



## Hardware developments II

E-CAM Deliverable 7.3

Deliverable Type: Report

Delivered in Month 21– June 2017



E-CAM

The European Centre of Excellence for  
Software, Training and Consultancy  
in Simulation and Modelling



Funded by the European Union under grant agreement 676531

### Project and Deliverable Information

Project Title	E-CAM: An e-infrastructure for software, training and discussion in simulation and modelling
Project Ref.	Grant Agreement 676531
Project Website	<a href="https://www.e-cam2020.eu">https://www.e-cam2020.eu</a>
EC Project Officer	Juan Pelegrín
Deliverable ID	D7.3
Deliverable Nature	Report
Dissemination Level	Public
Contractual Date of Delivery	Project Month 21 (June 2017)
Actual Date of Delivery	16.10.2017
Description of Deliverable	Update on "Hardware Developments I" (Deliverable 7.1) which covers: <ul style="list-style-type: none"> <li>- Report on hardware developments that will affect the scientific areas of interest to E-CAM and detailed feedback to the project software developers (STFC);</li> <li>- discussion of project software needs with hardware and software vendors, completion of survey of what is already available for particular hardware platforms (FR-IDF); and,</li> <li>- detailed output from direct face-to-face session between the project end-users, developers and hardware vendors (ICHEC).</li> </ul>

### Document Control Information

Document	Title:	Hardware developments II
	ID:	D7.3
	Version:	As of October 16, 2017
	Status:	Accepted by WP leader
	Available at:	<a href="https://www.e-cam2020.eu/deliverables">https://www.e-cam2020.eu/deliverables</a>
Review	Document history:	<a href="#">Internal Project Management Link</a>
	Review Status:	Reviewed
Authorship	Action Requested:	<b>Submit</b>
	Written by:	Liang Liang (MdlS/IDRIS/CNRS, France)
	Contributors:	Alan O'Cais (JSC), Jony Castagna (STFC), Simon Wong (ICHEC), Goar Sanchez (ICHEC),
	Reviewed by:	Daniel Borgis (MdlS/ENS/CNRS), Alan O'Cais (JSC)
	Approved by:	Godehard Sutmann (JSC)

### Document Keywords

Keywords:	E-CAM, HPC, Hardware, CECAM, Materials, ...
-----------	---

October 16, 2017

**Disclaimer:** This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the Project and to the extent foreseen in such agreements.

**Copyright notices:** This deliverable was co-ordinated by Liang Liang<sup>1</sup> (MdlS/IDRIS/CNRS, France) on behalf of the E-CAM consortium with contributions from Alan O'Cais (JSC), Jony Castagna (STFC), Simon Wong (ICHEC), Goar Sanchez (ICHEC). This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0>.



<sup>1</sup>[liang.liang@idris.fr](mailto:liang.liang@idris.fr)

# Contents

<b>Executive Summary</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Scope of Deliverable	2
1.1.1 Target Audience	2
1.2 Extent of the update to <i>Hardware Developments I</i>	2
<b>2 Hardware Developments</b>	<b>3</b>
2.1 Influential factors	3
2.1.1 The impact of power considerations	3
2.1.2 The impact of Deep Learning	4
2.2 Intel Many-Core	5
2.2.1 Influence of Deep Learning	5
2.2.2 Feedback for software developers	5
2.3 NVIDIA GPU	5
2.3.1 NVLINK 2.0	6
2.3.2 Influence of Deep Learning	6
2.3.3 Feedback for software developers	6
2.4 FPGA	7
2.4.1 EuroEXA	7
2.4.2 Feedback for software developers	7
2.5 Other significant developments	7
2.5.1 Non-volatile Memory	7
2.6 Extreme-scale resources at the European level	7
2.6.1 Current PRACE Resources	7
2.6.2 Extreme-scale Demonstrators	8
2.6.3 Feedback for software developers	9
<b>3 Software Needs</b>	<b>10</b>
3.1 Software needs as collected by the E-CAM Software Survey	10
3.1.1 Feedback for hardware and software vendors	12
3.2 Programming Paradigms	12
3.2.1 C++17	12
3.2.2 Fortran 2015	12
3.2.3 The (potential) role of Python	13
3.2.4 Open Standards	13
3.2.5 Runtime System Approaches	14
3.2.6 Feedback for software developers	14
<b>4 Interactions between end-users, developers and vendors</b>	<b>15</b>
4.1 The European exascale roadmap	15
4.2 Emerging hardware architectures relevant to exascale computing	16
4.2.1 The DEEP projects – overview of the architecture	16
4.2.2 The Mont-Blanc projects	17
4.2.3 Other discussions with hardware and software vendors	18
4.3 Exascale challenges in exploiting massive parallelism	18
4.3.1 “Large scale electronic structure calculations with CP2K” - Jurg Hutter	18
4.3.2 “Scalability of Path Sampling Simulations” – David Swenson	19
4.3.3 “PaPIM: A Code for (Quantum) Time Correlation Functions” – Momir Malis	19
4.3.4 “Porting DL_MESO_DPD on GPUs” – Jony Castagna	20
4.3.5 “MP2C: Multiple-particle collision dynamics on massively parallel computers” – Godehard Sutmänn	20
4.3.6 “POP – understanding applications and how to prepare for exascale” – Jesus Labarta	21
4.3.7 “Exascale challenges: the view from MaX” – Carlo Cavazzoni	22
4.3.8 “Exascale challenges: the view from NoMAD” – Claudia Draxl	22
4.3.9 “Exascale challenges: the view from EoCoE” – Matthieu Haefele	23
4.4 Outcomes and key recommendations for E-CAM	24
4.4.1 WP7 Pilot Projects	24
<b>References</b>	<b>25</b>

## List of Figures

1	Eurolab-4-HPC summary of roadmaps relevant to the development of HPC . . . . .	3
2	Hybrid Cube Mesh NVLink Topology as used in DGX-1 with V100. . . . .	6
3	V100 with NVLink Connected GPU-to-GPU and GPU-to-CPU (only currently available with Power8/9 architectures). . . . .	6
4	List of PRACE Resources (as of 2017) with associated hardware characteristics. . . . .	8
5	The European EsD developments mapped onto the timeline of the European Horizon 2020 research time-line . . . . .	8
6	Question: Who develops the software? . . . . .	10
7	Question: The Programming Language used by the software. . . . .	10
8	Supported (parallel) software capabilities . . . . .	11
9	Most common hardware used for generating publication-quality results . . . . .	11
10	Max number of cores used per run . . . . .	11
11	Max number of hours used per run . . . . .	11
12	The European HPC Technology Projects within the European HPC Eco-system. . . . .	16
13	The DEEP hardware architecture. . . . .	17
14	CP2K Presentation: GPU and KNL Performance . . . . .	19
15	DL_MESO: GPU porting results . . . . .	20
16	MP2C scaling on JUGENE . . . . .	21
17	POP: Different software layers . . . . .	21
18	Fundamental performance factors in evaluating performance . . . . .	22
19	Overview of materials data and their structure . . . . .	23
20	EoCoE Performance metrics table . . . . .	23

## Executive Summary

This deliverable is a update on *Hardware developments I*[1] which was submitted as D7.1 in the first reporting period of the project, in June 2017. Based on the perceived needs of our user community we have restructured the deliverable and it is now composed of 3 main parts, namely:

- *Hardware Developments* (Section 2) which provides a summarised update to Hardware Developments that we see as relevant to the E-CAM community in the 3-5 year horizon. In this section, roadmaps of HPC development, power considerations, hardware developments of Intel Many-Core machine, NVIDIA GPU and FPGA are presented, as well as the current and expected availability of extreme-scale resource at the European level. Detailed feedback to E-CAM software developers is also given for each subsection covering topics such as: enhancing parallelism through vectorization both from numerical algorithm point of view and at compiler level; using several development approaches such as CUDA, mOpenACC, OpenMP 4.5 to exploit the full power of GPUs, FPGAs and many-core architectures. It is fundamental to consider the several issues related the heterogeneity of exascale architectures, such as CPU-GPU bandwidth communication. Although it is too early to know the hardware technology content of the Extreme-scale Demonstrators (for whom a call for proposals is imminent), the technologies described in this section are very likely to feature.
- *Software Needs* (Section 3) gives an update to the software needs of the project and what will be required of the software extreme-scale resources. In this section, a first analysis of the ongoing "E-CAM Survey of Application Software" is provided. Feedback for hardware and software vendors is given, noting in particular that there is still a heavy reliance in the community on Fortran and support for this language will be essential on future systems for current workloads. A subset of possible programming paradigms are covered: C++17, Fortran 2015, several Open Standards (MPI/OpenMP/OpenACC/OpenCL) and runtime-system approaches (HPX/Kokkos/OmpSs). For software developers, awareness of the latest standards and the status of their implementations are critical during application development as these new features exist to target the scalability challenges on modern systems. We provide a limited set of recommendations for people in E-CAM community who are embarking on a new software project: using MPI+OpenMP with their latest standards remains the safest bet with good performance and scalability achievable; you can prototype quickly using Python and leveraging the Python APIs to the libraries you need; using C++ with Python interfaces can help achieve maximal performance (and it helps that computer scientists seem to prefer C++ when creating innovative runtime systems).
- *Interactions between end-users, developers and vendors* (Section 4) describes the face-to-face discussions that have taken place between E-CAM developers users and the hardware and software vendors. This section primarily reflects the E-CAM Extreme-Scale State-of-the-Art Workshop held in Barcelona in June 2017. The European exascale roadmap, emerging hardware architectures relevant to exascale computing and exascale challenges in exploiting massive parallelism are reported in this section. Key outcome and recommendations include: E-CAM needs to develop niche activities to provide mutual benefit for both our industry partners and E-CAM's HPC oriented goals. We should provide a reliable software delivery mechanism between academic software developers and industrial partners. We also seek to increasing the E-CAM community exposure to HPC technologies by organising a set of pilot projects covering High Throughput Computing (HTC) and novel runtime technologies. In order to achieve E-CAM's internal goals, additional steps are to be taken: enhancing collaboration with EoCoE, POP and PRACE to optimise the quality of the E-CAM modules developed; transversal ESDW workshop can be organised with the goal of increasing synergy between WPs while ensuring that E-CAM developers are on similar levels in terms of parallel code development skills; and increase the impact of the modules developed by E-CAM community by publishing pilot project webpage and related twitters.

Discussions are also realised with hardware/software vendors through our participation in the development of the ETP4HPC Strategic Research Agenda and our participation in the 2017's Teratec forum.

# 1 Introduction

## 1.1 Scope of Deliverable

This deliverable is divided into 3 main parts, namely:

- Analysis of hardware developments with feedback to software developers (Section 2);
- Survey of software and hardware currently used by the E-CAM community (and the level of parallelisation used). Analysis of software future needs and requirements with feedback to software developers, as well as hardware and software vendors (Section 3);
- Discussion of project software needs with hardware and software vendors, and detailed output from direct face-to-face session between the project end-users, developers and hardware vendors (Section 4).

### 1.1.1 Target Audience

The recommendations of this deliverable are targeted at the development community connected to E-CAM. We highlight the impact that immediate hardware developments are likely to have on the software applications of the E-CAM community (and the tools used to build them). We also provide a set of recommendations to our user community with respect to how to prepare for, and deal with, these developments.

In addition, our software survey, in particular, provides hardware and software vendors with valuable insight into the normal simulation software and practices of our user community.

## 1.2 Extent of the update to *Hardware Developments I*

In E-CAM's *Hardware Developments I* deliverable [1], we concentrated on the state-of-the-art of hardware and software in the context of our understanding of the E-CAM community's interests and gave recommendations to E-CAM's software developers. The massively parallel architectures of the PRACE Tier-0 or Tier-1 supercomputers were reviewed, as well as the other computing platforms soon to be available (at the time). We provided a set of recommendations on the state-of-the-art software that should be used by the E-CAM developers for programming, profiling, debugging, and using efficient input/output in order to take full advantage of the current and (immediate) future HPC machines.

In the current document, we narrow our focus to the disruptive aspects of these (updated) hardware and software technologies that are most likely to impact the software development practices of the E-CAM community. With respect to the availability of particular hardware, we update the platforms of the current Partnership for Advanced Computing in Europe (PRACE) infrastructures.

The design restrictions brought about by the power budget for future extreme-scale machines have encouraged us to focus exclusively on a subset of technologies (Intel Xeon/Xeon Phi, Nvidia Tesla and FPGA) that are most likely to feature in such machines in the next 5 years. Emergent memory/storage techniques are also mentioned, but the impact of these at the European level is only likely to be seen in the timeframe of the [Extreme-scale Demonstrators \(EsD\)](#) Call for Proposals.

At the end of *Hardware Developments I*, in order to make sure that the E-CAM pool of developers is prepared for the exascale challenge for their applications, we proposed to organise an E-CAM HPC workshop in Q1-2017. This workshop ultimately took place in June 2017 and gathered the pool of E-CAM developers from the various application WPs, High Performance Computing (HPC) vendor representatives and HPC specialists presenting the emerging platforms and their programming paradigms. The workshop was intended to drive and shape the development efforts within E-CAM over the remaining period of the project. The content and outcomes of this workshop are covered in Section 4 of this deliverable.

In addition, we have conducted a software usage and needs survey for the E-CAM community whose results are summarised here. We have had numerous contacts with [ETP4HPC](#) (an industry-led think-tank comprising of European HPC technology stakeholders) with respect to the development of the ETP4HPC Strategic Research Agenda (SRA) which outlines a roadmap for the achievement of exa-scale capabilities by the European High-Performance Computing (HPC) ecosystem. We have also participated in 2017's Teratec Forum. The scope of these contacts will be analysed and summarised in this deliverable.

## 2 Hardware Developments

	Goal	Timespan	SWOT/ Political	Scope
HiPEAC Vision	Steer European academic research (driven by industry)	Short: 3 years, Mid: 6 years, Long: > 2020	Yes	HPC + embedded
ETP4HPC SRA/EXDCI	Strengthening European (industrial) HPC ecosystem	6 years (2014 to 2020)	Yes	HPC except applica- tions
PRACE Scientific Case	(Academic) need for European HPC infras- tructure	8 years (2012 to 2020)	Yes	HPC applications
EESI (European Exascale Software Initiative)	Development of efficient Exascale applica- tions	5 to 10 years	No	Exascale applications
BDVA (Big Data Value Associa- tion)	Big Data technologies roadmap	2020	–	Big data
Rethink Big	Roadmap for European Technologies in Hardware and Networking for Big Data	–	–	Big data
ECSEL MASRIA	European leadership in enabling and indus- trial technologies. Competitive EU ECS indus- try.	2015 roadmap to about 2025	Yes	Electronic compo- nents and systems (ECS)
Next Genera- tion Computing Roadmap	Strengthening European industry	2014: 10 to 15 years	–	HPC extensively cov- ered
Eurolab-4-HPC	Academic excellence in HPC	2023 – 2030	No	Whole HPC stack

Figure 1: Eurolab-4-HPC summary of roadmaps relevant to the development of HPC

There are a number of different organisations and projects that are generating roadmaps about the current and future technologies that are (or may be in the future) available in the HPC space. A summary table has been created by Eurolab-4-HPC for their report on the "[Eurolab-4-HPC Long-Term Vision on High-Performance Computing](#)" (which is reproduced in Fig. 1). In this section we will focus on a small subset of the content of these roadmaps (primarily from Eurolab-4-HPC and ETP4HPC) that are most likely to impact the target community of E-CAM in the 3-5 year horizon.

### 2.1 Influential factors

There are a number of economical factors that will have a strong influence on aspects of the hardware that will be realised in exascale machines. We discuss two here: the practical consideration of the power budget required to run such a resource and the market-led influence of deep learning on *commodity* hardware that can be leveraged.

#### 2.1.1 The impact of power considerations

For the last decade, power and thermal management has been of high importance. The entire market focus has moved from achieving better performance through single-thread optimizations, e.g., speculative execution, towards simpler architectures that achieve better performance per watt, provided that vast parallelism exists. The HPC community, particularly at the higher end, focuses on the flops/watt metric since the running cost of high-end HPC systems are so significant. It is the potential power requirements of exa-scale systems that are the limiting factor (given currently available technologies).

The practical outcome of this is the rise of accelerating co-processors and many-core systems. In the following sections we will discuss three such technologies that are likely to form the major computing components of the first generation of exa-scale machines:

- Intel MIC (Xeon Phi or similar)
- NVIDIA GPU

- FPGA

We will outline the current generation of technologies in this space and also describe the (currently) most-productive programming model for each. We will not discuss other new CPU technologies (such as Power 9, Intel Skylake, or ARMv8) since in comparison to these technologies they would be expected to only provide ~10% or less of the compute power of potential exa-scale systems.

The problem with the current three-pronged advance is that it is not always easy to develop parallel programs for these technologies and, moreover, those parallel programs are not always performance portable between each technology, meaning that each time the architecture changes the code may have to be rewritten. While there are open standards available for each technology, each product currently has different preferred standards which are championed by the individual vendors (and therefore the best performing).

