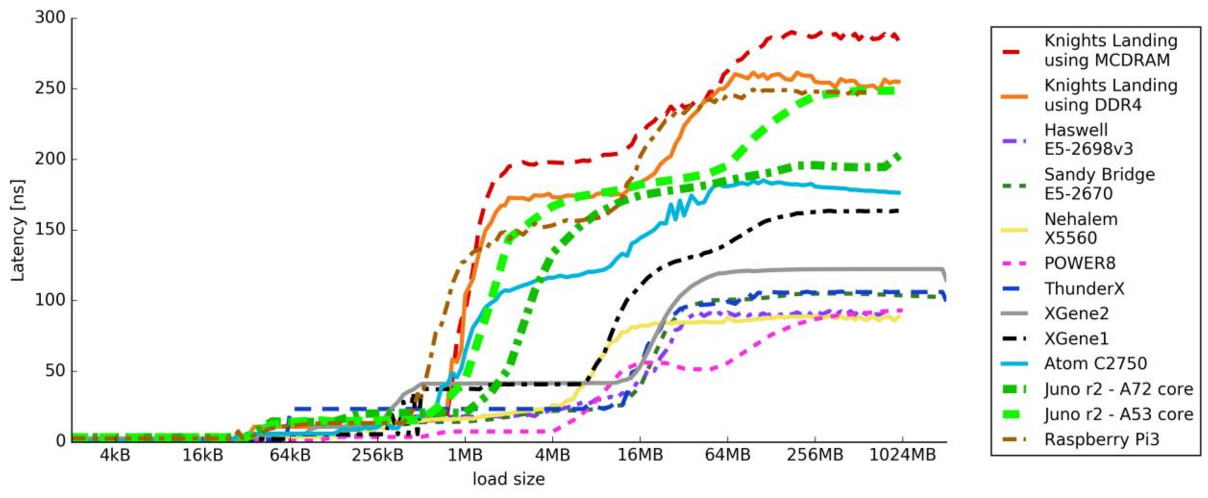

Figure 12: Access latency: L1, L2, L3 and main memory.X-axis shows the input dataset size.

4.2.2 Main memory size

The choice of main memory capacity is one of the most important aspects of HPC system design. Therefore, we analyse memory capacities of current HPC systems deployed world-wide. Figure 13, Figure 14 and Figure 15 show per-core memory capacities of the leading 50 TOP500-list systems with publicly available system architecture specifications, and the evolution of these systems from November 2014 until Jun 2016.[18] The figures cluster the systems into five groups based on per-core memory capacity and show the breakdown of CPU architectures in each group. These large-scale HPC systems are dominated by x86 architectures coupled with 2–3 GB of main memory per core. The next most prevalent systems are Blue Gene platforms based on PowerPC cores with 1 GB of memory per core (included in the [1,2) GB bar). This is a rule of thumb when provisioning the memory system in HPC. In the last two years, more systems with higher memory capacities have entered the TOP500 list. This trend is seen in the figures below, and over time, we see the increment in the number of systems with more than 4 GB of memory per core. Moreover, the majority of systems that comprise 2—3GB of memory per core have additional large-memory nodes that comprise 4x—8x more memory. In the context of the ExaNoDe project, it would be important to analyse memory capacity requirements for the application portfolio foreseen for the ExaNoDe prototype and potential follow-up commercial systems.

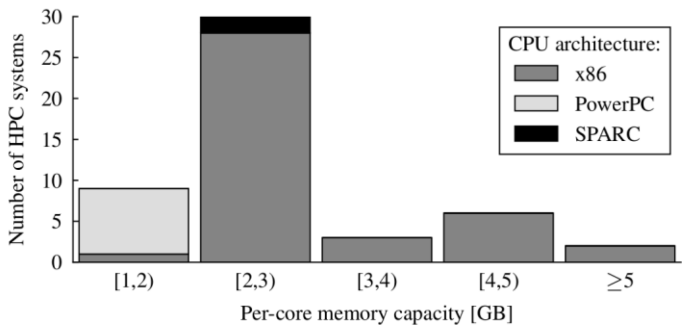

Figure 13. Per-core memory capacity of 50 HPC systems leading the TOP500 list, November 2014.

