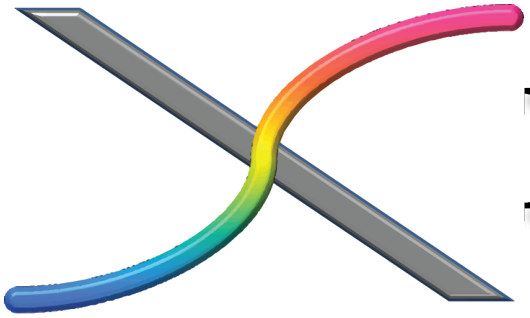
SEVENTH FRAMEWORK PROGRAMME
THEME ICT-2013.3.4
Advanced Computing, Embedded and Control Systems



EXCESS

Execution Models for Energy-Efficient Computing Systems
611183

D5.8
Report on the final evaluation results and discussion

Dmitry Khabi (HLRS), Dennis Hoppe (HLRS)
Ivan Walulya (Chalmers), Brendan Barry (Movidius)

Date of preparation (latest version): 30.08.2016

DOCUMENT INFORMATION

Deliverable Number	D5.8
Deliverable Name	Report on the final evaluation results and discussion
Authors	Dmitry Khabi (HLRS), Dennis Hoppe (HLRS), Ivan Walulya (Chalmers), Brendan Barry (Movidius)
Responsible Author	Dmitry Khabi (HLRS) e-mail: `khabi@hlrs.de` Phone: +49 711 685 65734
Keywords	High Performance Computing; Energy Efficiency
WP/Task	WP5/Task 5.5
Nature	R
Dissemination Level	PP
Planned Date	31.08.2016
Final Version Date	31.08.2016
Reviewed by	Paul Renaud Goud (Chalmers), Lu Li (LIU)
MGT Board Approval	YES

DOCUMENT HISTORY

Partner	**Date**	**Comment**	**Version**
Dmitry Khabi (HLRS)	11.06.2016	Initial structure of the deliverable	0.1
Dmitry Khabi (HLRS)	12.06.2016	Excess execution environment	0.2
Dmitry Khabi (HLRS)	11.07.2016	Evaluation results	0.3
Dmitry Khabi (HLRS)	12.07.2016	Excess HPC-Services	0.4
Dmitry Khabi (HLRS)	23.07.2016	Case study	0.5
Brendan Barry (Movidius)	01.08.2016	SGEMM Myriad2 results	0.55
Dennis Hoppe (HLRS)	12.08.2016	Review of evaluation results	0.6
Dmitry Khabi (HLRS)	01.08.2016	Conclusion	0.7
Dmitry Khabi (HLRS)	05.08.2016	Add SGEMM to evaluation results	0.8
Dmitry Khabi (HLRS)	12.08.2016	Add ethernet benchmark	0.9
Dmitry Khabi (HLRS)	15.08.2016	Small changes	0.91
Ivan Walulya (Chalmers)	16.08.2016	Add queue description	0.92
Dennis Hoppe (HLRS)	17.08.2016	Revision for internal review	0.93
Paul Renaud-Goud (Chalmers)	22.08.2016	Internal review	0.94
Lu Li (LIU)	23.08.2016	Internal review	0.94
Dmitry Khabi (HLRS)	23.08.2016	Integration of reviewer comments (Chalmers)	0.95
Dmitry Khabi (HLRS)	29.08.2016	Integration of reviewer comments (LIU)	0.96

Executive Summary

This document reports on work done in WP5 of the EXCESS project. Firstly, we present the EXCESS execution environment including conventional HPC hardware and embedded systems. We describe the architecture of the cluster and its power measurement system. The analysis of its accuracy and usability is also a part of the first chapter.

Moreover, we include in this report a thorough evaluation of the various benchmarks developed throughout the course of the project. We present interesting results with respect to performance, power and energy efficiency of various hardware components.

This report concludes with specific examples of how an embedded system, in particular the Movidius Myriad2, can be effectively exploited inside an HPC environment.

Contents

1 Introduction

This document presents the latest results obtained in work package WP5 of the EXCESS project. For brevity, we omit—as far as it does not impede comprehensibility—the fundamental descriptions and results obtained and described in previous EXCESS deliverables of WP5. Instead, we focus on key developments and investigations.

The remainder of this report is organized as follows: Firstly, we present in Section 2 the EXCESS execution environment. The key part of the environment is the EXCESS cluster including an innovative power measurement system and its integration into a HPC system.

Section 3 contains the metrics of various benchmarks obtained in our execution environment. We investigated the performance, power consumption and energy efficiency of various hardware components (GPUs and CPUs). Part of the work was also done in the field of embedded system, namely using Movidius embedded platforms.

Embedded systems potentially offer major improvements in energy efficiency. However, substantial efforts are still needed to overcome existing limitations, especially in problematic areas such as distributed computing. Such systems have a lower floating point and network performance than conventional systems. The overall objective is to find common problem-solving strategies in order to fully exploit the advantages of such systems. Hence, we provide two specific examples of how an embedded system can be used in the field of HPC. Thus, Section 4 explains briefly how to use the Movidius platform within an HPC cluster's environment, especially together with the PBS[1] system. The interested reader may find further information in Deliverable D5.6 [21]. Section 4.2 describes an alternative way of using embedded systems to solve common HPC problems. Especially, we highlight a use case that exploits the Movidius platform in a distributed simulation environment.

We conclude this report with a summary and final remarks in Section 5.

[1]Portable Batch System is a resource manager providing control over batch jobs and compute nodes.

2 EXCESS Execution Environment

The EXCESS cluster is a key part of the project's execution environment. It is used as an experimental platform for the development in the field of energy-aware high performance computing. Users of the cluster are enabled to perform technical implementations and evaluations, which are not feasible or at least currently limited on conventional HPC platforms. The EXCESS cluster provides the users with a common hardware and software environment: dual-socket compute nodes equipped with CPUs from two generations (*Ivy Bridge* and *Haswell*), a high-performance interconnect (*FDR InfiniBand*), a current Linux operating system, a portable batch system for job execution (*PBS*), various compilers and common development tools. In addition to this basic environment also found on conventional HPC platforms, the EXCESS cluster provides the users with the possibility to tune both hardware and software. Furthermore, the cluster includes additional components to monitor of those hardware and software components (i.e., the EXCESS monitoring framework) in order to support applications at runtime to determine the optimal configuration (or resource allocation, respectively) through a correlation analysis. This section briefly describes the latest cluster built with a special focus on hardware and the installed power measurement system.

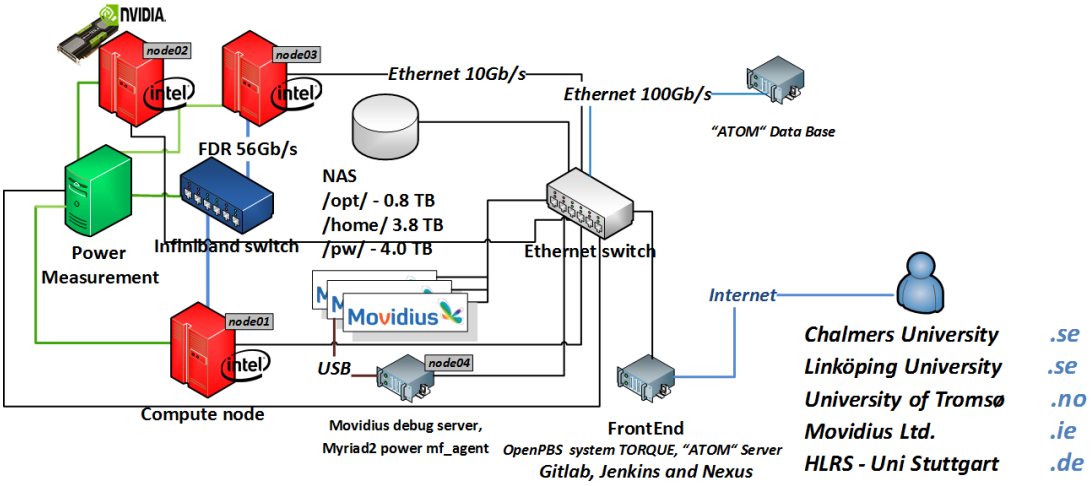


Figure 1: The EXCESS cluster

2.1 EXCESS Cluster

The cluster consists of three compute nodes, three Movidius platforms (Myriad2 development boards), and other essential infrastructure components such as a NAS server, a front end node, a power measurement system, and network switches (see Figure 1). The interested reader can find a detailed overview of these components in Deliverables D5.2 and D3.4 [19, 15].

2.2 Intel Platforms

The EXCESS cluster is a heterogeneous computing platform composed of three compute nodes: *node01*, *node02*, and *node03*. Tables 1 and 2 exemplifie a common configuration. It

should be noted that the compute node *node02* contains identical components as *node01*, except that it does not have a dedicated GPU, and that it is equipped with the power supply *EartWatts EA-650* instead of *Intel DPS-750XB A*.

Table 1: Configuration of compute nodes `node01` and `node02`

Amount	Hardware Component
1 x	Intel Server Chassis P4308XXMHGC;
1 x	Intel Server Board S2600COE
2 x	Intel Xeon CPU: E5-2690 v2 - 10 cores 25MB L3 smart cache 4 mem. ch. DDR3 1866 MHz = 59.7 GB/s[2]
2 x	Intel Heat sink AUPSRCB
8 x	HP Memory 4 GB DDR3 MFG 708633-B21 - 1866 MHz PC3-14900 CL13 - ungepuffert - Dual Rank - ECC[3]
1 x	GPU im node01: Tesla K40c - GPU Clock: 745 MHz; Shading Units: 2880 GDDR5 12288 MB; 288 GB/s; PCIe3.0 8GT/s x16[5]
1 x	Connect-IB HCA, Mellanox MCB193A-FCAT; single-port QSFP FDR IB (56Gb/s); PCIe3.0 8GT/s x16[4]
1 x	Hard disk: 500GB WD5003AZEX Black
1 x	SSD: 128GB Vertex450
1 x	Schaltnetzteil: node01: *DPS-750XB A REV02F* node02: *EartWatts EA-650*

Table 2: Configuration of compute node `node03`

Amount	Hardware Component
1x	Chenbro 4U 17.5" Compact Industrial Server Chassis RM42300
1x	Gigabyte Server Board - MD70-HB0 (Rev. 1,2)
2x	Intel Xeon CPU: E5-2680 v3 (Haswell) - 12 cores 30MB L3 Smart Cache 4 Mem. Ch. DDR4 2133 MHz = 68 GB/s [1]
2x	Enermax Heat sink ETS-T40-W
8x	SAMSUNG DRAM 16GB Samsung DDR4-M393A2G40DB0-CPB- 2133 MHz CL15 - Registered DIMM - Dual Rank - ECC [6]
1x	Intel Ethernet Server Adapter I350-T2 - PCI Express 2.1 x4 Low Profile - 1000Base-Tx2
1x	Hard disk: 500GB WD5003AZEX Black
1x	SSD: 240GB Vertex460A
1x	Power supply: `EartWatts EA-750`

2.2.1 Ivy Bridge

The *Ivy Bridge* processor *E5-2690 v2* was brought to the market in 2013. The CPU is equipped with 10 cores (20 HW-Threads), 25 MB *L3*-cache, 256 kB *L2*-cache and 32 kB *L1*-cache for data caching. The *L3*-cache is divided into 10 cache segments, which

Table 3: AVX-Turbo *frequency of Intel Xeon E5-2680 v3 [16].*

Cores 1-2	Core 3	Core 4	Cores 5-12
3.1	2.9	2.8	2.8

are connected with each other via a bi-directional bus—a so-called ring. Furthermore, 4 DDR3 memory channels, QPI and PCIe are connected to the ring.

The processor supports 15 different CPU frequencies: 1.2 GHz, 1.3 GHz, 1.5 GHz, 1.6 GHz, 1.7 GHz, 1.8 GHz, 2.0 GHz, 2.1 GHz, 2.2 GHz, 2.4 GHz, 2.5 GHz, 2.6 GHz, 2.7 GHz, 2.9 GHz, 3.0 GHz and a *Turbo* mode.

In *Turbo* mode, the processor is able to achieve up to zu 3.6 GHz. *Ivy Bridge*'s chip, like its successor *Haswell*, consists of *tri-gate* transistors[2] manufactured in 20 nm.

A single *AVX* register is able to store four floating-point numbers. Hence, one *AVX* instruction can perform up to four floating point operations in a cycle. The bandwidth between the *AVX* registers and the *L1* cache is 32 $B/cycle$ for read-access, and 32 $B/cycle$ for write-access. The bandwidth between *L1* and *L2* caches equals to 32 $B/cycle$ for read- and write-access.

The theoretical peak performance of the processor is 240 GFlops.

2.2.2 Haswell

The *Haswell* processor *E5-2690 v2* was brought to market in 2014. The *E5-2680 v3* is equipped with 12 cores (24 HW-Threads), 30 MB *L3*-cache, 256 kB *L2*-cache und 32 kB *L1*-cache for the data caching. In comparison to *Ivy Bridge*, *Haswell* has four *DDR4* memory channels, an extended ring, and additional minor modifications with respect to its architecture. The processor supports just 14 different CPU frequencies: 1.2 GHz, 1.3 GHz, 1.4 GHz, 1.5 GHz, 1.6 GHz, 1.7 GHz, 1.8 GHz, 1.9 GHz, 2.0 GHz, 2.1 GHz, 2.2 GHz, 2.3 GHz, 2.4 GHz, 2.5 GHz and the extra *turbo* mode.