In general, we see a clear trend towards more complex systems, which is expected to continue over the next decade. These developments will significantly increase software complexity, demanding more and more intelligence across the programming environment, including compiler, run-time and tool intelligence driven by appropriate programming models. Manual optimization of the data layout, placement, and caching will become uneconomic and time consuming, and will, in any case, most likely soon exceed the abilities of the best human programmers.

#### **Design consequences: MareNostrum 4**

MareNostrum 4 is a prime example of the recognition of these developments. It includes two separate parts: a general-purpose block (that can consume existing MareNostrum 3 workloads) and a block featuring emerging technologies. Once completed, its total speed will be 13.7 Petaflops.

The general-purpose block has 48 racks with 3,456 nodes. Each node has two Intel Xeon Platinum chips (Skylake architecture), each with 24 processors (48 cores per node could already be considered a many-core node), amounting to a total of 165,888 processors and a main memory of 390 Terabytes. Its peak performance is 11.15 Petaflops. While its performance is 10 times greater than its predecessor, MareNostrum 3, its energy consumption will only increase by 30% to 1.3 MW per year.

The block of emerging technologies is formed of clusters of three different technologies, which will be incorporated and updated as they become available on the market. These technologies are currently being developed in the United States and Japan to speed up the arrival of the new generation of pre-exascale supercomputers. They are as follows:

- Cluster comprising IBM POWER9 and NVIDIA Volta GPUs, with a computational capacity of over 1.5 Petaflops. IBM and NVIDIA will use these processors for the Summit and Sierra supercomputers that the US Department of Energy has ordered for its Oak Ridge and Lawrence Livermore National Laboratories.
- Cluster formed of Intel Knights Hill (KNH) processors, with a computational capacity of over 0.5 Petaflops. These are the same processors as those to be used in the Theta and Aurora supercomputers that the US Department of Energy has ordered for the Argonne National Laboratory.
- Cluster composed of 64-bit ARMv8 processors in a prototype machine with a computational capacity of over 0.5 Petaflops. This cluster will use the cutting-edge technology of the Japanese supercomputer Post-K.

The aim of gradually incorporating these emerging technologies into MareNostrum 4 is to allow Barcelona Supercomputing Center (BSC) to experiment with what are expected to be the most advanced technological developments over the next few years and evaluate their suitability for future iterations of MareNostrum.

#### **2.1.2 The impact of Deep Learning**

Traditional machine learning uses handwritten feature extraction and modality-specific machine learning algorithms to label images or recognize voices. However, this method has several drawbacks in both time-to-solution and accuracy. Today's advanced deep neural networks use algorithms, big data, and the computational power of the GPU (and other technologies) to change this dynamic. Machines are now able to learn at a speed, accuracy, and scale that are driving true artificial intelligence and AI Computing.

Deep learning is used in the research community and in industry to help solve many big data problems such as computer vision, speech recognition, and natural language processing. Practical examples include:

- Vehicle, pedestrian and landmark identification for driver assistance
- Image recognition
- Speech recognition and translation



- Natural language processing
- Life sciences

The influence of deep-learning on the market is significant with the design of *commodity* products such as the Intel MIC and NVIDIA Tesla being heavily impacted. Silicon is being dedicated to deep learning workloads and the scientific workloads for these products will need to adapt to leverage this silicon.

## 2.2 Intel Many-Core

The 2nd Generation Intel Xeon Phi platform, known as Knights Landing (KNL), has been released on the market in Q2 of 2016. The chip, based on a 14nm lithography, contains up to 72 cores @1.5GHz with a maximum memory bandwidth of 115.2 GB/s. One of the main features is the increased AVX512 ISE (Instruction Set Extensions) which includes SSE (Streaming SIMD Extensions) and AVX (Advanced Vector Extensions). More details are available at [Intel Knights Landing](#). The same component (Intel Xeon Phi 7250-F) is available in the JURECA Booster as part of the DEEP and DEEP-ER projects.

Knights Hill is the codename for the third-generation MIC architecture and it will be manufactured in a 10 nm process. Intel announced the first details at [SC14](#), however since then no further details have been released and the Aurora project from the DoE delayed (see [Some Surprises in the 2018 DoE Budget for Supercomputing](#)).

### 2.2.1 Influence of Deep Learning

[Knights Mill](#) is Intel's codename for a Xeon Phi product specialised in deep learning. It is expected to support reduced variable precision which have been used to accelerate machine learning in other products, such as half-precision floating-point variables in Nvidia's Tesla.

### 2.2.2 Feedback for software developers

Based on the latest hardware developments specified above (and the AVX512 instruction set used by this hardware), we strongly advise the software developer to take in consideration the importance of enhancing performance through vectorization both from numerical algorithm point of view and at the compiler level. Intel provides very good tools to achieve this through compiler flags (which allow you to have a full report about the vectorization efficiency) or more sophisticated software like [Intel Advisor](#).

At node-level the recommended parallelism is by shared memory. In this case [OpenMP](#) is the de facto standard and Intel provides good tools like [VTune](#).

Many training courses and documents are available on line (see training for [Intel Advisor](#) and [Vtune](#)).

## 2.3 NVIDIA GPU

The new NVIDIA [Tesla V100](#) accelerator incorporates the new Volta GV100 GPU. Equipped with 21 billion transistors, Volta delivers over 7.5 Teraflops per second of double precision performance, ~1.5x increase compared to the its predecessor, the Pascal GP100 GPU. Moreover, architectural improvements include:

- A tensor core is unit that multiplies two 4×4 FP16 matrices, and then adds a third FP16 or FP32 matrix to the result by using fused multiply-add operations, and obtains an FP32 result that could be optionally demoted to an FP16 result. Tensor cores are intended to speed up the training of neural networks.
- Tesla V100 uses a faster and more efficient HBM2 implementation. HBM2 memory is composed of memory stacks located on the same physical package as the GPU, providing substantial power and area savings compared to traditional GDDR5 memory designs, thus permitting more GPUs to be installed in servers. In addition to the higher peak DRAM bandwidth on Tesla V100 compared to Tesla P100, the HBM2 efficiency on V100 GPUs has been significantly improved as well. The combination of both a new generation HBM2 memory from Samsung, and a new generation memory controller in Volta, provides 1.5x delivered memory bandwidth versus Pascal GP100, and greater than 95% memory bandwidth efficiency running many workloads.
- NVlink 2.0 (see section below).

### 2.3.1 NVLINK 2.0

NVlink 2.0 is a high-bandwidth bus between multiple GPUs, and between the CPU and GPU. Compared to NVLink on Pascal, NVLink 2.0 on V100 increases the signaling rate from 20 to 25 Gigabits/second. Each link now provides 25 Gigabytes/second in each direction. The number of links supported has been increased from four to six pushing the supported GPU NVLink bandwidth to 300 Gigabytes/second. The links can be used exclusively for GPU-to-GPU communication as in the DGX-1 with V100 topology shown in Figure 2, or some combination of GPU-to-GPU and GPU-to-CPU communication as shown in Figure 3 (currently only available in combination with Power8/9 processors).

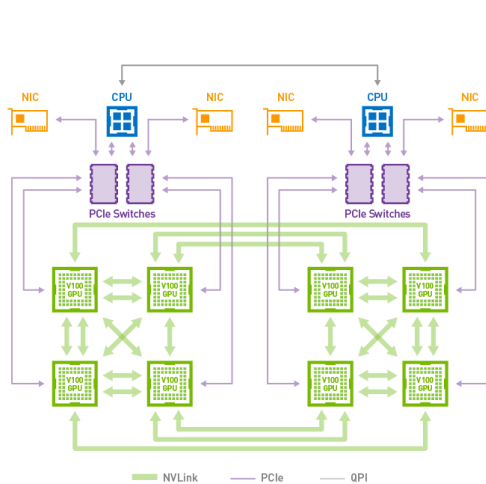


Figure 2: Hybrid Cube Mesh NVLink Topology as used in DGX-1 with V100.

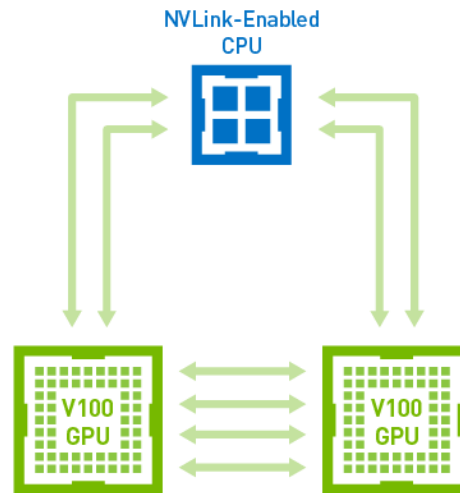


Figure 3: V100 with NVLink Connected GPU-to-GPU and GPU-to-CPU (only currently available with Power8/9 architectures).