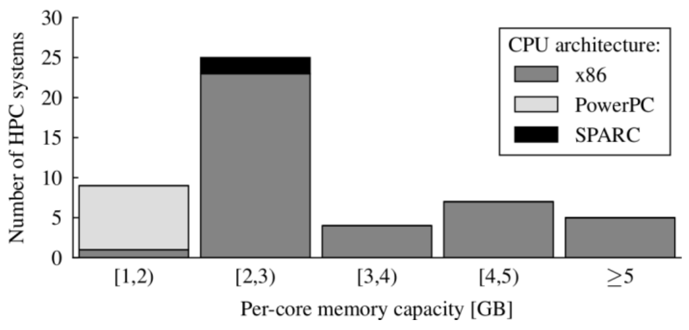

Figure 14. Per-core memory capacity of 50 HPC systems leading the TOP500 list, June 2015.

[18] TOP500 List. http://www.top500.org/.

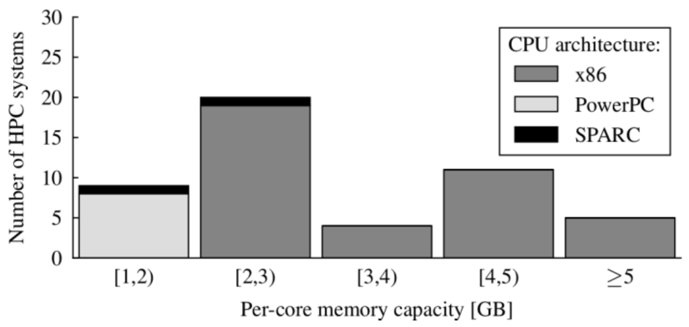

Figure 15. Per-core memory capacity of 50 HPC systems leading the TOP500 list, June 2016.[19]

4.3 Summary

The main outcomes of our analysis targeting the memory latency and capacity are:

- Our measurements show that cache and main memory latency can vary significantly among the platforms under study. In particular, Juno r2 platform (ExaNoDe prototype forerunner development kit) has significantly higher access latency than main-stream x86 or emerging ARM platforms especially for the datasets that exceed 1MB. During the ExaNoDe prototype evaluation this has to be analysed in detail because it may lead to a significant performance loss.
- In the context of the ExaNoDe project, the Unimem would enable sharing and borrowing of the main memory between the nodes. The usage of this approach to HPC applications however depends on the latency penalty of the remote memory access and its impact on the overall performance. Once that ExaNoDe prototype is deployed, this will be analysed in detail (ExaNoDe Task 2.6).
- The large-scale HPC systems are dominated by x86 architectures coupled with 2–3 GB of main memory per core, and over time, we see the increment in the number of systems with higher memory capacities. Regarding the ExaNoDe prototype main memory, we recommend provisioning of at least 2GB per core (application process), following the rule of the thumb used in the main-stream HPC systems.

[19]We exclude the top system Sunway TaihuLight from the chart, because it has a specific architecture. Sunway TaihuLight node comprise of 260-core processor with Sunway architecture and 0.13 GB of memory per core.

5 Important HPC benchmarks: HPL, HPCG, FFT

5.1 Experimental

The High-Performance Linpack (HPL) benchmark is the most widely recognized and discussed metric for ranking of HPC systems for more than 20 years. HPL measures the sustained floating-point rate (GFLOP/s) for solving a dense system of linear equations using double-precision floating-point arithmetic.[20] The linear system is randomly generated, with a user-specified size, so that the user can scale the problem size to achieve the best performance on a given system. The documentation recommends setting a problem size that uses approximately 80% of available memory. In our analysis, we follow this recommendation and measure HPL performance in GFLOP/s.

The High-Performance Conjugate Gradients (HPCG) benchmark is introduced as a complement to HPL and the TOP500 rankings, since the community questions whether HPL is a good proxy for production applications. HPCG is based on an iterative sparse-matrix conjugate gradient kernel with double-precision floating-point values.[21] HPCG is a good representative of HPC applications governed by differential equations, which tend to have much stronger needs for high memory bandwidth and low latency, and tend to access data using irregular patterns. As with HPL, the user can scale the problem size to achieve the best performance on a given system. In our analysis, we report the optimal HPCG performance in GFLOP/s.