The CPU frequency in *turbo* mode ranges from 2.8 to 3.3 GHz, depending on the used vector units (*AVX* or *SSE*), and the number of active cores. Tables 4 and 3 contain the frequencies by the computation with and without *AVX* instructions in *turbo* mode.

As mentioned above, a single *AVX* instruction can perform up to four multiplications or add operations in a single cycle. The new *FMA*[3] operation allows to perform four multiply-add operations and therefore doubles the peak performance of the processor. The bandwidth between the *AVX* registers and *L1* cache is doubled compared to *Ivy-Bridge*, and thus achieves 64 $B/cycle$ for read- and 32 $B/cycle$ for write-access. The bandwidth between *L1* and *L2* caches is also doubled, and equals to 64 $B/cycle$ for read- and write access. The theoretical peak performance of the processor is 480 GFlops. The interested reader can find further information about the processors in an online article [33], or in the technical description of the processors [16].

[2]A tri-gate or 3D transistor is used by Intel Corporation for the non-planar transistor architecture. Tri-gate transistors reduce leakage and consume less power than planar transistors.

[3]The so-called fused multiply-add operation computes the product of two numbers and adds that product to an accumulator.

Table 4: Turbo *frequency of Intel Xeon E5-2680 v3 [16].*

Cores 1-2	Core 3	Core 4	Cores 5-12
3.3	3.1	3	2.9

2.2.3 GPU Accelerators

As mentioned above, the compute node *node03* is equipped with the graphics processing unit (GPU) *NVidia Tesla K80*, which is the successor of the *NVidia Tesla K40*. In contrast to the K40, the *Tesla K80* has two independent GPU accelerators using a shared *PCIe 3.0* interconnect. Each of the *GPU*s has its own fast memory, the so-called *GDRAM*[4], and various types of caches. The processing cores are unable to communicate directly with the main memory of a compute node, though. As a consequence, either the input buffer to the GPU should be already in *GDRAM*, or an application needs to copy a buffer from the main memory into a local buffer on the *GDRAM* via the *PCIe* interface.

2.2.4 Embedded System Myriad2

Myriad2 is the name of the latest chip developed by Movidius. It is the successor of *Myriad1*. The embedded-board *MV182*, which has the *Myriad2* chip of revision *mv2150* installed, is part of the EXCESS cluster. The chip contains 2 cores of *Leon4* architecture and 12 accelerators, so called *SHAVE*s, which are optimized for performing computation-intensive tasks. The 2 *LEON4 RISC* processors are used to manage the execution: the *RISC1* core is designed to be the real-time controller for scheduling the activity of the chip, whereas the *RISC2* core is designed to run the operating system and to manage all the control interfaces.

The primary usage of *Movidius* systems is for embedded multi-media applications. For that reason, the *SHAVE*s do not support double floating point precision operations[5]. The vector units of the SHAVEs are able to store four floating point numbers, and execute *SIMD* instructions[6]. This makes it possible to process simultaneously multiple data points within a single instruction. The SHAVEs can be clocked at 600 (0.9V supply) or 800 MHz (0.99V supply). This yields a maximum of 100 GFlops of 32-bit floating point performance. Furthermore, the chip has a special kind of memory, namely *CMX*, which is composed of several smaller *SRAM*[7] blocks. The slides of *CMX* are distributed between the twelve *SHAVE*s and connected via a matrix. Each *SHAVE* has higher bandwidth and lower power access to its "own" local slice. Each SHAVE has also access to both caches: *L1*, which contains 2 KBytes for instruction and 1 KByte for data caching), and the *L2*-cache (256 KB). Additionally, *SHAVE*s are able to communicate with *DDR* memory if the L2-caches are specifically configured. Detailed information about the *Myriad2* can be found by the interested reader in the *"Programmer's Guide of Myriad 2 Development Kit (MDK)"* and Deliverable D3.1 [18].

[4]Graphic Dynamic Random Access Memory

[5]It should be noted that the *Leon* cores support double precision, though.

[6]*Single instruction—multiple data* (SIMD), is a class of parallel computing in Flynn's taxonomy.

[7]In contrast to *DRAM* (dynamic random-access memory), *SRAM* (static random-access memory) does not need to be periodically refreshed. SRAM is faster and more expensive than DRAM.

ISO C/C++ is a supported programming language by the *Myriad2*. Two operating modes are available on the Movidius platforms:

- *bare-metal*: software runs directly on the hardware instead of running on top of an operating system.
- *RTEMS OS*: RTEMS[8] is an open-source real-time operating system [29] under which a program can be run.

Although the performance would probably benefit from the use of bare-metal, we focus primarily on using *RTEMS*. The rationale behind this decision is as follows: Support of the Ethernet interface, dynamic memory allocation and multi-tasking only work under *RTEMS*, which are features that are used extensively in EXCESS framework. The embedded Movidius platform is directly connected to the Ethernet switch of the EXCESS cluster. This makes it possible to integrate the embedded system into the cluster including the EXCESS monitoring framework.

Programming for *RTEMS* is compatible with the programming for the *Linux* operating system. For example, *RTEMS* POSIX API [12] supports a large subset of POSIX 1003.1b [11]. This allows us to reuse most of C source code for both the *x_86* and Movidius platforms. Some exceptions are the programming core functionalities like frequency setting, and architecture-specific features including *SHAVE*s of the Movidius processor *Myriad2* and its cache controlling. The detailed information about the programming environment can be found in the Myriad2 documentation [26].

The Myriad2 Development Kit includes examples to facilitate the use of the platform, and to help with the development of new (multi-media) software. It should be noted that it is critical to get support directly from Movidius for development of software in the field of HPC. We are pleased to announce that we received the needed assistance by employees of Movidius in order to implement numerical algorithms common to HPC for the Myriad2 development board. That way, also Movidius's employees benefited from this new field of applications aside from multi-media and deep learning software.

2.3 Power Measurement

As mentioned above, the various cluster components such as entire nodes, processors, PCIe interfaces and so on, are monitored by the external power measurement system. In contrast to the power measurement of the Movidius hardware, where the measurement shunts and A/D converters are part of the system implemented as a daughter card (see [25]), we use four PCIe cards APCIE-3021-16 [7]. Each PCIe card includes eight analogue input channels and a single A/D converter. These cards are installed in the power measurement system based on the *x_86* server with Linux OS (Kernel 2.6.32). Its services provide wide functionalities covering the dynamic configuration, execution, filtering and analysis of the measurements, which all compose the EXCESS cluster's power tools.

We described the power tools and their usage in Deliverable D5.7 in the section "EXCESS Cluster Power Tools" [14]. Customized drivers, services and a technical description of the power measurement system, based on PCIe-A/D converters, are available in the EXCESS repository at:

[8]RTEMS version 4.10.99 is currently supported by the Myriad2 Development Kit.

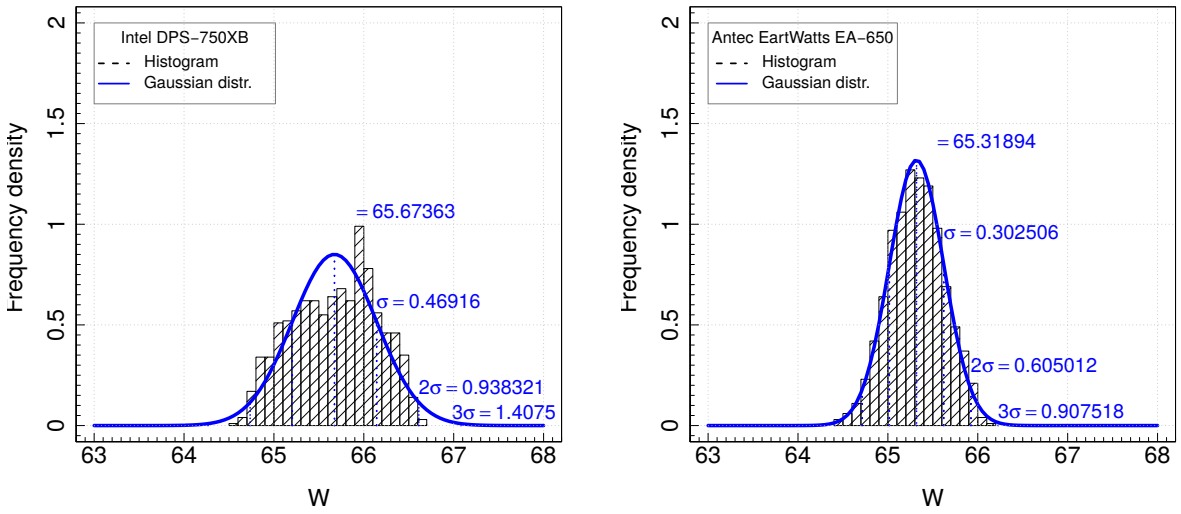


Figure 2: The histograms and density functions of the normal distribution are shown for the electric power of the circuit equipped with RB50-2R2-J 2.2Ω *and power supply Intel DPS-750XB A (on the left side) or Antec EartWatts EA-650 (on the right side). The sample contains a total of* 1000 *measurements of the voltage and electric current over the time. The measuring frequency was* 50.0 kHz.

`http://gitlab.excess-project.eu/excess-tools/apcie3121-driver-and-services`

2.4 Results

Firstly, we defined the set of benchmarks for obtaining information about the efficiency of the hardware and software. The benchmarks and their results serve as examples for both the use of the cluster environment and the analysis of the performance- and energy efficiency of the installed hardware and software.

The measurement system supports the electric power profiling of the hardware components with high accuracy and high frequency with up to 50 kHz. The electrical power of all processors (including the DRAM modules) are measured with a relative error of —$\varepsilon_{rel} = 100 \times \frac{P_{real} - P_{meas}}{P_{real}}$— less than ± 0.1%, where P_{meas} is the measured and P_{real} real power consumption. One can decrease the error to ± 0.01% by applying implemented filters. The latter can be implemented at the cost of reducing the time resolution. The service PTP[9] synchronizes the clocks between the power measurement system and the compute nodes, which hold the drift between the distributed clocks by less than 20 µs (experimentally established). We reserved a dedicated Ethernet interface on each of the nodes for this service (not shown on Figure 1).

[9]The Precision Time Protocol (PTP) is a protocol used to synchronize clocks throughout a computer network.

2.4.1 Power Measurement Accuracy Analysis

Each compute node of the EXCESS cluster receives electrical power from the provided power supply. The integrated VRMs[10] provide the chips with an appropriate supply voltage. Both of the steps involve the work of semiconductor switches such as *MOSFET*s[11]. These are the interference sources, referred to as ripple noise voltage, which limits the accuracy of the power profiling. To test this hypothesis, we compared three different power sources:

- *Server Power Supply* Intel DPS-750XB A REV02F
- *Desktop Power Supply* Antec EartWatts EA-650
- *Lithium polymer battery* LiPo 11.1 V

We connected a power resistor *RB50-2R2-J* 2.2 Ω to the above listed power sources as if it were a conventional processor, although with the difference that the resistor is a passive load. For each of the sources under load, we recorded the voltage and current profiles. The histograms of the calculated power, as product of applied voltage and the electric current for two power supplies, are illustrated in Figure 2. Each of the samples contains a total of 1.000 measurements for voltage and current, which were sampled at a frequency of 50 kHz. Additionally, the normal (or Gaussian) distribution with the adequate parameters is shown. To yield the distribution, the standard deviation σ and mean value μ were calculated for the measured values. As it can be seen from Figure 2, the generated sample for *EartWatts EA-650* comes very close to the normal distribution. In contrast, the *Intel DPS-750XB* generates samples that do not follow a Gaussian distribution.

A possible cause for this could be the modulation technique used for pulse-width modulation (PWM) within the power supplies[12]. This modulation is done through the synchronized switching of the *MOSFET* transistors. The number of transistors and the frequency of the switching, along with other features, affects the cleanness of the recorded signal. Figure 3 shows two oscilloscope profiles of the voltage signal for both of the power supplies under passive load (*RB50-2R2-J*). The amplitude of the *Intel DPS-750XB* signal is about 0.2% of the voltage level, and is four times higher than one of *Antec EartWatts EA-650*. The frequencies are 116 kHz for *Intel DPS-750XB* and 70 kHz for *EartWatts EA-650*. The interested reader can find an extensive study about *Antec EartWatts EA-650*'s features in [27]. Unfortunately, we could not find any detailed information about *Intel DPS-750XB*'s architecture. A book written by Valter Quercioli on "Width Modulated (PWM) Power Supplies" [35] contains detailed information about the functioning of *PWM*s, though.

[10] Voltage Regulator Module
[11] metaloxidesemiconductor field-effect transistor
[12] PWM controls the amount of power delivered to the hardware components.

Figure 4 shows the power measurements when the lithium polymer battery *LiPo 11.1 V* is connected to the power resistor. Additionally, the figure illustrates the recorded power profile. In contrast to a power supply, the battery does not produce a ripple noise voltage. This is clearly visible through the small value of σ, and the thin line of the power profile; the profile contains the measured and filtered values (cf. legend on the diagram).