In the GPU-GPU connection between nodes, two possible solutions are available that rely on the MPI layer:

- to use Unified Memory with CUDA-aware MPI library (currently supported by OpenMPI, Cray, IBM platform MPI and MVAPICH). This implementation allows a direct access from GPUs across different nodes without transferring the data to the CPU memory, resulting in much faster communication;
- to use the regular MPI from node to node, but data needs to be copied to and from the CPU memory.

### 2.3.2 Influence of Deep Learning

The tensor core of the Volta was explicitly added for deep learning workloads. The [NVIDIA Deep Learning SDK](#) provides powerful tools and libraries for designing and deploying GPU-accelerated deep learning applications. It includes libraries for deep learning primitives, inference, video analytics, linear algebra, sparse matrices, and multi-GPU communications.

### 2.3.3 Feedback for software developers

Several approaches have been developed to exploit the full power of GPUs: from parallel computing platform and application programming interface specific for NVidia GPU, like [CUDA 9.0](#), to the latest version of [OpenMP 4.5](#) which contains directives to offload computational work from the CPU to the GPU. While CUDA currently is likely to achieve best performance from the device, OpenMP allows for better portability of the code across different architectures. Finally, the [OpenACC open standard](#) is an intermediate between the two, more similar to OpenMP than CUDA, but allowing better usage of the GPU. Developers are strongly advised to look into these language paradigms.

Moreover, it is fundamental to consider that there are several issues linked to hybrid architectures, like CPU-GPU and GPU-GPU bandwidth communication (the latest greatly improved through NVlink), direct access through [Unified Virtual Addressing](#), the presence of new APIs for programming (such as [Tensor Core](#) multiplications specifically designed for deep learning algorithms).

Finally, it is important to stress the improvements made by NVidia on the implementation of *Unified Memory*. This allows the system to automatically migrate data allocated in Unified Memory between host and device so that it looks like CPU memory to code running on the CPU, and like GPU memory to code running on the GPU making programmability greatly simplified (see [Unified Memory](#) for more details on the latest developments).

At this stage, GPU programming is quite mainstream and there are many training courses available online, see for example the [NVidia education site](#) for material related to CUDA and OpenACC. Material for OpenMP is more limited, but as an increasing number of compilers begin to support the OpenMP 4.5 standard, we expect the amount of such material to grow (see this [presentation on performance of the Clang OpenMP 4.5 implementation on NVIDIA gpus](#) for a status report as of 2016).

## 2.4 FPGA

Field Programmable Gate Arrays ([FPGAs](#)) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks.

Xilinx Ultrascale FPGAs and ARM processors have been proposed by the EuroEXA project as a new path towards exascale.

### 2.4.1 EuroEXA

EuroEXA is an EU Funded 20 Million Euro for an [ARM+FPGA Exascale Project](#) which will lead Europe towards exascale, together with [ExaNeSt](#), [EcoScale](#) and [ExaNoDe](#) projects, scaling peak performance to 400 PFLOP in a peak system power envelope of 30MW; over four times the performance at four times the energy efficiency of today's HPC platforms.

### 2.4.2 Feedback for software developers

Despite their high efficiency in performance and power consumption, FPGA are known for being difficult to program. [OpenCL for FPGA](#) is an example of programming language for FPGA which we strongly recommend, particularly considering that these new technologies will be soon available within the E-CAM community through the EuroEXA project.

## 2.5 Other significant developments

### 2.5.1 Non-volatile Memory

New Non-volatile Memory (NVM) technologies will strongly influence the memory hierarchy and represent a further step towards energy-aware measures for future HPC architectures. Resistive memories, i.e. memristors (like PCM, ReRAM, CBRAM and STT-RAMs), are an emerging class of non-volatile memory technology. Among the most prominent memristor candidates and close to commercialization are phase change memory (PCM), metal oxide resistive random access memory (RRAM or ReRAM), and conductive bridge random access memory (CBRAM).

The potential of NVM is to avoid access to the hard disk to save the data (unless a large amount is needed). The advantages are clear: lower power consumption than common RAM and higher bandwidth than hard disk.

## 2.6 Extreme-scale resources at the European level

### 2.6.1 Current PRACE Resources

As far as the Partnership for Advanced Computing in Europe (PRACE) initiative is concerned, the complete list of available resources are shown in Figure 4.

Access to PRACE resources can be obtained by application to the [PRACE calls](#).

Moreover, the Distributed European Computing Initiative (DECI) is designed for projects requiring access to resources not currently available in the PI's own country but where those projects do not require resources on the very largest

		<i>Curie TN</i>	<i>Irene - SKL</i>	<i>Irene - KNL</i>	<i>Hazel Hen</i>	<i>Successor of Juqueen</i>	<i>Marconi Broadwell</i>	<i>Marconi KNL</i>	<i>MareNostrum 4</i>	<i>Piz Daint</i>	<i>SuperMUC Phase 1</i>	<i>SuperMUC Phase 2</i>
System Type		Bullx	Bull Sequana	Bull Sequana	Cray XC40	tbd	Lenovo System NeXtScale	Lenovo System Adam Pass	Lenovo	Hybrid Cray xC50	IBM System x iDataPlex	Lenovo NeXtScale
Compute	Processor type	Intel SandyBridge EP 2.7 GHz	Intel Skylake EP 2.7 GHz	Intel Knights Landing	Intel Xeon E5-2680v3 (Haswell)	Multicore	Intel Broadwell	Intel Knights Landing	Intel Xeon Platinum 8160 2.1 GHz	Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores)	Intel Sandy Bridge EP	Haswell Xeon E5-2697 v3 (Haswell)
	Total nb of nodes	5 040	1656	684	7 712	tbd	504	3 600	3 456	5 320	9 216	3 072
	Total nb of cores	80 640	79 488	46 512	185 088	tbd	18 144	244 800	165 888	63 840	147 456	86 016
	Nb of accelerators/node	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	1 GPU per node	n.a.	n.a.
	Type of accelerator	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	NVIDIA® Tesla® P100 16GB	n.a.	n.a.
Memory	Memory / Node	64 GB	192 GB DDR4	96 GB DDR4 + 16 GB MCDRAM	128 GB	min. 96 GB	128 GB - DDR4	96 GB - DDR4 + 16 GB - MCDRAM	96 GB (200 nodes with 384GB)	64 GB	32 GB	64 GB
Network	Network Type	Infiniband QDR	TBA*	TBA*	Cray Aries	tbd	Intel Omni-Path Architecture 2:1	Intel Omni-Path Architecture 2:1	Intel Omni-Path Architecture	Cray Aries	Infiniband FDR10	Infiniband FDR14
	Connectivity	Fat tree	Fat Tree	Fat Tree	Dragonfly	tbd	Fat Tree	Fat Tree	Fat Tree	Dragonfly	Fat tree within island (512 nodes) pruned tree between islands	Fat tree within island (512 nodes) pruned tree between islands

Figure 4: List of PRACE Resources (as of 2017) with associated hardware characteristics.

(Tier-0) European Supercomputers or very large allocations of CPU. To obtain resources from the DECI program, applications should be made via the [DECI calls](#).