BenchFFT benchmark incorporates a large number of publicly available Fast Fourier transform (FFT) implementations, in both C and Fortran, and measures their performance and accuracy.[22] The benchmark performs both real and complex transforms in one, two, and three dimensions. BenchFFT executes on a single core and its performance depends on the transform size, which represents the number of samples on which FFT is performed. Varying the transform size allows one to get a sense of the cache effects and compare FFT performance and efficiency for many different sizes on the same chart. In our experiments, we measured and reported the Bench FFT performance in MFLOP/s over various transform sizes. Since not all FFT implementations could be compiled on each of the platforms (because of different compilers and libraries), we compared a set of implementations which were compiled for most of the platforms under study. For POWER8, Sandy Bridge, ThunderX and XGene2, it is the *fftw3* implementation, while for other platforms it is one of the *ooura* implementations. In this report we compare only one-dimensional, complex, powers-of-two transforms, in double precision.

5.2 Results

Figure 16 shows the HPL performance on different platforms under study. For Nehalem, Sandy Bridge, Haswell, Knights Landing, ThunderX, XGene1, XGene2 and Atom platforms, we measure performance measured on an HPC server that comprise various sockets (see Table 1 for more details), while for Juno r2 and Raspberry Pi3 we measure performance on the available developer kits. In case that these emerging CPUs are subsequently offered as commercial products for HPC systems, the servers may comprise several (numerous) sockets,

[20] HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/.
[21] The HPCG Benchmark. http://www.hpcgbenchmark.org/.
[22] The BenchFFT benchmark: http://www.fftw.org/benchfft/

and the performance of each socket could exceed the ones that we measure on the developer kits, so the results that we present are a first-order comparison between different systems.

5.2.1 Performance

Main-stream x86 platforms show high HPL performance that increased significantly with each new generation, from 81.2 GFLOPS/s (Nehalem) to 303.4 GFLOPS/s (Sandy Bridge) and 772.3 GFLOPS/s (Haswell). ThunderX shows the best performance of the ARM platforms. Knights Landing has the best HPL scores, 2.4x over the second-ranked Haswell. We also detect a large gap between the main-stream x86 and emerging ARM platforms. In general, the HPL performance correlate well to the DGEMM results, see Figure 5 (Section 3.2.2)

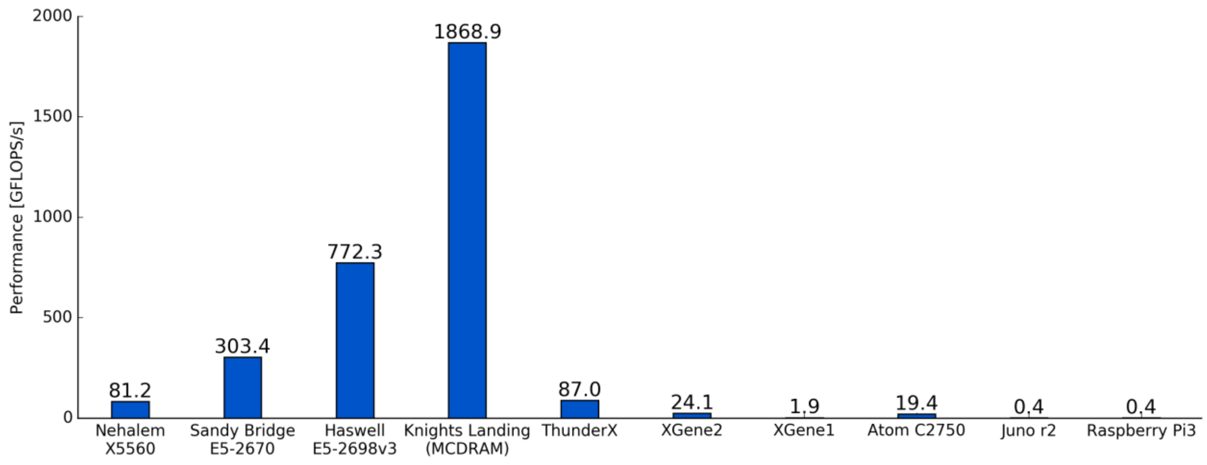

Figure 16. HPL performance.

Regarding the HPCG performance, again, main-stream x86 platforms show significant improvements between generations, from 3.0 GFLOPS/s (Nehalem) to 7.0 GFLOPS/s (Sandy Bridge), 10.7 GFLOPS/s (Haswell) and 16.6 GFLOPS/s (KNL), see Figure 17. The HPCG gap between main-stream x86 and ARM platforms is much smaller than the HPL gap. Actually, the ThunderX platform shows around 2x improvement w.r.t. to the Nehalem system, and approaches the Sandy Bridge server.