As mentioned earlier, the power measurement system records both the voltage and electrical current profiles. In the second step, we measured the power consumption of the processor *Intel E5-2690v2* (*Ivy Bridge*) during the idle state—meaning that no computa-

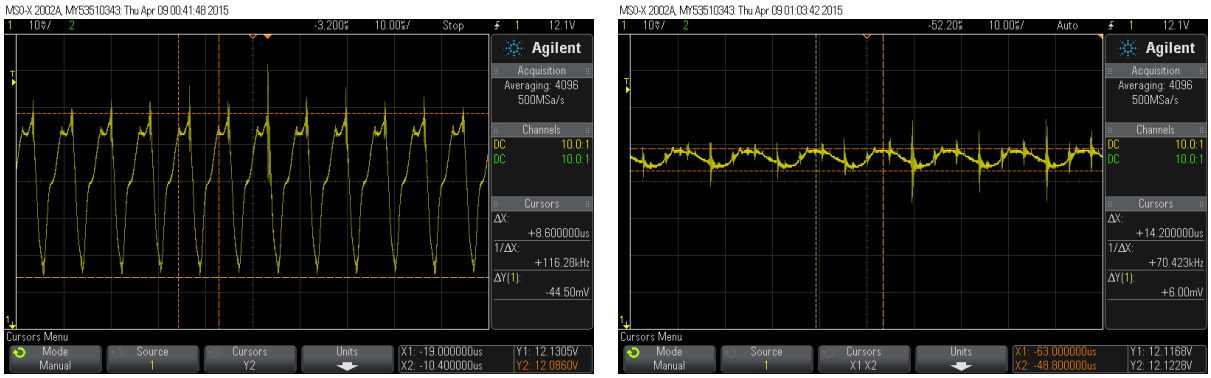


Figure 3: The figures show the oscillograms taken in mode "Average over 4096 samples" at 500 *MSa/s. The change in the voltage over time was measured in parallel to the resistor* RB50-2R2-J 2.2Ω*, which was connected to the +12 VCD output of the power supply Intel DPS-750XB A (on the left side) or Antec EartWatts EA-650 (on the right side).*

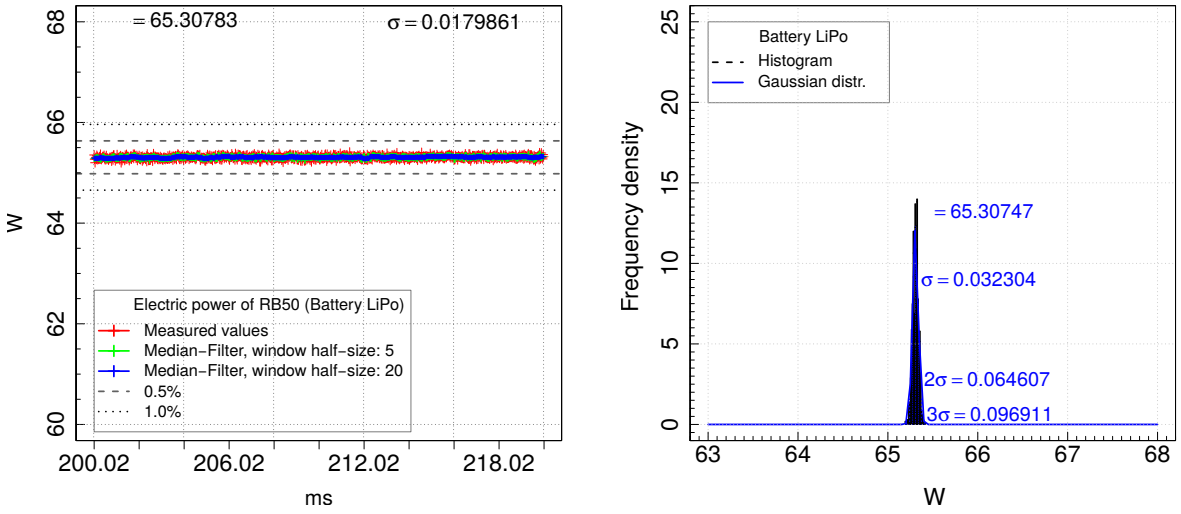


Figure 4: The measured power consumption ($P = V \cdot A$) over time (on the left side) and the histogram and density functions of the normal distribution are shown for the circuit with the resistor RB50-2R2-J 2.2 Ω *and lithium polymer battery* LiPo 11.1 V*. The measuring frequency was* 50.0 kHz*.*

tional job was executed on the compute node. The histogram for 500 voltage measurement for the power supply *Antec EartWatts EA-650* is shown on the left-hand diagram in Figure 6. When the lithium battery is connected to the processor, the standard deviation σ for the measured values is noticeably reduced.

Figure 5 depicts the corresponding histograms for the measured electric current. The results show that the battery, as a power source, affects much less the measurement of the power supply. Opposite to this, the measurement of the processor *E5-2680v3* (Haswell) shows no noticeable differences between the operation with *Antec* and *LiPo 11.1 V*. The histograms showing the latter case are depicted in Figure 7.

In contrast to *Ivy Bridge*, *Haswell* is able to switch to the power management state C6, which powers down the cores, ring and uncore components [10]. However, during the idle state, the operation system services are still active and activate these components at regular intervals (every $\sim$ 1 ms on the EXCESS cluster). This leads to a sharp increase or decrease in the electrical current (cf. Figure 8). Similarly to the voltage measurement, the choice of the power source has no noticeable effect on the measurements. A plausible reason for this may be that the *Haswell* processor includes a fully integrated voltage regulator additionally to the main-board regulator [9]. Both regulators cause additional noise in measurements, which is comparable with the power supply noise. The similarity between the normal distribution and the distribution of the measured values enables us to increase the measurement accuracy by applying the median-filter included in the EXCESS power tools. This works particularly well for the *Haswell* processor and voltage measurements. Further details on the power measurement for various hardware components of the EXCESS compute nodes in idle state are available in the Deliverable D5.4 [22].

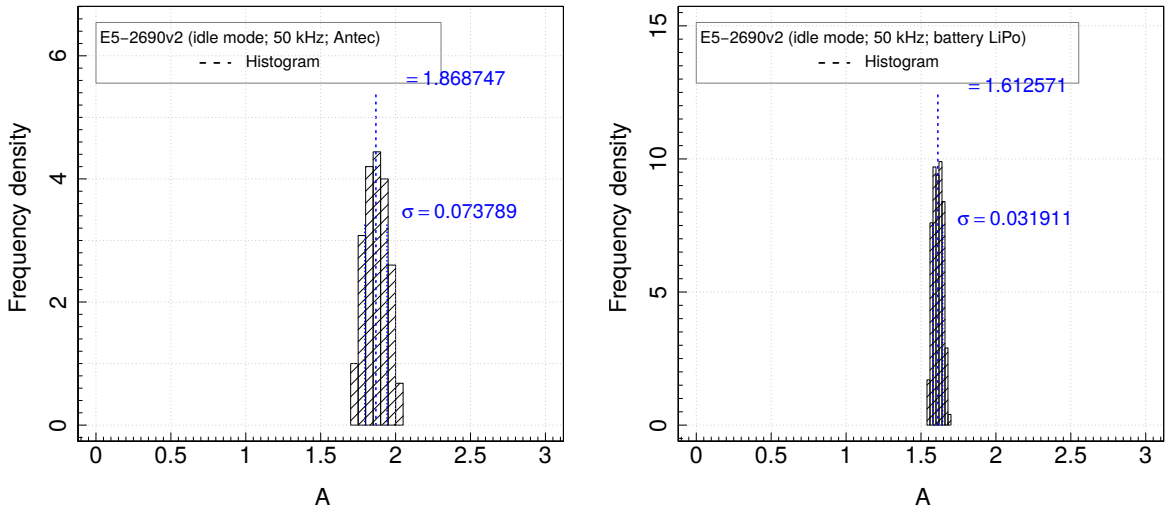


Figure 5: The histograms show the dispersion of 500 electric current measurements in the circuit of the first processor Intel E5-2690v2 *(*Ivy Bridge*) and the power supply Antec EartWatts or Intel DPS-750XB A. During the measurement, there was no computational jobs on the compute node. The measuring frequency is set to* 50.0 kHz.

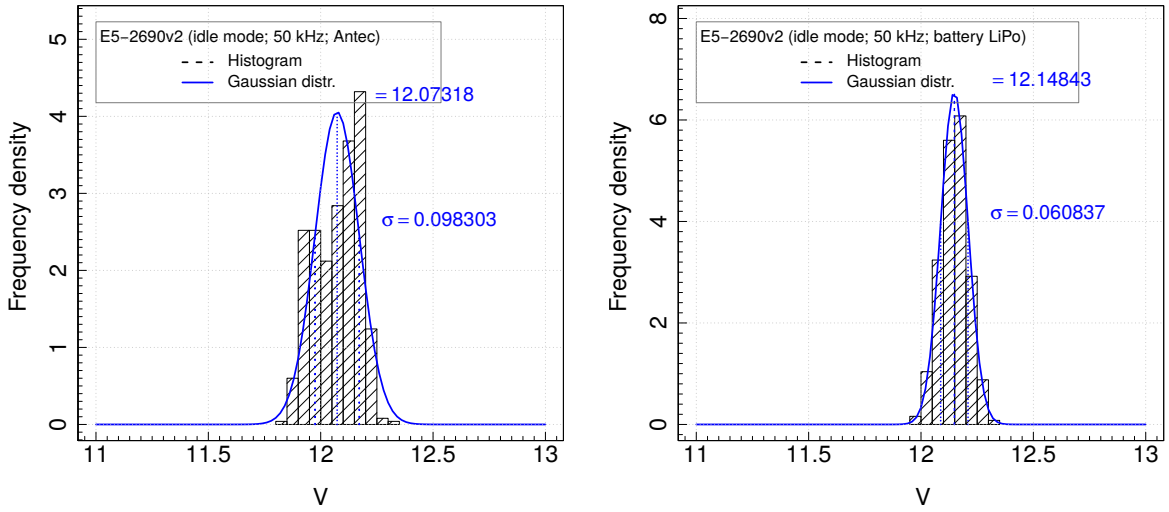


Figure 6: The histograms show the dispersion of 500 voltage measurements in the circuit of the first processor Intel E5-2690v2 *(*Ivy Bridge*) and the power supply Antec EartWatts or Intel DPS-750XB A. During the measurement, no computational jobs where running on the compute node. The measuring frequency is set to* 50.0 kHz.

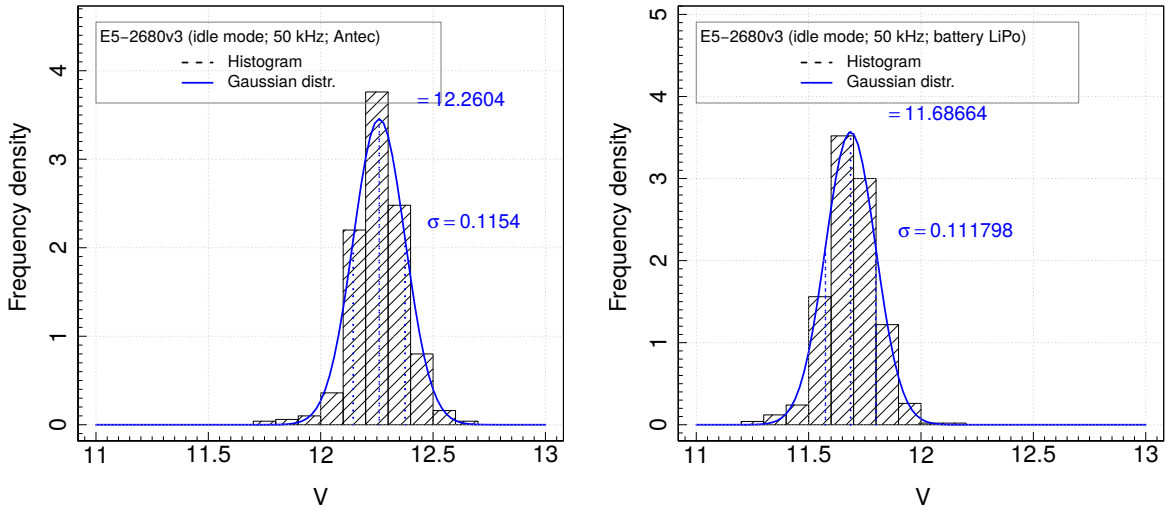


Figure 7: The histograms show the dispersion of 500 voltage measurements in the circuit of the first processor Intel E5-2680v3 *(*Haswell*) of a compute node and the power supply Antec EartWatts or Intel DPS-750XB A. During the measurement, there was no computational jobs on the compute node. The measuring frequency was* 50.0 kHz.

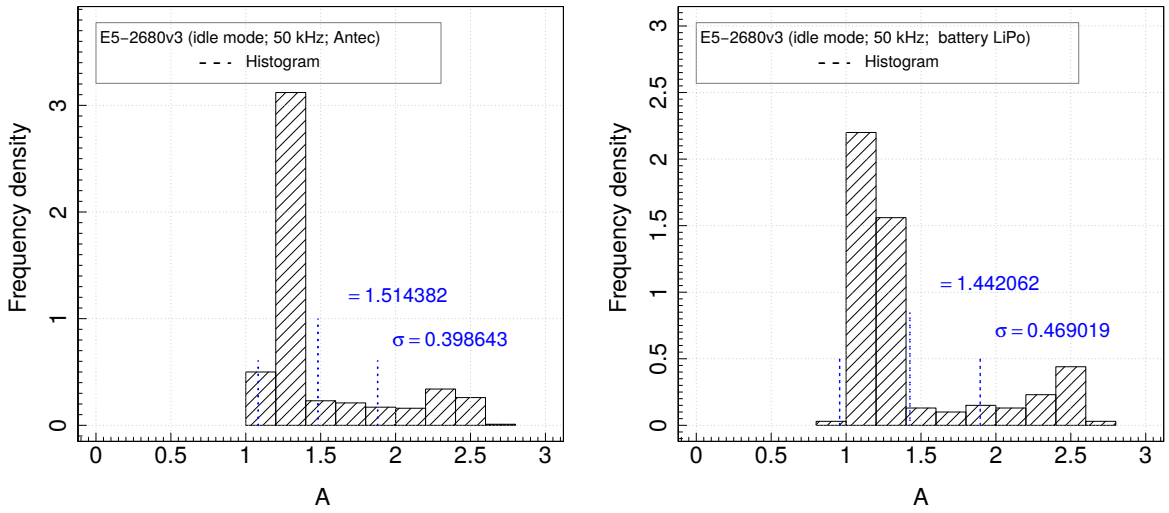


Figure 8: The histograms show the dispersion of 500 electric current measurements in the circuit of the first processor Intel E5-2680v3 *(*Haswell*) of a compute node and the power supply Antec EartWatts or lithium polymer battery* LiPo 11.1 V*. During the measurement, no computational jobs were executed on the compute node. The measuring frequency was set to* 50.0 kHz

.