## 2.6.2 Extreme-scale Demonstrators

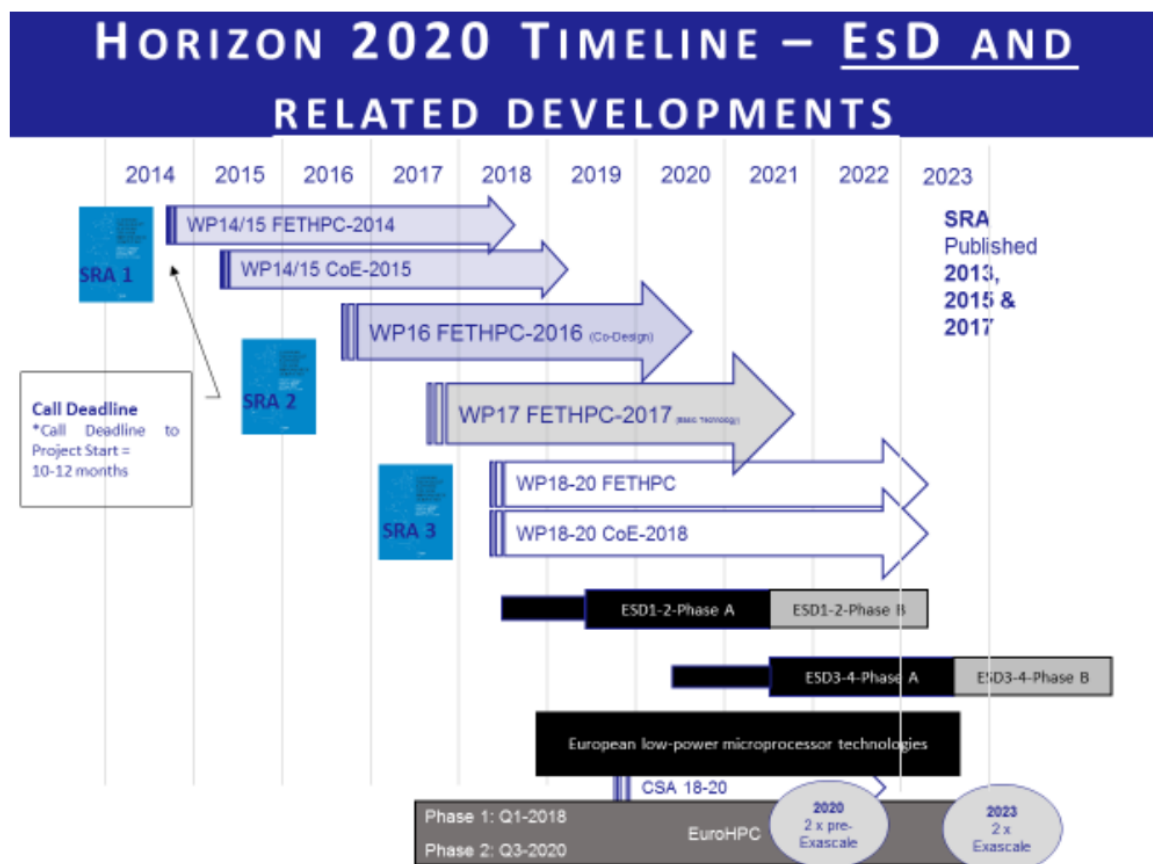


Figure 5: The European EsD developments mapped onto the timeline of the European Horizon 2020 research timeline

The first step in the European drive to exascale computing will be Extreme-scale Demonstrators that should provide pre-Exascale platforms deployed by HPC centres and used by Centres of Excellence for their production of new and relevant applications (for an implementation timeline see Fig. 5).

The technology scope of the demonstrators is being outlined by ETP4HPC initiative in their [Strategic Research Agenda](#) and is to be included in the EC LEIT-ICT 2018 calls. At project end, the EsDs will have a high *Technical Readiness Level* that will enable stable application production at reasonable scale.

### **2.6.3 Feedback for software developers**

While it is too early to know the hardware technology content of the EsDs, the technologies described already are very likely to feature in the demonstrators. Practically speaking one must then consider the software implementation implications, something that is covered in some detail in Section 3.2.

### 3 Software Needs

While the original scope of this section was intended to collect information from the E-CAM community that could be relayed back to hardware and software vendors, it is clear that the hardware developments described in Section 2 will greatly impact the software development practices of the E-CAM development community. For this reason, we go beyond this original scope and in addition we highlight the the language standards, runtime environments, workflows and software tools that can help E-CAM developers to deliver high quality, resilient software for current and next generation machines.

#### 3.1 Software needs as collected by the E-CAM Software Survey

The [E-CAM Survey of Application Software](#) was created to collect basic information about the software, libraries and applications that are frequently used by people in the extended CECAM community, with a view to trying to develop a support infrastructure for heavily-used software within E-CAM. It includes questions about the application; scientific area; parallelization techniques; the users familiarity with, and capabilities of, the application; the sizes of users typical resource requirements; and finally whether they are a user or developer of the application.

The survey was circulated to our complete set of mailing lists (including industrialists) and is distributed to all Extended Software Development Workshop (ESDW) participants. It is also available on the homepage of [CECAM website](#) since the middle of April 2017. To date, we have received 82 responses with 47 distinct application codes listed. We are conscious that the responses related to the survey are currently not sufficient to be representative of the entire community and will need to be extended. Nevertheless, we do an initial analysis of these results in this section and will update this information in future deliverables.

The Fig.6 shows that more than 60% of them simply use an open-source community code, while about 20% of them are open-source code developers with about 10% of them using a commercial code.

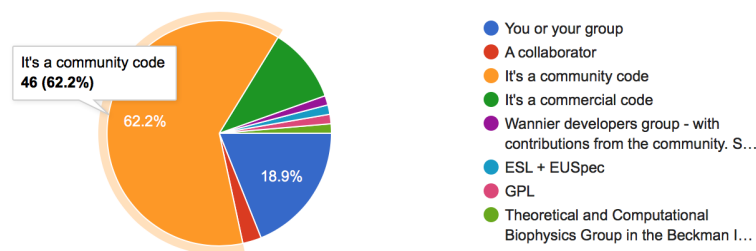


Figure 6: Question: Who develops the software?

Fig.7 shows that ~60% of the software of the survey is written in Fortran, with about ~40% in C/C++ and a further 27% using Python (or Python bindings).

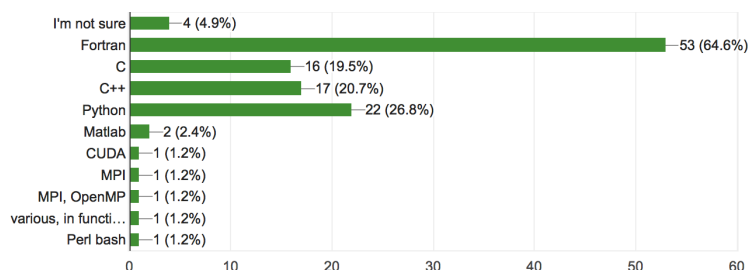


Figure 7: Question: The Programming Language used by the software.

Fig.8 shows that three quarter of the respondees use a code with an MPI implementation and half with an OpenMP implementation. One third of them are under Hybrid MPI/OpenMP mode. GPU capabilities are available to one third of the applications.

Fig.9 shows that 70% of respondees get their publication-quality results through the use of resources at HPC centres.

Fig.10 shows that almost half them use less than 128 cores for simulation and less than 20% use 1024 cores or beyond. This clearly indicates that only a small subset of respondees have requirements that potentially extend to extreme-

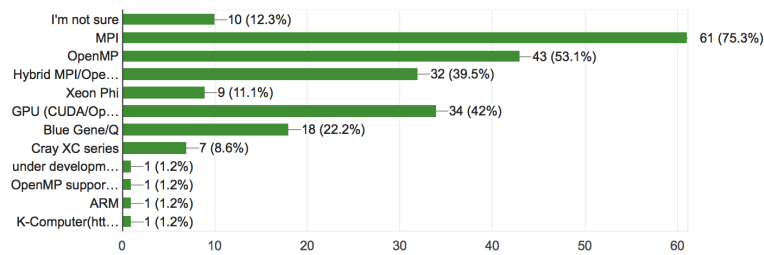


Figure 8: Supported (parallel) software capabilities

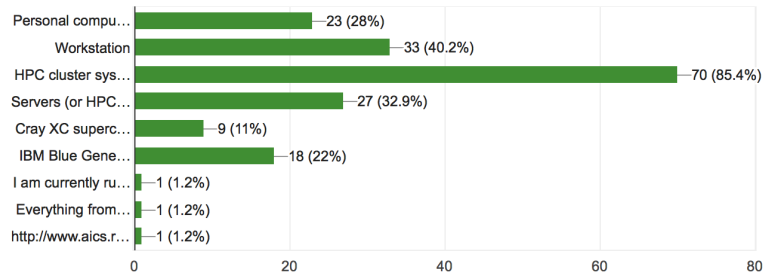


Figure 9: Most common hardware used for generating publication-quality results

scale resources (although the potential use case for High Throughput Computing is currently not covered by the questionnaire).

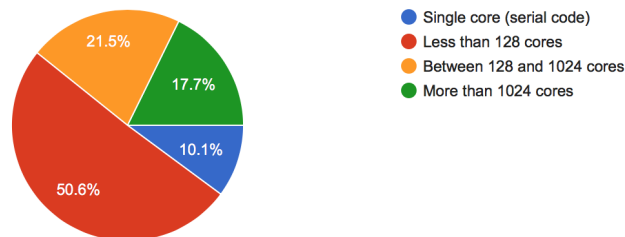


Figure 10: Max number of cores used per run

Fig.11 shows that more than 80% of them get their jobs finished within 24 hours (or have restart capabilities in the application).