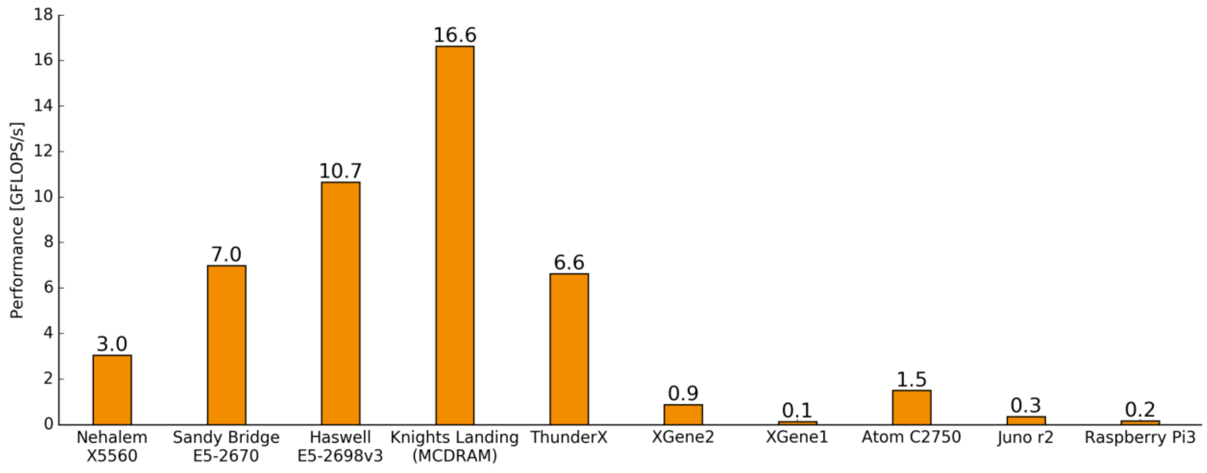

Figure 17. HPCG performance.

Figure 18 shows comparison of platforms under study for BenchFFT benchmark. The performance in MFLOP/s of one-dimensional, double-precision, complex FFT transform, is given as a function of FFT transform size, listed on the X axis. The BenchFFT performance highly depends on the transform size, which represents the number of samples on which FFT is performed. Varying the transform size allows one to get a sense of the cache effects and compare FFT performance and efficiency for many different sizes on the same chart. Similar to HPCG, FFT performance shows a relatively small performance gap between main-stream x86 and ARM platforms. The best FFT performance (approximately 18.5 GFLOP/s) is measured for the Sandy Bridge system, and this performance boost is caused mainly by the well optimized FFT library. POWER8 follows with approximately 6500MFLOP/s. Haswell, Nehalem and Knights Landing[23] platforms show similar FFT performance slightly below 5000 MFLOP/s. The best ARM platform in this benchmark is XGene2 achieving above 3000 MFLOP/s on certain transform sizes. Juno r2 and Atom reach around 2500 MFLOP/s. ThunderX and Raspberry Pi3 platforms are at the bottom of the list with the FFT performance of approximately 1300 MFLOP/s and 960 MFLOP/s, respectively.