2.4.2 Clock Synchronization Analysis

Figure 9 shows the power profile of the first processor *E5-2680v3* (Haswell) on node03 during the execution of the kernel operation *ADD*, $a[i] = b[i] + c[i]; 0 < i < 57664$, on each of the 12 cores. The double-precision floating-point arrays fit into the L3-cache. The loop was repeated 30 times. The test is scheduled as follows:

1. Start 12 OpenMP Threads
2. Refill the cache with the arrays
3. Start the PAPI counters (e.g. RAPL, UNHALTED_CORE_CYCLES)
4. Read the timestamp
5. Read the TSC register[13]
6. Compute the kernel *ADD* 30 times
7. Read the TSC register
8. Read the timestamp
9. Save metrics and go to the next test

Two determined timestamps are shown in Figure 9 with two vertical dotted lines. The difference between these two timestamps is 2.122 ms. The duration time, which was measured with TSC, equals to 2.029 ms. This results in a time-synchronization error, and instrumentation overhead of 0.093 ms. The median-filter was applied to the measured values. The result of the filtering step, the standard deviation σ, and the mean value μ are also depicted in the figure. At the beginning of the computation, the power increases within 0.2 ms and then stays stable until the end of the execution.

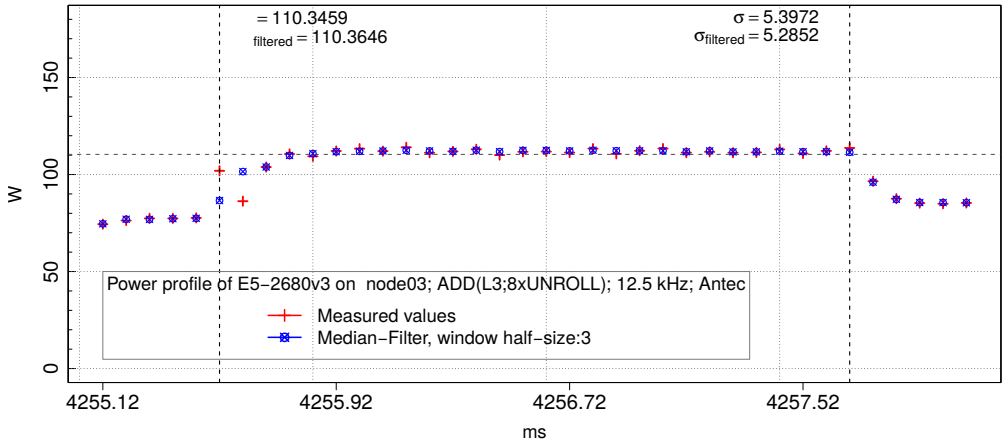


Figure 9: The diagram shows the power profile of the first processor E5-2680v3 *(Haswell) on node03 during the execution of a kernel operation* ADD $a[i] = b[i] + c[i]; 0 < i < 57664$ *on each of 12 cores.*

[13]The Time Stamp Counter (TSC) register counts the number of cycles since reset. It provides a high-resolution, low-overhead way to measure the time.

2.4.3 Comparison between RAPL and A/D Converters

The power stream benchmark contains various kernel operations including the previously mentioned kernel *ADD*. During the benchmark, the kernel is started with various configurations, which include all combinations of the possible CPU frequencies and the number of active cores (see Deliverable D3.4 [15] for further details).

The results of the power measurement system were compared with the results of the RAPL counters for all configurations. The RAPL counters of the Ivy Bridge and Haswell processors show less power consumption than the measurements with the A/D converters. This difference grows with the increase of the measured power, and can be approximated with $K(P_{rapl}) = \alpha_0 + \alpha_1 P_{rapl}$, where P_{rapl} is the obtained RAPL value.

Hence, the results—obtained through the power measurement system $P_{a/d}$—can be approximated with the equation $P_{a/d} = P_{a/d} + K(P_{rapl}) + \epsilon_{P_{rapl+K}}$, where $\epsilon_{P_{rapl+K}}$ is the error of the approximation. The comparison for the kernel *ADD*, if the data fits RAM: $a[i] = b[i] + c[i]; 0 < i < 42949664$ on each of cores, is plotted on Figure 10. Additionally, the upper limits of the error $\epsilon_{P_{rapl}} = P_{a/d} - P_{rapl}$ are specified.

Table 5 contains the approximation of the difference between the measurements done with RAPL counters and A/D converters for various configurations of the kernel operation *ADD*.

We have also compared the RAPL measurements with other kernels. The observed differences were within the limits that were calculated for the kernel *ADD*. It should be noted that the RAPL counters do not consider the power consumption of the board's *VRM*. Another property is that the differences between the RAPL and A/D converter measurements are approximately linear for the same kernel both as a function of the CPU frequency while the number of active cores is fixed, and as a function of the number of

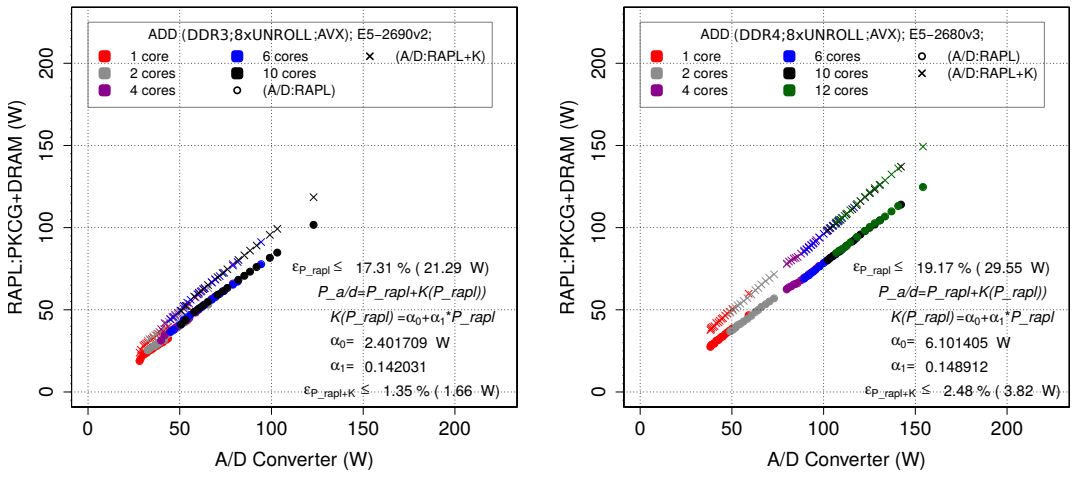


Figure 10: Comparison between RAPL counters and the EXCESS power monitoring system based on A/D converters. The values on the diagrams are obtained by the power measurements with the RAPL counters and A/D converters for all combinations of the number of threads and CPU frequencies during the execution of the kernel operation ADD*, if the data fits into the RAM:* $a[i] = b[i] + c[i]; 0 < i < 42949664$ *on each core.*

active cores while the CPU frequency is constant.

2.5 Integration of Power Measurement System

The cluster environment provides two methods for the power profiling, which can be divided in two main groups:

High-Resolution Mode All measured values are stored on a local hard disk on the power measurement system. The values for each metric (voltage V and electric current A) are stored in chains, which contain the measurements for 1 second; the interval is configurable. Each of the individual chains is stored in an unformatted file, which hides the single precision floating point numbers. The timestamp of the first measurement in the chain is also stored in a dedicated file. After the measurement is stopped, all data is archived and copied to a shared folder owned by a user. The power measurement tools were specially designed to handle and analyse such data.

Runtime Mode The power measurement system, parallel to the measurements, calculates the average power consumption of the hardware components, and sends the consumption to the database of the EXCESS monitoring framework. The transaction will be initiated when new incoming chains with the measurements for the voltage and electric current are ready. The time interval for the average values is also configurable. Currently, the system sends values averaged over 100 ms.

The *High-resolution* and *Runtime* modes can be used together in any combination. The power measurement of the compute nodes is switched on automatically at the beginning of a job. At the end of a job, the power measurement will be finished. Alternatively, the measurement can be started and finished manually.

2.6 Conclusion

The EXCESS cluster provides an execution environment together with the EXCESS power monitoring tools. The tools are implemented with respect to usability, accuracy and scalability. The current configuration of the system supports the simultaneous operation of a maximum of 32 measurement channels running at 12.5 kHz. It should be

Table 5: Differences between the measurements done with RAPL counters and A/D converters for various configurations of the kernel operation ADD.

CPU	Kernel	$K(P_{rapl})$	$\epsilon_{P_{rapl+K}}$	$\epsilon_{P_{rapl}}$
E5-2690v2 (Ivy Bridge)	ADD;L1	$1.460\,56 + 0.212\,115 \cdot P_{rapl}$	$< 2.22\%(2.72\text{W})$	$\leq 24.5\%(29.98\text{W})$
E5-2690v2 (Ivy Bridge)	ADD;L2	$5.137\,535 + 0.002\,18 \cdot P_{rapl}$	$< 0.76\%(0.91\text{W})$	$\leq 5.11\%(6.17\text{W})$
E5-2690v2 (Ivy Bridge)	ADD;L3	$3.414\,631 + 0.112\,679 \cdot P_{rapl}$	$< 0.73\%(0.97\text{W})$	$\leq 14.81\%(19.63\text{W})$
E5-2690v2 (Ivy Bridge)	ADD;DDR3	$2.401\,709 + 0.142\,031 \cdot P_{rapl}$	$< 1.35\%(1.66\text{W})$	$\leq 17.31\%(21.29\text{W})$
E5-2680v3 (Haswell)	ADD;L1	$3.407\,782 + 0.144\,827 \cdot P_{rapl}$	$< 1.02\%(1.19\text{W})$	$\leq 17.49\%(20.34\text{W})$
E5-2680v3 (Haswell)	ADD;L2	$4.302\,786 + 0.125\,509 \cdot P_{rapl}$	$\leq 0.84\%(0.9\text{W})$	$\leq 17.32\%(18.68\text{W})$
E5-2680v3 (Haswell)	ADD;L3	$3.291\,371 + 0.143\,92 \cdot P_{rapl}$	$\leq 1.7\%(2.31\text{W})$	$\leq 17.37\%(23.67\text{W})$
E5-2680v3 (Haswell)	ADD;DDR4	$6.101\,405 + 0.148\,912 \cdot P_{rapl}$	$\leq 2.48\%(3.82\text{W})$	$\leq 19.17\%(29.55\text{W})$

noted that the number of the measurement channels can be at least doubled. However, there are two disadvantages of such a system, which can be probably eliminated.

The first disadvantage is the measurement of the processors together with the memory modules. The reason for this is that the same power connector is used for both the processor and memory. Probably the most reliable method to distinguish the CPU power from the power of the RAM is using so-called *DDR Riser Cards*: the measurement shunts, which are integrated in these cards, can be connected to the measurement channels of the system. However, these cards could affect the memory frequency in a negative way.

The second disadvantage comes from the fact that the power measurement and the progress of the program execution are synchronized with the help of timestamps. Currently, a user has to add manually additional instructions to store the timestamps of the various phases of the program. These timestamps and measurement values can then be used as input data for the power tools to create a detailed analysis of the program. In this context, we aim to write a plug-in for the visualization performance tool *Vampir* [28]. *Vampir* provides various visualization techniques to show the collected performance and power data. Then, the user can use automatic instrumentation provided by several trace tools including *Score-P trace* [32] or *Vampir trace* [28].

3 Evaluation Results

This section discusses the results of benchmarks, which were developed or extended, respectively, throughout the course of the EXCESS project. Firstly, we show the results of a benchmark, which measures the performance of the PCIe version 3.0 interface. The interface is used to connect the accelerators to the host CPU. Secondly, we highlight the main results obtained of analysing the kernel called *XGEMM*. This kernel operation performs a matrix-matrix multiplication, and is included in any Level 3 *BLAS* library [36]. *XGEMM* is defined as

$$C = \alpha * A \times B + \beta * C,$$

where A, B and C are matrices of sizes $m \times k$, $k \times n$, $m \times n$, respectively, α and β are of scalar type. We conclude this section with information about the execution of two benchmarks on the *Movidius* embedded platform.

3.1 PCIe Latency Bandwidth

Current accelerators are connected to the host CPU through the PCIe version 3.0 interface. That interconnect type has three big disadvantages: higher latency, low bandwidth and no support of the cache coherence protocol used by the main memory. While the lack of cache-coherency adds additional complexity to develop software, the higher latency and the low bandwidth limits the overall usage of accelerators for real-world applications—this is in particular true for the domain of distributed computing, which is of high importance in HPC. Figure 11 shows, for example, that the maximal bandwidth can only be achieved for data sizes bigger than 4 MBytes. If we consider that a typical dedicated graphics card has only a few Gigabytes of memory available, the latency and the bandwidth will be dominated by the actual computation time. The kernel start-up time is an additional factor, which can slowdown applications, particularly in the case of small local problem sizes. To be fair, one also has to remember that the initial costs of the compute nodes with the accelerators much higher than without these. The heterogeneous *CPU-GPU* cluster waste unnecessary energy in *"CPU"* as well in *"GPU"* exclusive modes. The amount of resources that are wasted are shown in the results of *XGEMM* benchmark for the processors, which is described further down in this document.