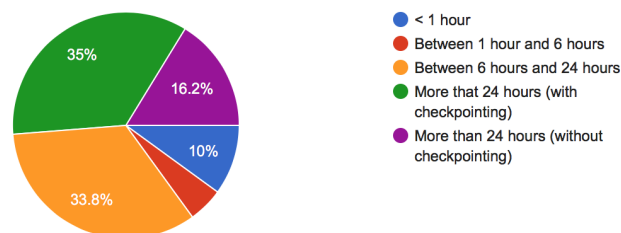


Figure 11: Max number of hours used per run

While the survey is not large enough to draw strong conclusions, we can extract some trends with respect to the **softwares needs** of the E-CAM community:

- Most of the users use open-source community codes and don't necessarily have the capacity to maintain/develop them. The needs for commercial codes represent only a very small portion of the community.
- Since the majority of the codes included are written with Fortran, any contributions that our community make to these codes will necessarily have to be written in Fortran. However, the development and use of Python bindings becomes increasingly popular. Teaching the community how to use (and create) efficient Python bindings and using Python for non-performance-critical components of their workflow should be very helpful for the community (and also ease any future transition other programming models).



- MPI, OpenMP, GPU (CUDA/OpenACC) are the main techniques currently used to enable code parallelisation. Training in these techniques, in particular targeting "accelerator" technologies (GPU/Xeon-Phi), are needed.
- More than 80% of them use HPC centres for producing science. Guiding them to get access to PRACE resources should be very helpful *but* the scalability of their resource requirements will need to be demonstrated. Given that *ensemble* calculations are very common across E-CAM, the inherent scalability of such workflows need to be emphasised to the community.

### 3.1.1 Feedback for hardware and software vendors

As the survey currently stands (and the results should be considered limited despite 83 responses), the indications are that the E-CAM community relies heavily on a diverse set of community-developed applications. The most commonly occurring, at present, are GROMACS, QuantumESPRESSO, Yambo, CP2K, CPMD, SIESTA, OpenMM, DL\_POLY, DL\_MESO and ESPRESSO++. Only two commercial applications have occurred a number of times: Turbomole and VASP.

There is still a heavy reliance in the community on Fortran, meaning that support for this language will be essential on future systems if people are to be expected to migrate their current workloads there.

The typical production use cases observed in the community do not necessarily require extreme scalability. This is not to say, however, that the community does not require extreme-scale resources. Vendors should be wary that ensemble calculations are an important part of the workflow across the E-CAM community and that this has potential cost implications for resources suitable to this case (for example, this may significantly impact the design of cost-effective network topologies for E-CAM workloads).

There are many cases E-CAM developers are interfacing at a relatively abstract level with community codes. Therefore, the appetite for novel technologies is typically reliant on an implementation (and sufficient documentation) within these codes. This is both an opportunity and a burden: a subset of community applications ported to novel technologies may bring large user communities to such resources *but* such applications are so feature-rich (and therefore monolithic) that this may be exceedingly difficult to implement satisfactorily in practice.

## 3.2 Programming Paradigms

### 3.2.1 C++17

C++17 is the informal name for the next revision of the ISO/IEC standard for the C++ programming language. Its Draft International Standard can be found [here](#), and the final standard is expected to be published by the end of 2017.

The previous C++ versions show very limited parallel processing capabilities when using multi/many core architectures. This situation will change with the C++17, in which the parallelised version of [Standard Template Library](#) is included. The STL is a software library for C++ programming which has 4 components: Algorithms, Containers, Functors and Iterators. *"Parallel STL advances the evolution of C++, adding vectorization and parallelization capabilities without resorting to nonstandard or proprietary extensions, and leading to code modernization and the development of new applications on modern architectures."*[2]

A [multi-threading programming model for C++](#) is supported since C++11.

E-CAM is very aware of the parallelism potential of C++ and is co-hosting a PRACE Advanced Training Centre (PATC) workshop entitled "High-performance scientific computing in C++" to promote the use of C++ in E-CAM.

### 3.2.2 Fortran 2015

[Fortran 2015](#) is a minor revision of Fortran 2008 (which was when Fortran became a Partitioned Global Address Space (PGAS) language with the introduction of [coarrays](#)). The revisions mostly target additional parallelisation features and increased interoperability with C.

Most Fortran-based software E-CAM sees in practice is implemented in Fortran 95 and there appears to be little awareness of the parallel features of the latest Fortran standards. E-CAM is considering organising a workshop that addresses this lack of awareness (similar to the "[Software Engineering and Parallel Programming in Modern Fortran](#)" held at the Cranfield University).

It should be noted that [compiler support for the latest Fortran standards](#) is limited. This is most likely due to the fact that Fortran is not widely used outside of the scientific research (limiting its commercial scope).



### 3.2.3 The (potential) role of Python

Given that it is an interpreted language (i.e., it is only compiled at runtime), Python is not usually discussed much in the HPC space since there is limited scope for control over many factors that influence performance. Where we are observing a lot of growth is where applications are being written in languages like C++ under the hood but are intended to be primarily used via their Python interfaces.

This is a valuable, and user friendly, development model that allows users to leverage Python for fast prototyping while maintaining the potential for high performance application codes.

A warning to would be users: [Python 2 will stop being developed in 2020](#) so please make sure that your code is Python3 compliant.

### 3.2.4 Open Standards

We describe here some of the open standards that are most likely to be leveraged on next generation HPC resources.

#### MPI

Now more than 25 years old, Message Passing Interface (MPI) is still with us and remains the de facto standard for internode communication (though it is not the only option, alternatives such as [GASNet](#) exist). [MPI-3.1](#) was approved by the MPI Forum on June 4, 2015. It was mainly an errata release for MPI 3.0 which included some important enhancements to MPI:

- Nonblocking collectives
- Sparse and scalable irregular collectives
- Enhancements to one-sided communication (very important for extreme scalability)
- Shared memory extensions (on clusters of SMP nodes)
- Fortran interface

Maintaining awareness of the [scope of past and future updates to the MPI standard](#) is important since it is the latest features that target the latest architectural developments.

#### OpenMP

OpenMP is also 20 years old and remains the most portable option for on-node workloads. The standard has introduced new features to deal with increasing node-level heterogeneity (device offloading, such as for the GPU, in particular) and varied workloads (task level parallelism).

From GCC 6.1, OpenMP 4.5 is fully supported for C and C++ (with Fortran support coming in the GCC 7 series). The [level of OpenMP support among other compilers](#) varies significantly.

#### OpenACC

[OpenACC](#) (for open accelerators) is a programming standard for parallel computing developed by Cray, CAPS, Nvidia and PGI. The standard is designed to simplify parallel programming of heterogeneous CPU/GPU systems. Since the paradigm is very similar to the latest OpenMP specs, a future merger into OpenMP is not unlikely.

It should be noted that CUDA (with the [nvcc compiler](#)) is still the most commonly used (and highest performing) library for programming NVIDIA GPUs.

#### OpenCL

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs, see Section 2.4 for the extreme relevance of this) and other processors or hardware accelerators.

OpenCL 2.2 brings the OpenCL C++ kernel language into the core specification for significantly enhanced parallel programming productivity. When releasing OpenCL version 2.2, the Khronos Group announced that OpenCL would be merging into [Vulkan](#) (which targets high-performance realtime 3D graphics applications) in the future, leaving some uncertainty as to how this may affect the HPC space.

### 3.2.5 Runtime System Approaches

As noted already, programming paradigm standards are moving forward to adapt to the technologies that we see in the market place. The complexity of the hardware infrastructure (as outline in Section 2) necessarily brings complexity to the implementation of the programming standards.