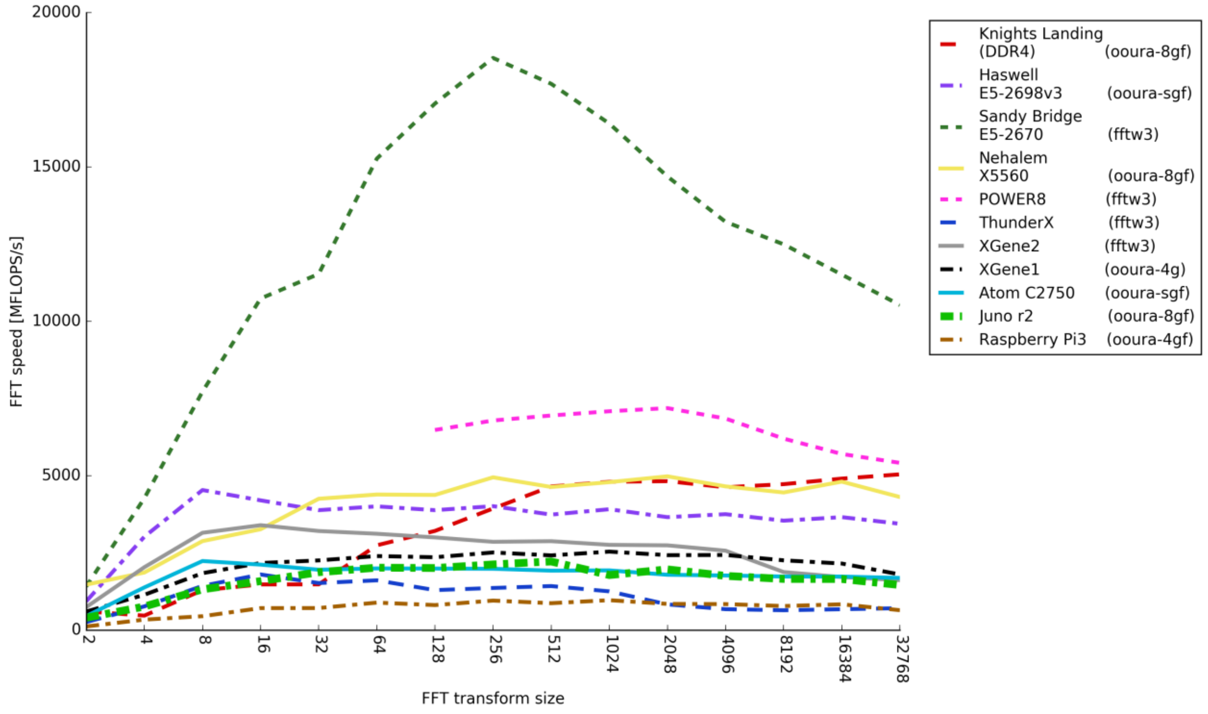

Figure 18. FFT performance.

5.2.2 Performance per watt

In Figure 19, we show that power consumption of the fully-fledged HPC servers used in the study, followed by the performance per Watt analysis. As in Section 3.3, for the sake of providing results that can be compared in a consistent way, we restrict ourselves to Intel Xeon based platforms.[24] In previous charts, we have seen that the performance increased significantly from Nehalem to Sandy Bridge and Haswell systems. Power consumption increment follows a much lower trend, in all three cases – idle system, HPL and HPCG workloads.

[23]FFT benchmark on KNL platform was executed using DDR4 memory, since excessive effort is required to modify the setup for benchmark to execute using MCDRAM.
[24] Energy efficiency analysis prototype remains a pending tasks for the evaluation of the ExaNoDe prototype within task T5.4 using the power measurement infrastructure provided by ETHZ.

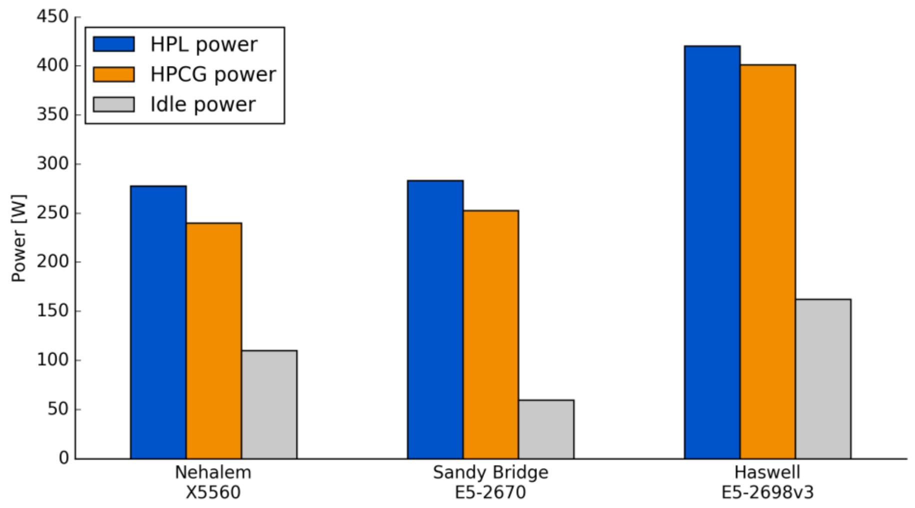

Figure 19. Power consumption of the HPC servers under study: HPL, HPCG power and Idle.