However, GPUs can be very efficient. This is in particular true for large data sizes; the results in Section 3.2.2 establish this hypothesis.

3.2 Matrix Matrix Multiplication (XGEMM)

In order to perform a pure matrix-matrix operation, we set the coefficient β to `null`. Additionally, we consider only the square matrices ($m = k = n$). It should be noted that any highly-optimised *BLAS* implementation should include this special case to avoid two unnecessary operations: $*$ and $+$. Hence, depending on the size of the matrices, there are effectively $mn(2k - 1)) \approx 2mnk$ floating point operations to be performed by the matrix-matrix multiplication.

The first letter X means that *BLAS* supports both double-precision (*DGEMM*) and single-precision (*SGEMM*) floating point numbers. The performance of *XGEMM* on the EXCESS hardware is measured by using the metric *GFlops*, which is defined as billion

floating point operations per second. Additionally, we measured the electric power of the hardware components in order to calculate the energy quantities consumed by the matrix-matrix multiplication. An important metric for assessing the energy efficiency is defined as *Joule/GFlop*, which indicates how much energy in Joules is needed to perform the billion floating point operations.

Joule/GFlop can be calculated by dividing the average electric power in Watt by the peak performance during the benchmark execution[14]. Most hardware vendors implement their own *BLAS* library to demonstrate, amongst others, the peak performance and other characteristics such as power consumption under full load of the hardware. The assessment of sustained performance on the basis of benchmarks of this kind can therefore only be taken into account for a particular group of applications including *CNN* and *HPL*[15] [23].

3.2.1 XGEMM on Intel Processors

Figures 12 show the performance of the operation *DGEMM* on *Ivy Bridge* and *Haswell* processors. The performance increased in stages on both processors:

- *Ivy Bridge* from 1.239 (matrix size is 64×64) up to 110.592 *GFlops* (matrix size 1,472×1,472), from 145.806 (1536×1536) up to 155.074 *GFlops* (2240×2240) and from 234.931 (2,272×2,272) up to 234.504 *GFlops* (4,096×4,096). The peak performance is achieved with the matrix size 4,032×4,032 and equals to 240.536 *GFlops*.

- *Haswell* from 2.381812 (matrix size is 64×64) up to 255.183 Gflops (matrix size 1,472×1,472), from 255.342 (1,536×1,536) up to 248.123 (2,048×2,048), from 389.4416 (2,112×2,112) up to 402.327 (4,096×4,096). The best performance is also achieved with the matrix size 4,032×4,032 and is 426.629 *GFlops*.

[14] 1 J=1 VAs = 1 Ws => *W/(GFlop/s)=Joule/GFlop*

[15] A portable implementation of the high performance Linpack benchmark for distributed-memory computers (HPL) is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. The Top500 Linpack benchmark [34] is based on *HPL*.

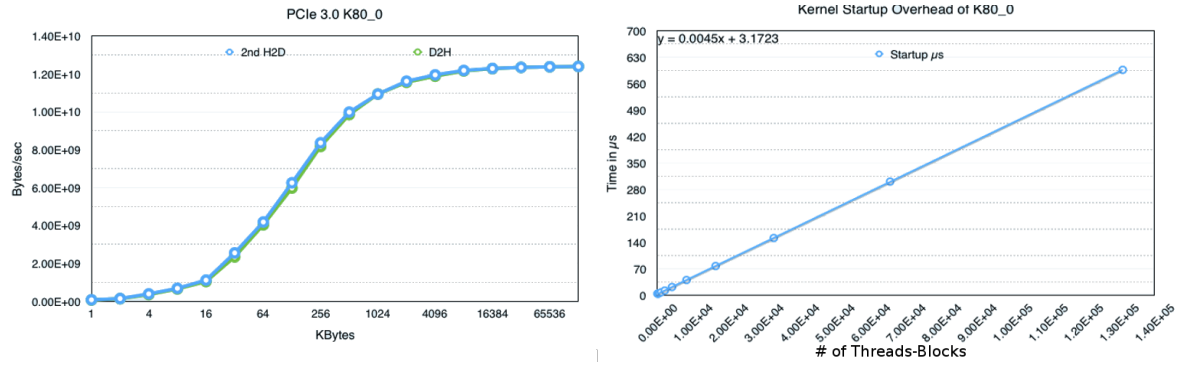


Figure 11: The left diagram shows the PCIe 3.0 bandwidth for data moving between the host CPU system (E5-2690v3) and NVIDIA Tesla K80. The right diagram shows the kernel start-up time on the NVIDIA Tesla K80.

It can be seen that the *FMA* in Haswell almost doubles the performance. Another important observation is that a matrix must be large enough (here 4,032×4,032) to achieve best results.

Figure 13 shows the visualization of electric power changes during the matrix-matrix operation. The multiplication was performed at least 10s for each matrix size; a two second pause is introduced between tests. Although the multiplication was only performed on one processor, the power consumption of the second CPU increases due to the cache coherence protocol.

In contrast to Figure 13, Figure 14 shows the average power consumption for each test. We can see that the power consumption of the first CPU of *node01* increased from ∼ 115 W up to ∼ 146 W. We assume that the main reason for this is that different cache levels are activated depending on the matrix size. The observed power consumption correlates very closely to the power consumption of the benchmark *Power Stream* (cf. Deliverable [15] for further details).

Moreover, the diagrams show the power consumption of other compute nodes components with the result that the electric power of *Haswell* is higher than of *Ivy Bridge*. This could be explained by more cores and more memory of *Haswell*.

Figure 15 indicates how much energy in Joules is needed to perform a billion floating points operations. Due to a better performance, *Haswell* needs less energy (0.37 *Joule/GFlop* for a matrix size of 4,032x4032) than *Ivy Bridge* (0.61 *Joule/GFlop* for a matrix size of 4,032x4,032). For the small matrices (64x64), the energy costs are equal to 93.63 *Joule/GFlop* for *Ivy Bridge* and 56.60 *Haswell*.

The above presented results were obtained with double-precision floating point numbers. *SGEMM* performs a single-precision matrix-matrix multiplication, though. We have executed *SGEMM* for the same matrix sizes as for *DGEMM* on the processor *Haswell*. Figure 16 illustrates the results of our experiments. The left diagram depicts performance and the right one energy costs. The lowest performance was achieved with the smallest matrix size 64x64, which is 1.455 *GFlops*, and results in energy costs of

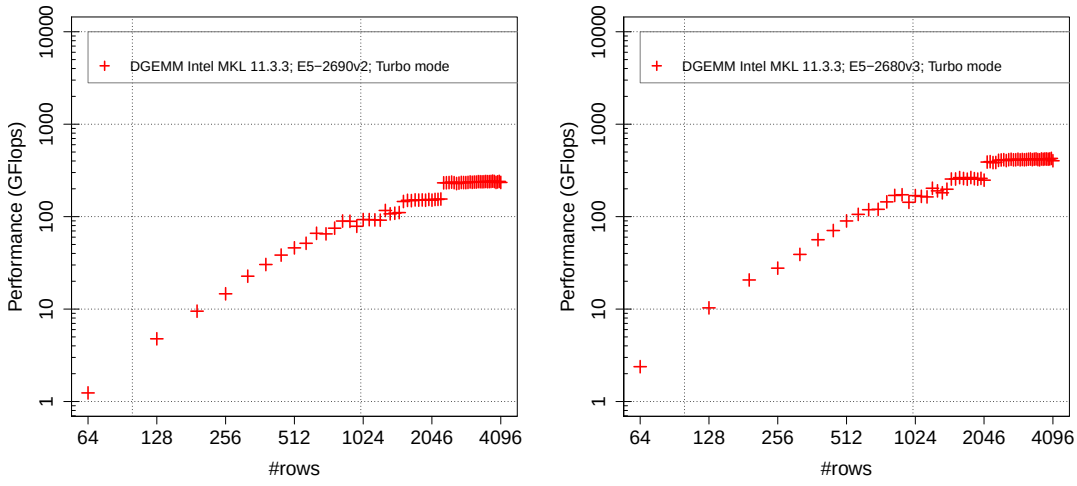


Figure 12: Performance of DGEMM on E5-2690 v2 Ivy Bridge *(left) and on E5-2680 v3* Haswell *(right).*

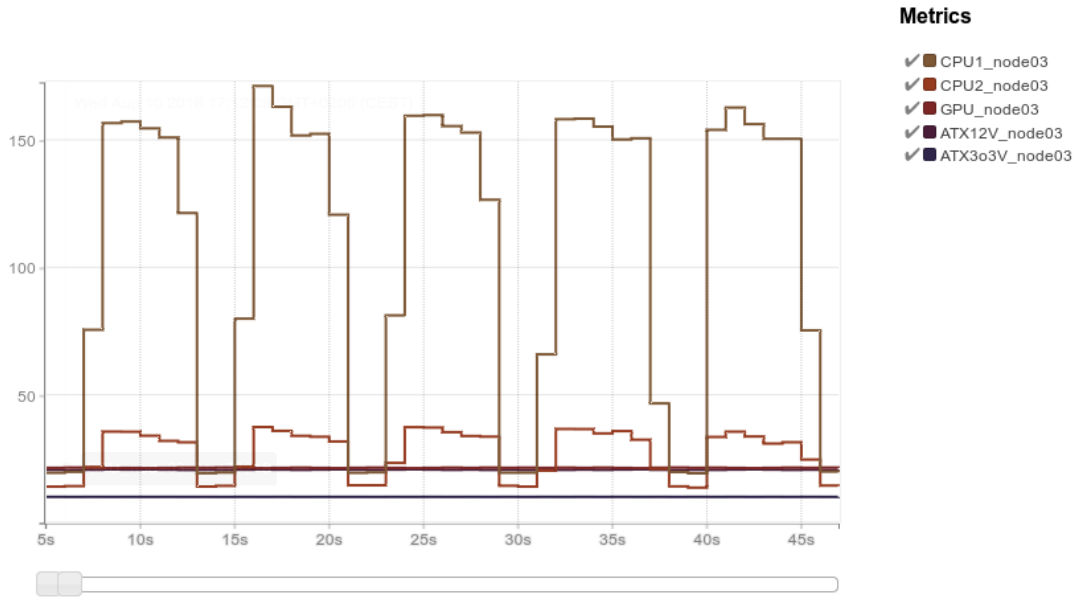


Figure 13: ATOM *power profile for DGEMM E5-2690 v2* Ivy Bridge *(left) and E5-2680 v3* Haswell *(right).*

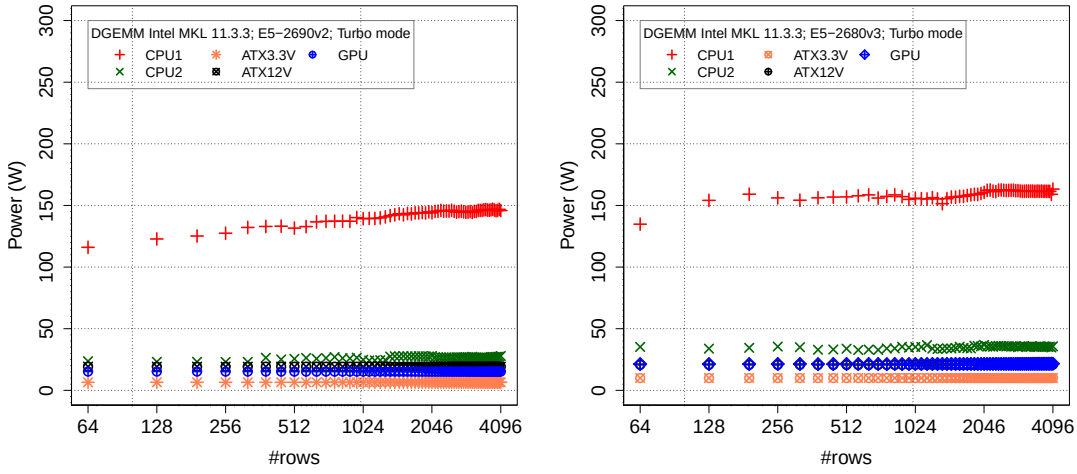


Figure 14: Average electric power of DGEMM on E5-2690 v2 Ivy Bridge *(left) and E5-2680 v3* Haswell *(right).*

83.227 *Joule/GFlop*. Peak performance yielded 1,110.578 *GFlops* while using a matrix size of 3,840) with the energy costs of 0.151 *Joule/GFlop*.