There are number of programming models that leverage runtime systems under development. They promise to abstract away hardware during the development process, with the proviso that tuning at runtime may be required. Our experience to date with these systems is limited so we simply provide a list of three such systems here (which is certainly not exhaustive) in no particular order:

- **HPX**, a C++ Standard Library for concurrency and parallelism. The goal of the HPX project is to create a high quality, freely available, open source implementation of **ParalleX** concepts for conventional and future systems by building a modular and standards conforming runtime system for SMP and distributed application environments. (Most recent release: v1.0, April 2017)
- **Kokkos** implements a programming model in C++ for writing performance portable applications targeting all major HPC platforms. For that purpose it provides abstractions for both parallel execution of code and data management. Kokkos is designed to target complex node architectures with N-level memory hierarchies and multiple types of execution resources. It currently can use OpenMP, Pthreads and CUDA as backend programming models. (Most recent release: v2.04.04, 11 Sept 2017)
- **OmpSs** is an effort to integrate features from the StarSs programming model developed at Barcelona Supercomputing Centre (BSC) into a single programming model. In particular, the objective is to extend OpenMP with new directives to support asynchronous parallelism and heterogeneity (devices like GPUs). However, it can also be understood as new directives extending other accelerator based APIs like CUDA or OpenCL. The OmpSs environment is built on top of BSCs Mercurium compiler and Nanos++ runtime system. (Most recent release: v17.06, June 2017)

### 3.2.6 Feedback for software developers

Awareness of the latest standards and the status of their implementations are critical at all times during application development. The adoption of new features from standards are likely to have large impact on the scalability of application codes precisely because it is very likely that these features exist to target the scalability challenges on modern systems. Nevertheless, you should be aware that there can be very long gaps between the publication of a standard and the implementation in compilers (which is frequently also biased by who is pushing which standard and why: Intel pushes OpenMP because of their Xeon Phi line, NVIDIA who now own PGI pushes OpenACC because of their GPUs, AMD pushed OpenCL for their own GPUs to compete with CUDA). The likelihood of there being a single common (set of) standards that performs well on all architectures is not high in the immediate future. For typical developers that we see in E-CAM, MPI+OpenMP remains the safest bet and is likely to perform well, as long as the latest standards are used.

More disruptive software technologies (such as GASNet) are more likely to gain traction if they are used by popular abstraction layers (which could be PGAS languages, runtime systems or even domain specific languages) "under the hood". This would make the transition to new paradigms an implementation issue for the abstraction layer. Indeed, given the expected complexity of next generation machines, new programming paradigms that help remove the performance workload from the shoulders of scientific software developers will gain increasing importance.

As you may have noticed in the previous discussion, the computer scientists developing these abstractions are working mostly in C++, and the implementation of new standards in compilers is also seen first for C++. From a practical perspective this has some clear implications: if you want to access the latest software technologies then you had better consider C++ for your application. This may appear harsh given that the Fortran standard has clear capabilities in this space, but it is a current reality that cannot be ignored. Also, given that the vast majority of researchers will eventually transition to industry (because there simply aren't enough permanent jobs in academia) it is more responsible to ensure they have programming expertise in a language that is heavily used in the commercial space. Finally, the ecosystem surrounding C++ (IDEs, testing frameworks, libraries,...) is much richer because of it's use in industry and computer science.

Taking all of the above into consideration, if you are starting out with an application we would distil the discussion into the following advice: prototype your application using Python leveraging the Python APIs to the libraries you need; write unit tests as you go; and, when you start doing computationally intensive work, use C++ with Python interfaces to allow you to squeeze out maximal performance using the latest software technologies.

## 4 Interactions between end-users, developers and vendors

This section has been produced primarily based on some key input from the [E-CAM Extreme-Scale State-of-the-Art Workshop](#) that took place on 6-7 July 2017 in Barcelona. The event represents a gathering of key E-CAM personnel (code developers, application users) as well as representatives from other European projects (e.g. PRACE, ETP4HPC, Center of Excellence (CoE)s including POP, NoMAD, EoCoE and MAX) and the E-CAM user community. There were discussion sessions on both days which includes representatives of *hardware vendors* (a term which we loosely interpret to include EU-funded hardware development projects which clearly include industrial partners). *Software development* was represented by the presence of 5 CoEs, PRACE and developers of high profile applications in the E-CAM space.

One of the main objectives of the workshop was to inform, as well as instigating discussions with, E-CAM developers and end-users about some of the key HPC issues and challenges in the field of molecular and atomistic simulations, with particular emphasis on forward-looking exascale topics. Some of the main issues and topics include the following, and are discussed in more details at subsequent sections as indicated in parentheses:

- The European exascale roadmap (Section 4.1)
- Emerging hardware architectures relevant to exascale computing (Section 4.2).
- Exascale challenges in exploiting massive parallelism (Section 4.3).

Finally, some of the outcome that arose from discussions at the workshop along with future recommendations and directions for E-CAM are outlined in Section 4.4.

### 4.1 The European exascale roadmap

The European HPC Technology Platform, ETP4HPC, is an industry-led think-tank comprising of European HPC technology stakeholders: technology vendors, research centres and end-users. The main objective of ETP4HPC is to define research priorities and action plans in the area of HPC technology provision (i.e. the provision of supercomputing systems). It has been responsible for the production and maintenance of the [European HPC Technology Strategic Research Agenda \(SRA\)](#), a document that serves as a mechanism to provide contextual guidance to European researchers and businesses as well as to guide EU priorities for research in the Horizon 2020 HPC programme, i.e. it represents a roadmap for the achievement of European exascale capabilities.

We have had numerous discussions of the E-CAM community software needs through our exchanges with ETP4HPC during the course of our contributions to the SRA. The particular contribution from our discussion related to the software needs for exascale computing within the ETP4HPC SRA report is shown in the paragraphs below:

E-CAM has not committed itself to a single set of applications or use cases that can be represented in such a manner, it is instead driven by the needs of the industrial pilot projects within the project (as well as the wider community). Taking into consideration the CECAM community and the industrial collaborations targeted by E-CAM, probably the largest exa-scale challenge is ensuring that the skillsets of the application developers from academia and industry are sufficiently up to date and are aligned with programming best practices. This means that they are at least competent in the latest relevant language specification (Fortran 2015, C++17,...) and aware of additional tools and libraries that are necessary (or useful) for application development at the exa-scale. For application users, this means that they have sufficient knowledge of architecture, software installation and typical supercomputing environment to build, test and run application software optimised for the target.

While quite specific "key applications" are under continuous support by other CoEs, this is not the current model of E-CAM. E-CAM is more likely to support and develop a software installation framework (such as [EasyBuild](#)) that simplifies building the (increasingly non-trivial) software stack of a particular application in a reliable, reproducible and portable way. Industry has already shown significant interest in this and E-CAM is particularly interested in extending the capabilities of EasyBuild to EsD architectures, performance analysis workflows and to new scientific software packages. Such an effort could easily be viewed as transversal since such developments could be leveraged by any other CoE.

One important focus of the SRA is the development of the "Extreme-Scale Demonstrators" (EsDs) that are vehicles to optimise and synergise the effectiveness of the entire HPC H2020 programme (see Figure 12 for a list of European HPC technology projects involved, including E-CAM), through the integration of R&D outcomes into fully integrated HPC system prototypes.

The work in developing the EsDs will fill critical gaps in the H2020 programme, including the following activities:



Figure 12: The European HPC Technology Projects within the European HPC Eco-system.

- Bring technologies from FET-HPC projects closer to commercialisation.
- Combined results from targeted R&D efforts into a complete system (European HPC technology ecosystem).
- Provide the missing link between the three HPC pillars: technology providers, user communities (e.g. E-CAM) and infrastructure

As one of the CoEs, E-CAM should aim to provide insight and input into the requirements of future exascale systems based on lessons learnt from activities within E-CAM (e.g. software development and relevant performance optimisation and scaling work). This would entail further knowledge and understanding within E-CAM on exploiting current multi-petaflop infrastructures, what future exascale architectures may look like, as well as interaction and close collaboration between E-CAM and other projects (i.e. the projects shown in Figure 12); these are also covered in subsequent sections of this paper.

## 4.2 Emerging hardware architectures relevant to exascale computing

The European Commission supports a number of projects in developing and testing innovative architectures for next generation supercomputers, aimed at tackling some of the biggest challenges in achieving exascale computing. They often involve co-design involving HPC technologists, hardware vendors and code developer/end-user communities in order to develop prototype systems. Some of these projects include:

- The DEEP (Dynamic Exascale Entry Platform) projects (DEEP, DEEP-ER and DEEP-EST)
- The Mont-Blanc projects (Mont-Blanc 1, 2 and 3)
- The PRACE PCP (Pre-Commercial Procurement) initiative

We discuss further the first two of these since they were explicitly discussed during the workshop. E-CAM is also now represented in the PRACE PCP (Pre-Commercial Procurement) initiative.

### 4.2.1 The DEEP projects – overview of the architecture

The DEEP projects have been developing a modular approach to extreme-scale computing via an innovative “cluster booster architecture”. In the initial DEEP and DEEP-ER projects, they have successfully integrated a standard, Infini-band cluster using Intel Xeon nodes (Cluster Nodes) that is connected to a novel, highly scalable Booster consisting

of Intel Xeon Phi co-processors (Booster Nodes) connected via an underlying EXTOLL high-performance 3D torus network (see Figure 13).