Overall, HPL performance-per-watt improved significantly from Nehalem to SandyBridge and Haswell architecture, more than 6x in the period of 5years (Figure 20). HPCG performance per Watt, also improved from Nehalem to SandyBrige (around 2x), but then it stagnated in the Haswell systems (Figure 21).

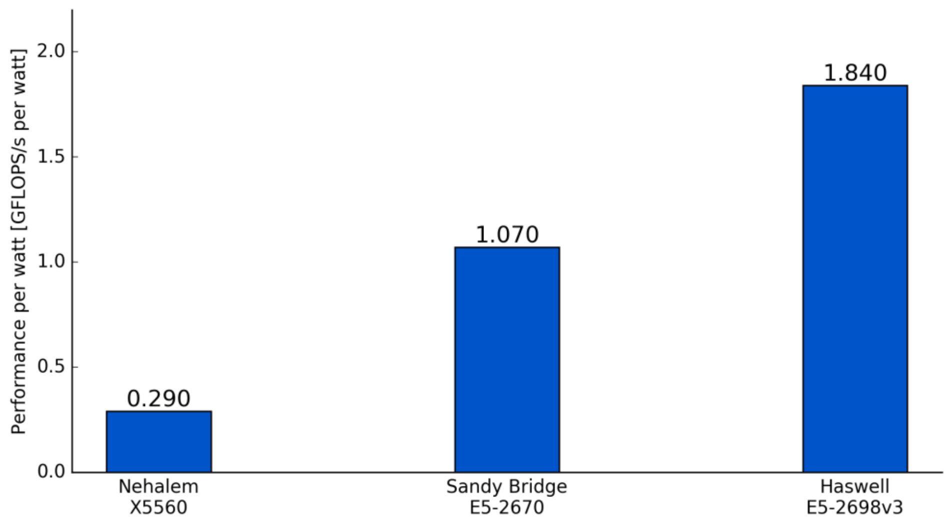

Figure 20. HPL performance per watt.

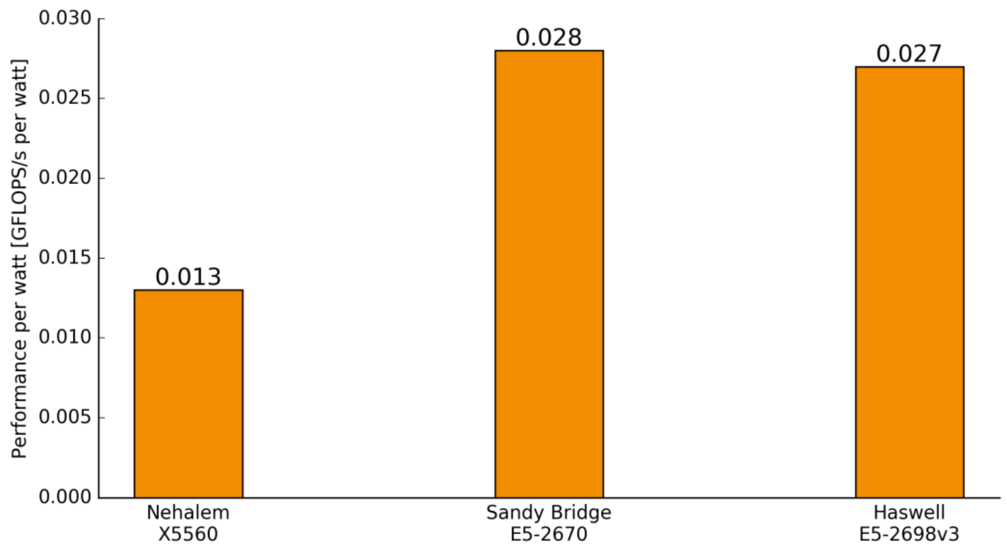

Figure 21. HPCG performance per watt.

In the context of the ExaNoDe project, we will also evaluate the prototype performance-per-watt ratio for important HPC benchmarks and analyze its power- and energy-proportionality.[25] The results will be compared not only with the state-of-the-art HPC systems (presented in the previous charts), but also with the requirement to reach the Exascale HPC within acceptable power budget.