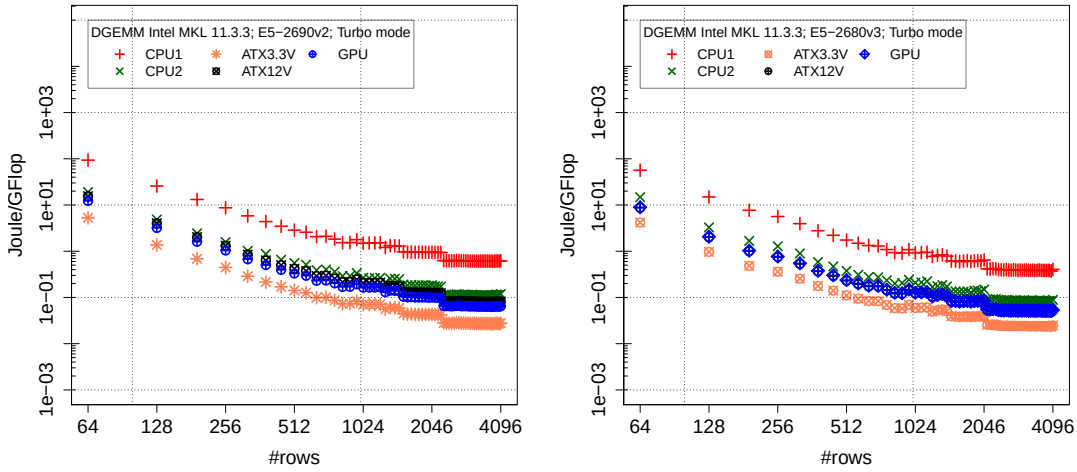


Figure 15: Energy costs of DGEMM E5-2690 v2 Ivy Bridge *(left) and E5-2680 v3* Haswell *(right).*

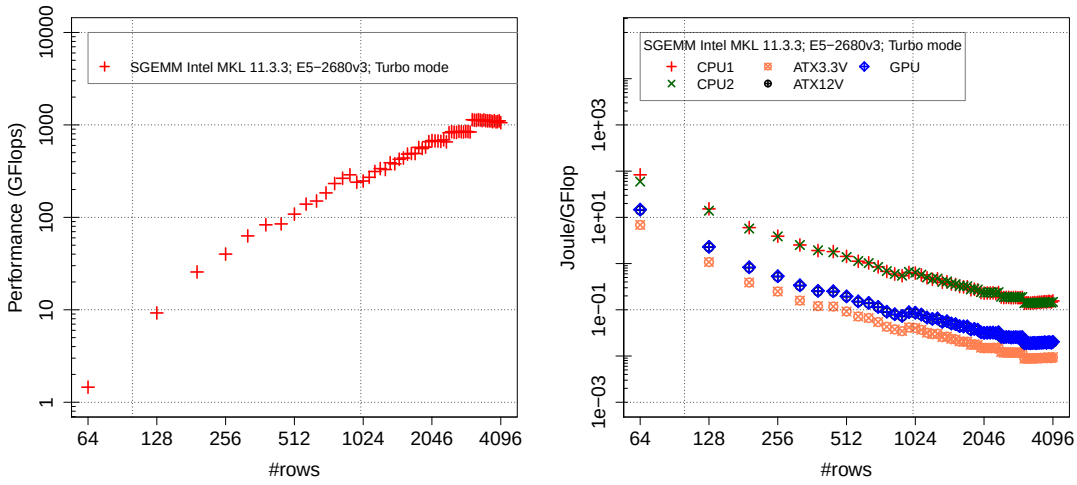


Figure 16: Performance (left) and energy costs of SGEMM on E5-2680 v3 Haswell *(right).*

3.2.2 XGEMM on Tesla K80

The left diagram on Figure 17 illustrates the performance of the operation *DGEMM* on *NVIDIA Tesla K80* Accelerator. In contrast to the results for the processors, the highest performance is achieved by the matrix size 640x640 and is located at 979.076 *GFlops*. The highest performance is achieved with the matrix size 896x896, which results in 985.949 *GFlops*. That is 230% over the performance of *Haswell*.
The right diagram on Figure 17 indicates how much energy is needed by compute node components during the computation. To calculate the power consumption of the *GPU*, we need to consider the power consumption of various components including *ATX12V*, *ATX3.3V* and *GPU* (see Deliverable D5.4 [22] for further details). The result is that *Tesla K80* needs only 0.182 J to perform one billion floating point operations for matrix size 640x640. The energy efficiency decreases for small matrices, though: 5.136 *Joule/GFlop* and 1.055 *Joule/GFlop* for the matrices 64x64 and 128x128. Again, the results are much better than compared to *Haswell*. *Haswell* consumed 56.601 *Joule/GFlop* and *14.9508427 Joule/GFlop*, respectively, for the same matrices.

Figure 18 depicts the electric power changes during the matrix-matrix operation. All components need more power, if the *GPU* starts computation. Figure 19 illustrates the average power consumption of the compute node components.

The above presented results were obtained using double-precision floating point numbers. Figure 20 shows the performance and energy costs for the single-precision matrix-matrix multiplication, instead. The performance increases from 29.435 *GFlops* (64x64) to 2,260.350 *GFlops* (1,152x1,152); while energy costs decrease from 3.405 *Joule/GFlop* to 0.0785 *Joule/GFlop*, respectively.

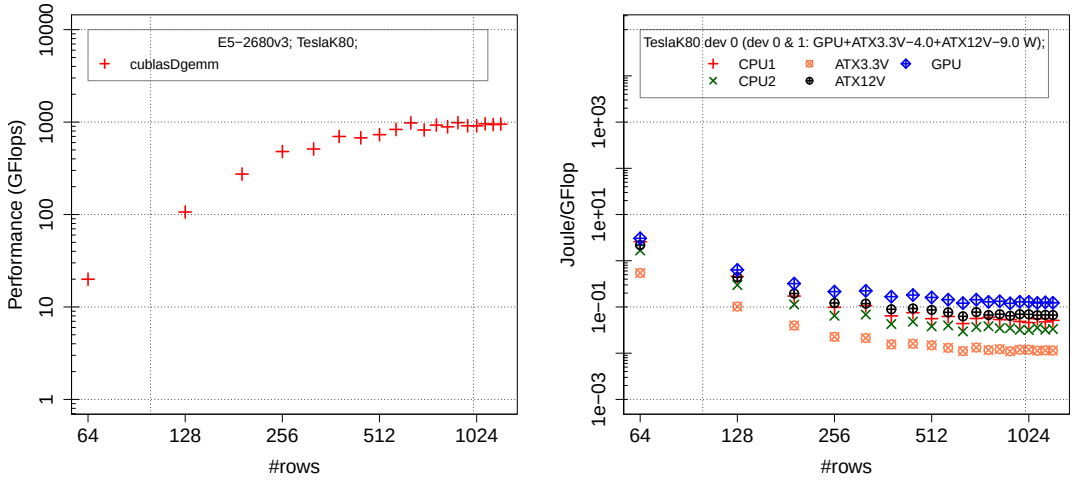


Figure 17: Performance of the benchmark DGEMM on Tesla K80 (left) and energy (right).

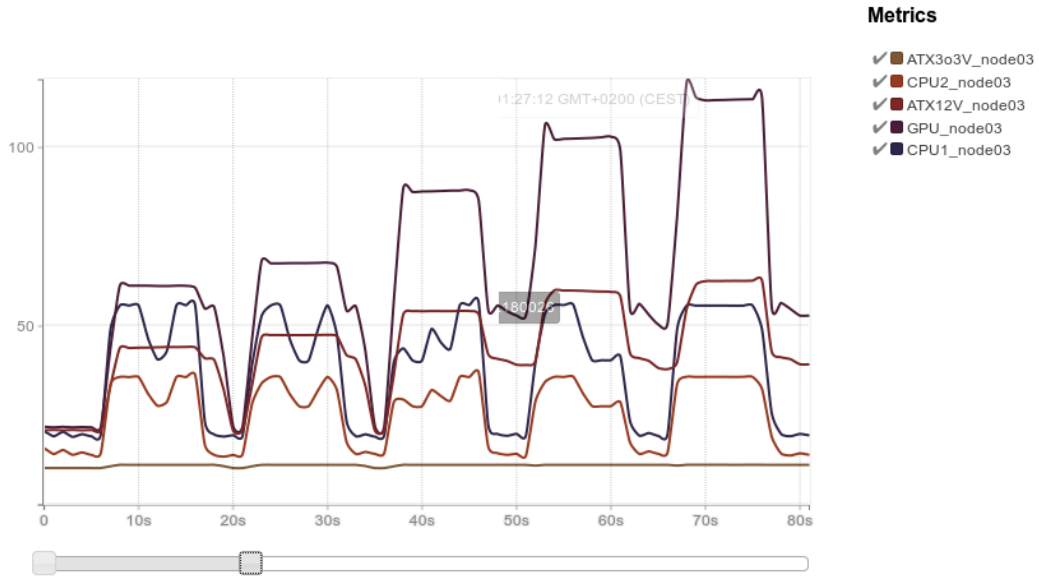


Figure 18: ATOM *power profile of DGEMM on* Tesla K80.

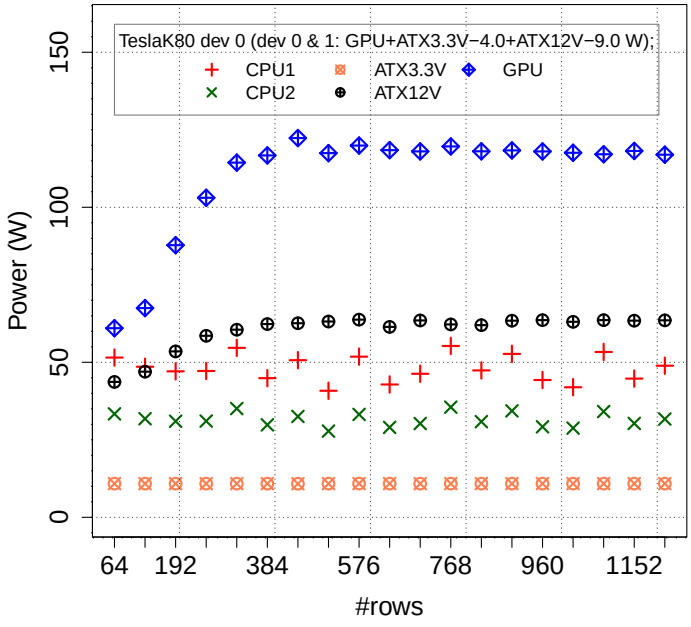


Figure 19: Performance of the benchmark DGEMM (left) and its electric power consumption on Tesla K80 *(right).*

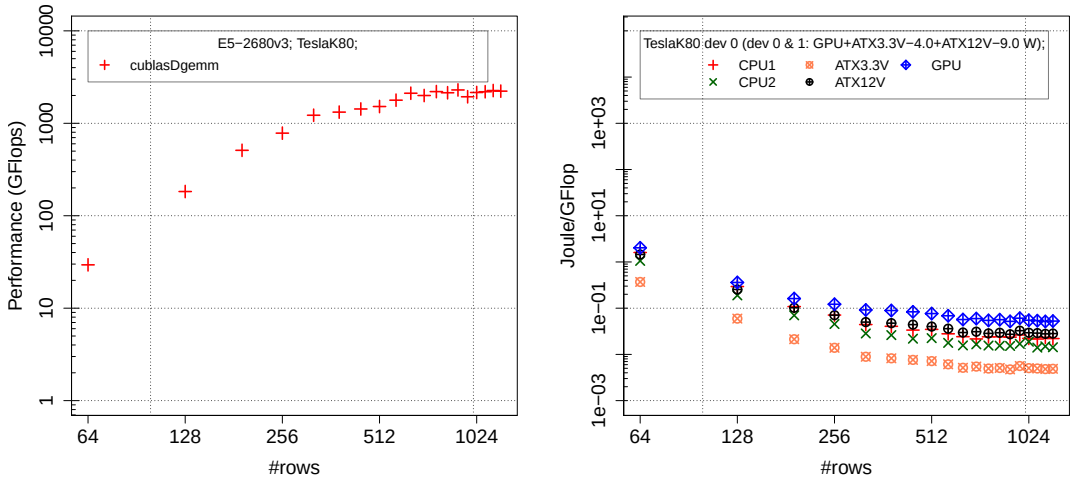


Figure 20: Performance of benchmark SGEMM (left) and its electric power consumption on Tesla K80 *(right).*

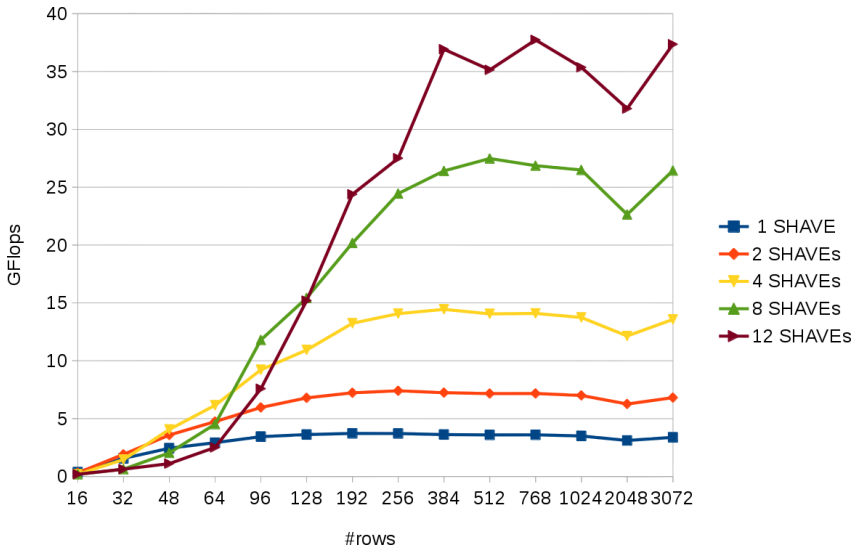


Figure 21: Performance of the benchmark SGEMM on Myriad2.

3.2.3 SGEMM on Myriad2

Unfortunately, *Myriad2* supports only partly double-precision floating point calculations. While *Leon*'s core provides double-precision arithmetic, *SHAVE*'s does not. For this reason, we can only present results for the single-precision case; they are shown on Figure 21. It can be seen that the operations scale with the number of SHAVEs, if the matrix is large enough. The maximal performance for a matrix of size 64x64 is 6.15 *GFlops*. This is 423% higher than the performance of *Haswell*, and almost five times lower than the results of *Tesla K80*. As expected, the performance for the larger matrices is lower than the performance achieved by CPU and GPU (see further results in Deliverable [8]). The interested reader can find a detailed description of the implementation in the paper [17].

3.3 Movidius Ethernet Bandwidth

The Movidius compute card prototype is equipped with a 1 GBytes Ethernet interface to enable network communication. In the first tests, we could not get the appropriated results for the messages larger than 16 KBytes. After several transactions, the bandwidth dropped rapidly down to a few KBytes (see Deliverable 5.6 [21]). We identified an issue with the Movidius driver. The cause was a faulty cache management. The issue was reported to the Movidius support, but the latest Movidius MDK (version 16.08.5) provides only a partial solution to the problem: The issue still occurs, but less frequently. We could achieve a bandwidth of approximately 100 *MBytes*/s using the latest MDK. During our testing phase, the power consumption of the *Myriad2* was around 0.403 W.

3.4 Sparse Linear Solver on Movidius

The sparse linear algebra algorithms, particularly implemented for Movidius platforms, are implemented on both *Leon* and *SHAVE*. The benchmark is based on the *Conjugate Gradient method* (CG) to solve sparse linear systems. For the evaluation, we have utilized sparse linear systems from the application *Leg Implant Simulations* (see Deliverable D5.6 [21] for further details). We have defined those sparse linear systems in the header files of the sources, which gave us the possibility to configure the memory for data storage; the data is directly initialized in CMX memory. The sparse matrix of size 711x711 contains 70,000 non-zero elements. The average performance of the CG method on *Leon* core was 17.977 MFlops with energy costs of $\approx$ 18 *Joule/GFlop* for single-precision. The achieved memory bandwidth was 27.28 MB/s. This is about two times higher than the energy costs of the computation on *Ivy Bridge* (see [30] for further details).

To improve the performance, we have implemented an improved version of the sparse linear solver, which uses SHAVEs of *Myriad2*. Unfortunately, we could not use the L2-cache due to the some issues in the Movidius MDK that was available at the time of performing the experiments. However, the latest version of the MDK, 16.08.5, provides required fixes, and thus we aim to support the L2-cache in the future. Without using L2-cache, the performance was about 6 MFlops.

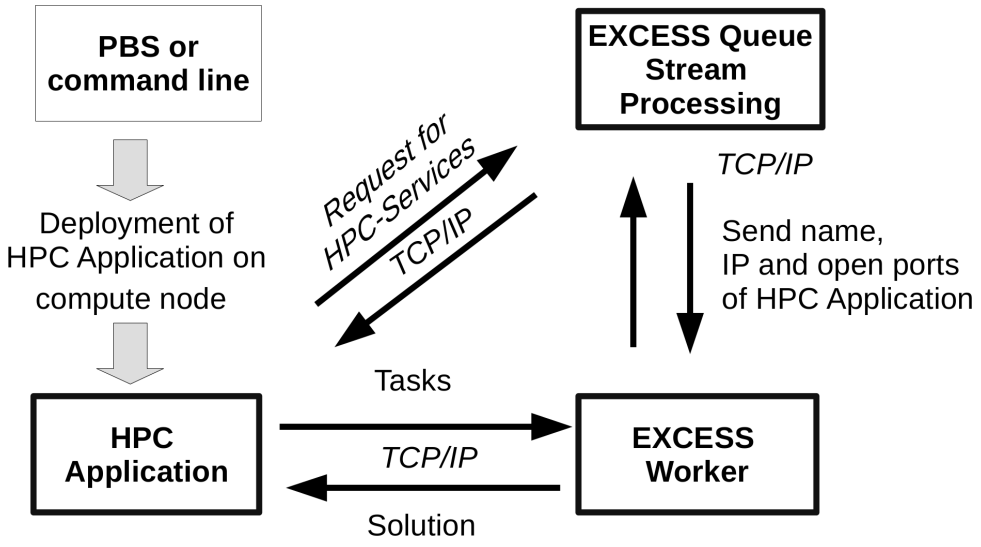


Figure 22

4 Case Studies

4.1 EXCESS HPC-Services and Leg Implant Simulation

EXCESS HPC-Services is an extension to the PBS[16] system, where compute-intensive tasks of an HPC application can be solved aside from the assigned compute nodes of a submitted PBS job. Figure 22 illustrates a typical scenario of the such a workflow.

Due to the focus on the Movidius embedded platform—equipped with Ethernet and USB interfaces—, and in the interests of uniform interface and scalability, we have chosen the Ethernet network interface as the main communication channel between the components of EXCESS HPC-Services.

The prototype of EXCESS HPC-Services consists of the following main components:

- *Application:* We selected a leg implant simulation [31], which also includes Cancellous bone simulation (see [37]), as our main application. The purpose of the leg implant simulation is to understand the structure of the cancellous, or spongy, parts of the human bone, in order to efficiently and safely attach the required implants to it for the medical treatment of certain pathologies. The cancellous bone simulation computes the stiffness matrices for various bone structures, which are provided by micro CT scan[17] of small piece of bone, for example $1 \times 1 \times 1$ cm. These computed elastic coefficients will be mapped to the hospital CT of the patient's bone depending on various parameters, for example the geometry, porousness. This makes it possible to carry out the patient-specific simulations. Using this simulation, implants can be produced in a more accurate fashion, which increases production yield and at the same time, improves the healing time for bone implant surgeries and improves the lifespan of the implant as well. All these advances will

[16]Portable Batch System is a resource manager providing control over batch jobs and compute nodes.

[17]Micro computed tomography or "micro-CT" is x-ray imaging in 3D, by the same method used in hospital CT scans, but on a small scale with increased resolution. X-ray intensities of micro-CT are very strong, and can not be applied on patients.

improve at the end the patients quality of life. The leg implant simulation bone simulations require a lot of computational resources (processors, memory, storage, network, etc.), however, typically hospitals cannot afford HPC systems for specific simulations. Hospitals run their simulations on HPC systems from third parties. In this use case simulations run in the Supercomputing Center of Stuttgart University (HLRS - www.hlrs.de).
The simulation is highly-representative for a compute-intensive application that is executed on HPC platforms. The simulation serves as a *producer* of the computational tasks in our EXCESS HPC-Services scenario.

- *Worker:* The *worker* receives tasks produced by the simulation (application), solves the problem, and sends back the solution. We integrated the sparse linear solver in the *worker*, because this is a core task of the leg implant simulation. Furthermore, the linear solver is as one of the most important algorithms to support simulation in HPC.

- *Queue:* The *queue* module is responsible for the assignments of the workers to an application, making use of EXCESS HPC-Services.

4.1.1 Application

The application package includes an interface to the leg implant sumulation and an executable example to produce the computational tasks for the *worker*, which are similar to the above mentioned simulation. The *application* generates a sparse linear system, sends it to the *worker*, and receives the solution vectors from them. The operation is repeated several times. The *application* includes the definition of several linear systems with a various number of rows, which are range from 12 to several thousands.

4.1.2 Worker

The *worker* listens on sockets connected to the EXCESS queue, until there is a task to be solved. After the establishment of the connection to the application, it receives the tasks, solves them and send the solution back.

The *worker* provides the *Conjugate Gradient* method (CG), therefore it can solve the problem and send the correct solution to the *application*.

The available EXCESS linear solver for the Movidius platform is programmed for the execution on the bare metal [20]. We port the main part of the software to the new target system *RTEMS*. There is still some work to be done: the *SHAVE*s of the *Myriad2* processor are currently not involved in the solving the sparse linear systems.

4.1.3 Queue Handler (QHandler)

The Queue Handler (*QHandler* is responsible for managing workers and assigning workers to a requesting application. In this system, the *QHandler* is the only process that is required to always run as it can register new workers and also allocate jobs to existing workers.

When a worker node is activated, it registers with the *QHandler* , sending details with server ports, compute capabilities such as processor type (CPU, GPU, Myriad),

memory, buffer sizes e.t.c. On receiving the register request from the worker, the Queue handler adds the worker to the idle workers list. When an application requires worker nodes, it sends a requests to the *QHandler* with details of the application requirements and ports for the worker to connect to the application(numerical algorithms, IP address, and opened ports). The *QHandler* then forwards these details to one of the workers in the idle workers list.

When a worker node receives a job assignment from the *QHandler* , it proceeds to connect with the application and process the assigned workload. From this point on the worker and the application continue to synchronize and process the workload until the application terminates the job. On termination of the current jjob, the corresponding worker informs the *QHandler* of the change in status to idle.

An application can request for more than one worker, depending on the task and the parallelism that can be achieved by using multiple workers.

The *queue* currently provides only one operation. It sends the name of the needed numerical algorithms, IP address, and opened ports of the application to the selected *worker*.

4.2 Underground Mine Ventilation

Development of Distributed Simulation Environment for Security-Critical Technological Objects by Means of Microservices Modelling and simulation are well-established techniques used in science and technology for prediction, analysis, and evaluation of properties of various complex dynamic systems. The actual trend in the simulation technology is the integration with industrial automated control systems which aims to ensure a higher level of control and a better quality of the decision to be made. The model-aided support is of substantial importance for security-critical systems, such as underground mine ventilation networks, in which the risk of a spontaneous explosion of hazardous gases (like methane) is very high and might cause an enormous damage to human and technical resources. The high complexity of the dynamic processes and analysis methods requires a piece of high-performance hardware.

In case of big systems such as the targeted ventilation networks the use of supercomputers is necessary. However, there are cases in which the simulation should be performed closely to the controlled object (an element of the ventilation system). For example, this is required when an emergency situation has happened and the air distribution is being manually controlled by an underground rescue team, e.g. according to a special emergency response plan. In that case, the availability of a simulation platform that would be capable of predicting the development of the air- and gas-dynamic situation based on the current conditions will be of a great advantage, and will contribute to the quickest solving of the encountered problem. The goal of our research is to elaborate new concepts for the development of a portable simulation platform for use in industrial technological environments. The platform should provide support to automate control systems for the case of unexpected events that cannot be handled by the native control systems due to their limited functionality (cf. Figure 23).

The modelling environment should be designed in a service-oriented way in order to ensure interoperations with the high-performance computing infrastructures whenever a better quality of results is needed.

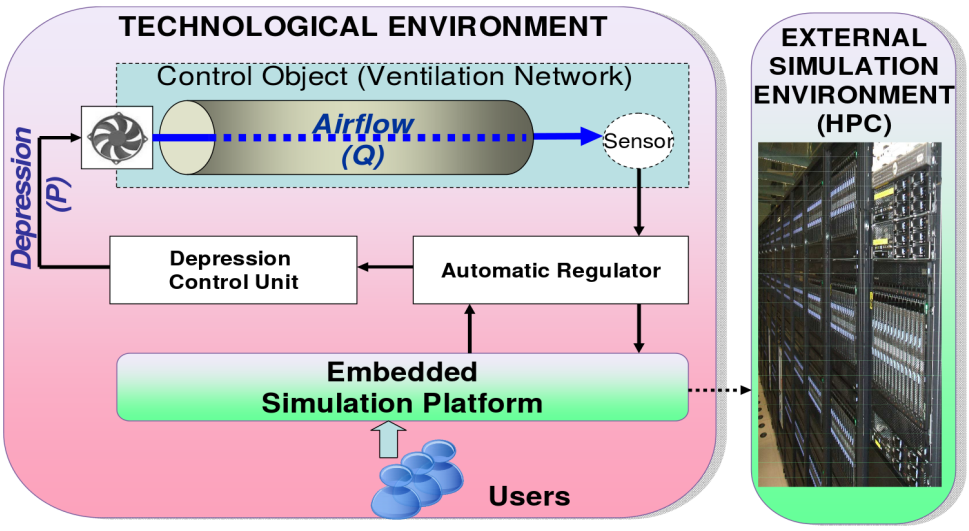


Figure 23: Technological Einvironment with a portable simulation platform

Envisioned Solution The use of conventional hardware in conditions of security-critical technological objects is often impossible due to factors like excessive dustiness, humidity, vibration, et cetera. The energy-efficiency requirements put another strict limitation to the hardware that is allowed to be used. The embedded hardware, for example Movidius platform, on the contrary to conventional ones (*x_86* and *x_64*), is expected to meet those requirements of the use in the technological environment. However, the performance of embedded systems is lower than that of conventional ones, which requires a trade-off between the required quality and performance of the simulation algorithms on the one hand, and the capabilities of the hardware platform on the other hand.

Well-established simulation software packages like OpenFOAM[18] are prevalently designed for conventional hardware. The broad spectrum of the embedded hardware platforms (from high-bandwidth microprocessors to SoCs) that is capable of use in industrial conditions as well as their adherence to the RISC CPU architectures prevents the portability of the simulation software to the "light-weighted" hardware. Cloud computing—the IT organization strategy that was widely established in 2000's—pushed the modelling and simulation community to reconsider their software development approaches towards their brighter service-orientation. The technological foundation for the development of simulation software in a decentralised way has been established, which allows the software components to become interoperable on heterogeneous hardware platforms. Service-oriented development in form of "microservices" (small interoperable components that implement a specific part of the modelling algorithm, see an example in Figure 24) seems extremely promising for the next-generation simulation platforms. This, however, requires new approaches to the organization of middleware frameworks for the development and execution of the service-oriented modelling software.