5.3 Summary

The main outcomes of our analysis of the measurements with the of important HPC benchmarks, HPL, HPCG and FFT, are:

- Main-stream x86 platforms show high HPL performance that increased significantly in each generation. ThunderX shows the best performance of the ARM platforms. We detect a large gap between the main-stream x86 and emerging ARM platforms, e.g. Haswell provides 8.9x higher HPL performance then ThunderX servers. This performance difference can be at least partially explained by need for software optimised for ARM platforms.
- Regarding the HPCG performance, again, main-stream x86 platforms show significant improvements in each generation. The HPCG gap between main-stream x86 and ARM platforms is much smaller than the HPL gap, e.g. Haswell has only 60% higher HPCG performance w.r.t. to the ThunderX, the best-ranking ARM platform.
- Similar to HPCG, FFT performance shows a relatively small performance gap between main-stream x86 and ARM platforms.
- Overall, the HPL, HPCG and FFT measurements show the variety of HPC applications and their hardware requirements. In the context of the project, it will be important to identify a subset of HPC applications (and their characteristics) that will perform well on the ExaNoDe architecture (e.g. memory bandwidth intensive applications), and also understand which application characteristics might lead to performance bottlenecks (ExaNoDe Task 2.6).

[25]Hölzle and Barroso. The Case for Energy-Proportional Computing, Computer 40, 2007.

- Sandy Bridge servers show by far the best FFT performance. This performance is delivered mainly due to the well optimized FFT library; this result is a reminder of the system software importance in HPC domain.

Finally, it is also important to comment on the programmability of the HPC platforms. All the benchmarks used in this study are programmed with a standard and widely accepted set of languages such as C/C++/Fortran with MPI and/or OpenMP 4.x. The ExaNoDe prototype should keep the same level of programmability.

6 Reliability, availability and serviceability

Reliability, availability and serviceability (RAS) features are essential to consider in the design of large-scale systems, such as HPC supercomputers. Even though individual components such as CPUs or memory DIMMs are made very reliable, the probability of failure increases exponentially with the number of parts in the system and the number of components is likely to increase.

We will illustrate this with an example. Consider an exaflop class HPC machine with 500,000 processors. If a single processor has a statistical lifespan between errors of 5 years (43,800 hours or 2,628,000 minutes), the resulting 500,000 processors machine will have a fault every 5minutes on average (2,628,000/500,000). This example does not take other components into account (memory DIMMs, interconnect or storage) that will decrease the mean time between failures (MTBF) even further by their sheer number.

In state-of-the-art HPC systems based on publicly available data, MTBF is in the range of a few to tens of hours, depending on the system. For example, failures occur a few times per day on Titan at Oak Ridge National Laboratory (ORNL), and MTBF is 8—12 hours on BlueWaters and 37.5 hours on Jaguar at ORNL.[26,27]

The community is very clear about the importance of RAS features in HPC systems. At the Salishan workshop[28] organised by the US Department of Energy (DOE), the need to address these issues was made very clear. The talk IEEE Rel. Soc. SCV Meeting: 23 of March 2005*"How Cosmic Rays Cause Computer Downtime - Ray Heald"* is another illustration that "Soft errors" have been studied since the '70s and that they have to be taken into account from the first day of system design. The European Technology Platform for High-Performance Computing (EPT4HPC) is also very clear about the importance of increasing the system resiliency when targeting exascale.[29,30]

State-of-the-art supercomputer implements RAS features and the HPC centre system administrators monitor the machine health. For example, on the Curie supercomputer[31]we find:

- Error-correcting code (ECC) in memory controller that protects the memory DIMMs and the busses between the CPU and DIMMs: Single Correctable Errors are corrected automatically by ECC and reported to the monitoring system. Double Un-correctable Errors are monitored and lead to a node crash or halt for maintenance.
- Support for patrol memory scrubbing[32].