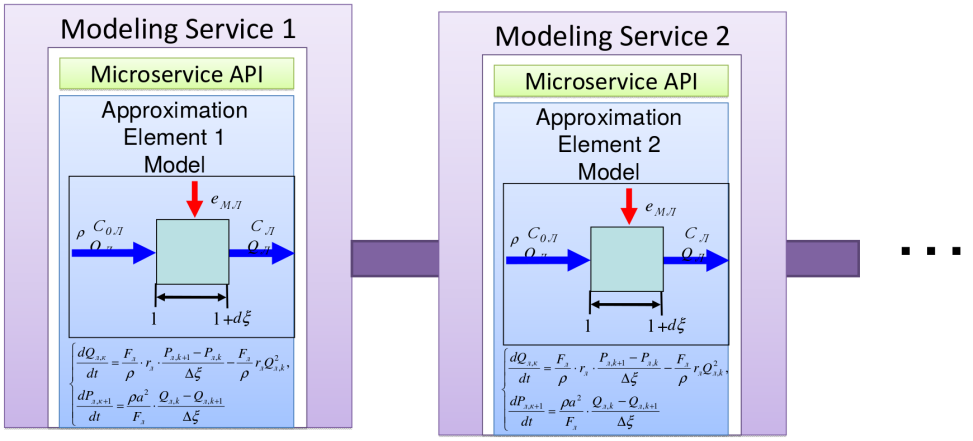


Figure 24: Example of a microservice-oriented simulation framework.

4.2.1 Future Development

The microservice-based approach allows a transition of the already known component-based simulation techniques (e.g., as proposed by Matlab and Simulink [24]) into a modern technological platform, leveraging the advances of service-oriented Cloud technologies.

[18] http://www.openfoam.com

The challenge of low-power hardware support is to be tackled by the software compilation and execution technologies developed within PHANTOM [13]—an EU-funded project started in December 2015.

Our goal is to elaborate a methodology for the development of portable, efficient, and scalable component-based simulation software based on a microservice architecture. The methodology will be implemented in a software platform for model-aided support of the security-critical technological processes.

We are going to reimplement the available modelling algorithms within the framework we are proposing, and evaluate the effects of porting to low-power embedded hardware in terms of performance and power consumption.

5 Conclusion

This deliverable has presented work done to evaluate well-established hardware and software in the field of HPC. We will give a brief summary of our main contributions, followed by final remarks.

Section 2 described the EXCESS cluster and its power measurement system. We showed the disadvantages and advantages of the system; we recall next the most important outcomes. The EXCESS cluster provides an execution environment together with the EXCESS power monitoring tools. The current configuration of the system supports the simultaneous operation of a maximum of 32 measurement channels running at 12.5 kHz. It should be noted that the number of measurement channels can be at least doubled. The first disadvantage is the fact that the measurement of the processors and of the memory cannot currently be decoupled. Nevertheless, we have some ideas how to distinguish the CPU power from the power of the RAM using so-called *DDR Riser Cards*. The second disadvantage comes from the fact that the power measurement and the progress of the program execution are synchronized with the help of timestamps. Currently, a user has to add manually additional instructions to store the timestamps of the various phases of the program. In this context, we aim to write a plug-in for the visualization performance tool *Vampir* [28]. Then, the user can use automatic instrumentation provided by several trace tools including *Score-P trace* [32] or *Vampir trace* [28]. With our power measurement hardware and tools, and the well-defined interface of *Vampir*, we do not see any possible complications.

In the second part of this document, we showed various results of the benchmarks, which were defined or extended in the project. Our study of the accelerators showed that our GPUs can be twice efficient as our CPUs. *TeslaK80* can achieve its peak performance for the small problem sizes, while *Ivy Bridge* and *Haswell* needed two times larger problem sizes for that. Unfortunately, current accelerators are connected to the host CPU through the PCIe version 3.0 interface, whose performance is a bottleneck for the distributed computation.

The Movidius platform has a high performance for the small problem sizes, particularly for a dense algebra. The energy efficiency for small problem sizes is higher than that of CPUs and GPUs. Perhaps, the most important reason for it is the lower static power of *Myriad2*. However, substantial effort is still needed to overcome existing limitations, especially in problematic areas for distributed computations such as sparse linear algebra, whose memory access patterns are not regular. It makes the optimization on low level very complex. This was also one of the reason, why we have not yet optimized our code. At this point we would like to point out that the Movidius MDK contains too few examples for such kind of algorithms, which is also understandable because the primary target of the system are multimedia applications. However, the results for dense algebra and deep learning algorithms illustrate the potential of the system. Nevertheless, there are two main issues, which should be resolved in order to be used for the distributed algorithms in HPC: double-precision arithmetic and support for high-performance networks. We hope that our project and the close co-operation with the Movidius company will trigger off further initiatives in these fields. From our part, we use the Movidius platforms for further development within PHANTOM [13].

References

[1] Intel Xeon Processor E5-2680 v3, Technical description. https://ark.intel.com/products/81908/Intel-Xeon-Processor-E5-2680-v3-30M-Cache-2_50-GHz, . Accessed: 2015-06-30.

[2] Intel Xeon Processor E5-2690 v2, Technical description. http://ark.intel.com/products/75279/Intel-Xeon-Processor-E5-2690-v2-25M-Cache-3_00-GHz, . Accessed: 2014-11-26.

[3] HP Memory 4 GB DDR3, Technical description. http://h30094.www3.hp.com/product/sku/10715899. Accessed: 2015-01-18.

[4] Mellanox FDR Single/Dual-Port InfiniBand Host Channel Adapter Card, technical description. http://www.mellanox.com/related-docs/prod_adapter_cards/PB_Connect-IB.pdf. Accessed: 2015-01-18.

[5] NVIDIA TeslaK40c Graphik-Accelerator, Technical description. http://www.nvidia.com/object/tesla-servers.html. Accessed: 2015-01-18.

[6] Samsung DDR4 SDRAM Memory, Product Guide. http://www.samsung.com/global/business/semiconductor/file/product/DDR4_Product_guide_Dec13.pdf. Accessed: 2015-06-30.

[7] ADDI-DATA. Technical description apcie-3021, apcie-3121 und apcie-3521 multifunction board and analog/input board, optically isolated. http://www.addi-data.com/manuals/english/analog/APCIe-3x21_CPCIs-3121_e.pdf. Ausgabe:02.07-11/2013.

[8] Brendan Barry. D4.4: EXCESS Report on the final evaluation of compute card prototype. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2016.

[9] E. A. Burton, G. Schrom, F. Paillet, J. Douglas, W. J. Lambert, K. Radhakrishnan, and M. J. Hill. Fivr #x2014; fully integrated voltage regulators on 4th generation intel #x00ae; core #x2122; socs. In *2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014*, pages 432–439, March 2014. doi: 10.1109/APEC.2014.6803344.

[10] Intel® Corporation. *Intel Xeon 64 Processor E5-1600 and E5-2600 v3 Product Families,Volume 1 of 2, Electrical*. Number 330783-001. 9 2014.

[11] On-Line Applications Research Corporation. 1003.1b-1993 - ieee standard for information technology - portable operating system interfaces (posix(r)). https://docs.rtems.org/doc-current/share/rtems/pdf/posix_users.pdf, 07 2015.

[12] On-Line Applications Research Corporation. Rtems posix api users guide, edition 4.10.99.0, for rtems 4.10.99.0. `https://docs.rtems.org/doc-current/share/rtems/pdf/posix_users.pdf`, 07 2015.

[13] PHANTOM EU funded project. Cross-layer and multi-objective programming approach for next generation heterogeneous parallel computing systems. `http://www.phantom-project.eu`, 2016-2018.

[14] Dennis Hoppe, Dmitry Khabi, Fangli Pi, Phuong Ha, Ibrahim Umar, Erik Hansson, Lu Li, Cristoph Kessler, Anders Gidenstann, and Ivan Walulya. D5.7: Report on Integration the Final Designs and Prototypes from Technical WPs. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2016.

[15] Dennis Hoppe, Dmitry Khabi, Fangli Pi, Christoph Kessler, and Erik Hansson. D3.4: Final Prototype of Benchmarks and Monitoring Framework including Runtime System Support. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2016.

[16] Intel Corporation Inc. Intel Xeon Processor E5 v3 Family Specification Update. `http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-e5-v3-spec-update.pdf`, February 2016. Accessed: 2016-08-01.

[17] M. H. Ionica and D. Gregg. The Movidius Myriad Architectures Potential for Scientific Computing. *Micro, IEEE*, 35:6–14, Jan 2015.

[18] D. Khabi, B. Barry, P. Renaud-Goud, D. Moloney, and I. Walue. D3.1: Analysis of the Movidius platform architecture from the HPC developer's standpoint. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2014.

[19] Dmitry Khabi. D5.4: Prototype of an energy-aware system based on conventional HPC technology. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2014.

[20] Dmitry Khabi. D5.5: Linear solvers ported to EXCESS platforms. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2015.

[21] Dmitry Khabi and Ivan Walulya. D5.6: HPC application prototype ported to EXCESS platforms. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2016.

[22] Dmitry Khabi, Vi Tran, and Ivan Walulya. D5.4: Report on the first evaluation results and discussion. Public deliverable, The EXCESS Project (FP7/2013-2016 grant agreement no 611183), 2014.

[23] Innovative Computing Laboratory. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. `http://www.netlib.org/benchmark/hpl/`, 2016. Accessed: 2016-08-12.

[24] MATLAB. Introduction to matlab functional blocks. http://de.mathworks.com/help/simulink/ug/what-is-a-matlab-function-block.html?s_tid=gn_loc_drop, 2016.

[25] Movidius. Myriad 2 Development Kit Programmer's Guide. Movidius confidential, Movidius, 2015.

[26] Movidius Inc. Software Development Kit. http://www.movidius.com/solutions/software-development-kit, 2016. Accessed: 2016-08-01.

[27] Aris Mpitziopoulos. Antec vpf series 650w psu review. http://http://www.tomshardware.com/reviews/antec-vpf-650w-power-supply,4232-3.html, August 2015.

[28] Aris Mpitziopoulos. Vampir - performance optimization. https://www.vampir.eu/, 2016.

[29] The RTEMS Project. RTEMS Real Time Operating System (RTOS). https://www.rtems.org/, 2016. Accessed: 2016-04-11.

[30] Yosandra Sandoval, Dennis Hoppe, and et al. EXCESS: Execution Models for Energy-Efficient Computing Systems. Technical report, fEEDBACk Workshop 2015: ENERGY EFFICIENT DISTRIBUTED AND PARALLEL COMPUTING, Held in conjunction with PODC-2015, Donostia-San Sebastin, Spain, on July 20th, 2015, April 2015.

[31] Ralf Schneider. Identification of anisotropic elastic material properties by direct mechanical simulations: estimation of process chain resource requirements. In Michael Resch, Katharina Benkert, Xin Wang, Martin Galle, Wolfgang Bez, Hiroaki Kobayashi, and Sabine Roller, editors, *High Performance Computing on Vector Systems 2010*, chapter 3, pages 149 – 159. Springer, Berlin, 2010. ISBN 978-3-642-11850-0.

[32] SCORE-P. Scalable performance measurement infrastructure for parallel codes. http://www.vi-hps.org/projects/score-p/, 2016.

[33] Anand Lal Shimpi. Intel's haswell architecture analyzed: Building a new pc ans a new intel. *AnandTech*, 2012. URL http://www.anandtech.com/show/6355/intels-haswell-architecture/.

[34] TOP500. The Linpack Benchmark. https://www.top500.org/project/linpack/, 2016. Accessed: 2016-08-12.

[35] Valter QUercioli. *Pulse Width Modulated (PWM) Power Supplies.* Elsevier, Amsterdam, 1993. ISBN 0444897909.

[36] Wikipedia. Basic Linear Algebra Subprograms (BLAS). https://de.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms, 2016. Accessed: 2016-08-12.

[37] Wikipedia. Cancellous bone. *The Free Encyclopedia*, 2016. URL https://en.wikipedia.org/wiki/Cancellous_bone. Accessed: 2016-08-12.

6 Glossary

Term	Description
API	Application Programming Interface
ATOM	Monitoring framework for the conventional HPC and Movidius platforms
AVX	Advanced Vector Extensions
BSD	Berkeley Software Distribution (BSD) is a Unix operating system derivative
Chalmers	Chalmers University of Technology
FMA	Fused Multiply-Add instruction
CG	Conjugate Gradient method
CT	Computed tomography
CRS	Compressed sparse row storage format represents a sparse matrix.
EXCESS	Execution Models for Energy-Efficient Computing Systems
GPGPU	General Purpose Computation on Graphics Processing Unit
HLRS	High Performance Computing Center Stuttgart
HPC	High Performance Computing
I/O	Input/Output
JTAG	Joint Test Action Group specifies the use of a dedicated debug port
LiU	Linköping University
MPI	Message Parsing Interface
PBS	Portable Batch System
PTP	Precision Time Protocol
PCIe	PCI Express (Peripheral Component Interconnect Express)
RISC	Reduced instruction set computing
RTEMS	Real-Time Executive for Multiprocessor Systems
SHAVE	Movidius SHAVE processor is a hybrid stream processor architecture combining the various features of GPUs, DSPs and RISC
TCP/IP	Transmission Control Protocol / Internet Protocol (TCP/IP)
UiT	University of Tromsø
USB	Universal Serial Bus
USTUTT	Stuttgart University
x_86	x86 is a family of instruction set for architectures based on the Intel CPUs.