[26]Chao Wang, Sudharshan S. Vazhkudai, Xiaosong Ma, Frank Mueller, "Transparent Fault Tolerance for Job Input Data in HPC Environments ", Chapter in "Handbook on Data Centers", edited by Albert Y. Zomaya and Samee U. Khan, Springer, 2014.
[27]Yves Robert, "Fault_Tolerance Techniques for Computing at Scale", Keynote Talk, CCGrid 2014.
[28] Salishan Conference on High Speed Computing 2015. Resilience at Scale. http://salishan.ahsc-nm.org/2015.html
[29] European Technology Platform for High-Performance Computing (EPT4HPC). *Strategic Research Agenda. Achieving HPC Leadership in Europe.* 2013.
[30] European Technology Platform for High-Performance Computing (EPT4HPC). *Strategic Research Agenda 2015 Update. European Technology Multi-annual Roadmap Towards Exascale.* 2015.
[31]Curie supercomputer, http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm
[32] Bruce Jacob, Spencer W. Ng, David T. Wang "Memory Systems: Cache, DRAM, Disk", 2008, (Section 30.3.7, page 879).

- ECC or at least parity protected cache memories.
- Several probes on all motherboards are monitored by a baseboard management controller (BMC).
- A BMC on every node that collects all the logs coming from the probes and centralises them on a special cluster to allow close to real time analysis that can trigger alerts when, for example, the rate of Single Error in increasing on a node – a sign that a Double Un-correctable Error is about to happen.
- Checksum protected interconnect links.

In the context of ExaNoDe project, the system design must assure that the necessary features can be realised. Otherwise, the ExaNoDe prototype may not be a viable building block for the Exascale HPC systems.

7 Conclusions

As various servers with different characteristics are hitting the market, it is essential to define and use a systematic methodology to compare them in terms of features that are important for HPC. In the context of the ExaNoDe Task 2.2, we defined the methodology for comparison of HPC servers in terms of floating-point performance, memory bandwidth, latency and capacity, power consumption and RAS features. The presented methodology provides a good overall picture of an architecture targeting the HPC domain, while it is still simple, easy to understand and based on the benchmarks widely accepted by the HPC community. In Task 2.2, we also follow the presented methodology to compare various main-stream and emerging HPC platforms.

The main outcome of our analysis is a set of guidelines for the HPC node design:

- The first step in the HPC node design is to decide whether it use processors with strong or weaker cores. The main-stream HPC follows the former approach by using strong cores (e.g. Intel Haswell). The cores that will be used in the ExaNoDe prototype show performance significantly below the main-stream HPC ones; still they can be used to build massively parallel systems, using higher number of cores, sockets and servers. Therefore, in the context of the project it is important to deploy integration technologies (3D integration, Multi-Chip-Modules) which improve scale-out performances, analyse the trade-offs between scale-up and scale-out HPC, and confirm that scale-out approach indeed is a viable alternative.
- Regarding the floating-point performance, we would recommend building systems that can provide high utilization of the execution units (e.g. above 80%), at least when running compute intensive benchmarks. Many of the emerging platforms under study, however, show a large gap between the theoretical and sustained floating point performance, which can partially explained by lacking optimised software.
- On the memory side, it is important to consider three design aspects – latency, bandwidth and capacity. Main memory latency of main-steam HPC servers under study is below 100ns, while for some of the emerging platforms it exceeds 250ns. Since main memory latency has a direct performance impact, HPC servers should try to minimize it, and any increment above the current 100ns should be analysed and justified.
- Our memory bandwidth analysis clearly showed that ratio between the memory bandwidth and the floating-point performance of the main-stream HPC platforms was decreasing in last few generations. In contradiction, the community is recognizing that for a large class of HPC applications, the memory bandwidth is the primary performance bottleneck. Therefore, our recommendation would be to maximize the utilization of the available ExaNoDe prototype memory bandwidth, and look toward high-bandwidth memories in future ExaNoDe architectures.
- Regarding the prototype memory capacity, we recommend provisioning of at least 2GB of main memory per core (application process), following the rule of the thumb used in the main-stream HPC systems.
- We also illustrate the importance of the reliability, availability and serviceability (RAS) features for the HPC platforms and conclude that it is essential that the ExaNoDe prototype design includes the RAS features present in the state-of-the-art HPC servers. Ideally these features should be considered from the early design stage; otherwise, the prototype may not be a viable building block for the Exascale HPC systems.
- All the benchmarks used in this study are programmed with a standard and widely accepted set of languages such as C/C++/Fortran with MPI and/or OpenMP 4.x. The analysis has shown significant performance differences on emerging platforms using widely used open-source implementations versus highly optimised libraries provided by vendors. A focus on the ability of exploiting performance using commonly used development chains is of key importance for the success of emerging architectures.