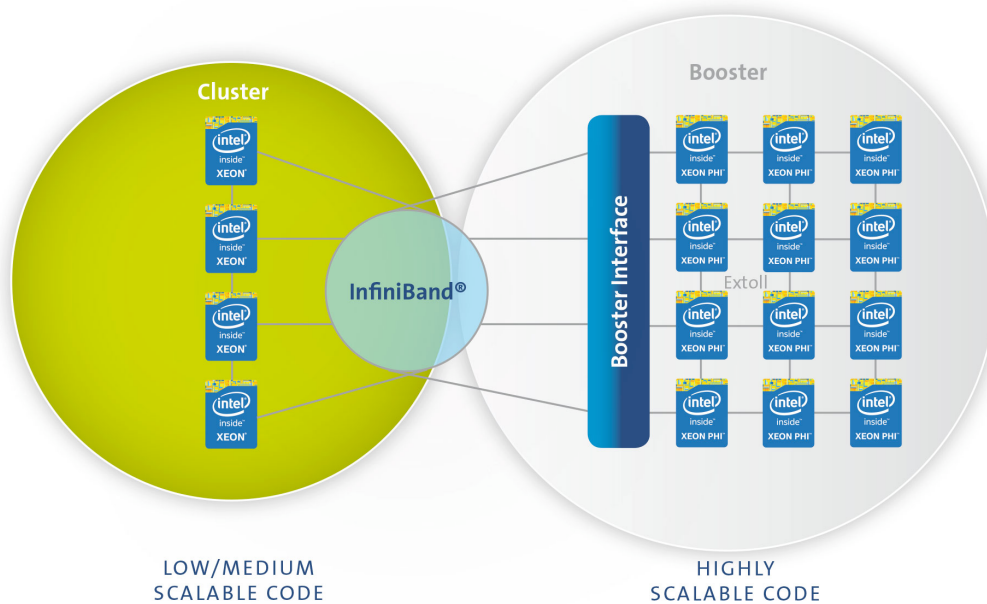


Figure 13: The DEEP hardware architecture.

This architecture enables high throughput and scalability on the Booster component, aimed at highly scalable code execution that supports parallelism via proven and standards-based programming models such as MPI and OmpSs. Parts of the code with more limited scalability (e.g. owing to complex control flow or data dependencies) are executed on the Cluster component, with transparent bridging of both interconnects to ensure high-speed data transfer. Innovative aspects:

- on-node memory, network attached memory.
- Novel fabric, uses ASIC-based network controller designed and produced by EXTOLL.
- Self booting Xeon Phi nodes.

Special optimisations:

- Resilience
- I/O: SIONlib, Exascale10

The project makes heavy use of co-design with applications selected from the beginning of the project. In the next phase, the DEEP-EST project will generalise the cluster-boosters approach by extending the modular concepts of the current system. Apart from cluster and booster (Intel Xeon Phi) components, additional components will be developed with architectures adapted for data-intensive and data analysis applications.

#### 4.2.2 The Mont-Blanc projects

The Mont-Blanc projects focus on developing prototype HPC systems based on energy-efficient embedded technologies. It has developed several ARM-based computational platforms, leveraging some of the technologies that have emerged from the highly-competitive low-power mobile computing market, along with the requisite software stack to facilitate development and execution of complex parallel codes.

During the initial two phases of the project, which begun in 2011, it has examined a range of embedded and mobile processor architectures. The activities have included the development of small-scale, mini-clusters built using a variety of node architectures, e.g. an Nvidia Tegra K1 SoC with ARM Cortex processors and Kepler GPU, or a Samsung Exynos 5 SoC combined with Cortex processors and the ARM Mali T-604 GPU. The first official prototype for the project has since been built and operational since 2015. This system consists of 1,080 compute cards where each card or node consists of a Samsung Exynos 5 Dual SoC, powered by ARM Cortex-A15 processors (2x cores) and an ARM Mali-T604 GPU along with 4GB of LPDDR3-1600 memory, 64 GB of local storage on microSD and 10 GbE networking bridged via USB 3.0.



The project is currently in its third phase, coordinated by Bull, the Atos brand for technology products and software. It adopts a co-design approach to ensure that the hardware and system innovations can be translated into HPC application performance and energy efficiency. Here, a new demonstrator system, named Dibona, is being built based on the 64-bit ThunderX2 processors from Cavium® that implements the ARM v8 instruction set. The full system will eventually be composed of 48 compute nodes with 96 ThunderX2 CPUs or 3,000 cores in total.

Apart from hardware developments, the project has also invested significant efforts in developing a software ecosystem for ARM-based architectures in HPC. These include the porting and extension of performance tools to work on ARM-based platforms (e.g. Score-P, Scalasca, and BSC's performance analysis toolset including Extrae, Paraver and Dimemas). The Mont-Blanc project has also contributed to the OmpSs parallel programming model to support ARMv8 64-bit processors, further enabling it to exploit multiple cluster ARM-based nodes along with transparent application check-pointing for fault tolerance. Furthermore the project also developed a novel power measurement infrastructure in order to determine power efficiency that is key to exascale. This takes high-frequency measurements of power consumption at the granularity of a single compute node, and up to the entire prototype system.

#### 4.2.3 Other discussions with hardware and software vendors

E-CAM people participated the 2017's Teratec forum and discussed with software and hardware vendors. Nvidia presented the top ten HPC applications accelerated by GPU, among them, Gaussian, Gromacs, NAMD, VASP, AMBER are CECAM's community codes. This shows a real efforts done by the community people.

Bull presented to us the prototype of Exascale-focused ARM-based Sequana X1310 supercomputer. And the Mixed Intel Xeon-Xeon Phi based Sequana X1000 machine will be available at Spring 2018 in French TGCC center with up to 20 Petaflops. We've also discussed with researchers from the CEA-UVSQ-Intel joint Exascale Computing Research Lab. We're very interested in their exascale efforts for one of the CECAM's community code, Abinit. They presented their work via poster at the forum. They used the MAQAO and CERE tools to analyse the Abinit code performance and use the MPC framework to facilitate the paralleling programming. In collaboration with the Abinit developers, they run directly this code on Intel specific architecture by developing an *hardware abstraction layer*. This may provide a good example, or possible orientation, for the other important community codes which have a real need to go to exascale.

### 4.3 Exascale challenges in exploiting massive parallelism

This section is a summary of various talks presented at the workshop that are related to the discussion of exascale challenges in exploiting massive parallelism. The presentations include those from application developers, code developers and users within E-CAM and beyond, and representatives from other CoEs.

#### 4.3.1 "Large scale electronic structure calculations with CP2K" - Jurg Hutter

CP2K is a popular, freely available software package written in Fortran and used for atomistic and molecular simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. It provides a framework for different modelling methods such as DFT and wave-function methods (two methods that are highlighted in the talk) as well as many others. The talk focused on some of the large-scale (towards exascale) challenges and contributions from the CP2K development team. The talk introduced the audience to the use of sparse matrix libraries for KS-DFT simulations for large scale computations. By using a blocked compressed sparse row format, followed by load balancing using randomized indices, Cannon's algorithm can then be applied, with computations optionally offloaded to accelerators, to achieve better scaling (up to  $\frac{1}{\sqrt{N}}$ , which could be a bottleneck for up to 1 million processes). Some of the results shown included the enhanced performance of a mini-app (using the LIBCUSMM library) on the NVidia Pascal architecture relative to its previous Kepler architecture, with up 20-50% increase in flops. In another example dominated by matrix multiplications, the performance figures (Fig.14) showed comparable performance between a GPU node (on the Daint system) compared to a (*less expensive*) KNL node. More recent developments try to address the bottleneck of Cannon's method, including the use of remote memory access to replace point-to-point communication, and using a 2.5-dimensional variant of Canon's algorithm at the expense of higher memory usage. These have resulted in performance gains of 1.2-1.4x on large scale systems of up to 50,000 cores and 4,000 GPUs. Application of these methods have enabled researchers to carry out large-scale energy calculations on large systems, e.g. 1 million atoms of a STM virus, albeit only for a few femtoseconds.

For wavefunction methods, the talk introduced a number of Resolution of Identity (RI) approaches based on Gaussian and plane waves (GPW), including RI-dRPA, RI-MP2, RI-G0W0. The switch to new algorithms using sparse tensor representations and overlap metric have allowed scaling to thousands of nodes for large systems.

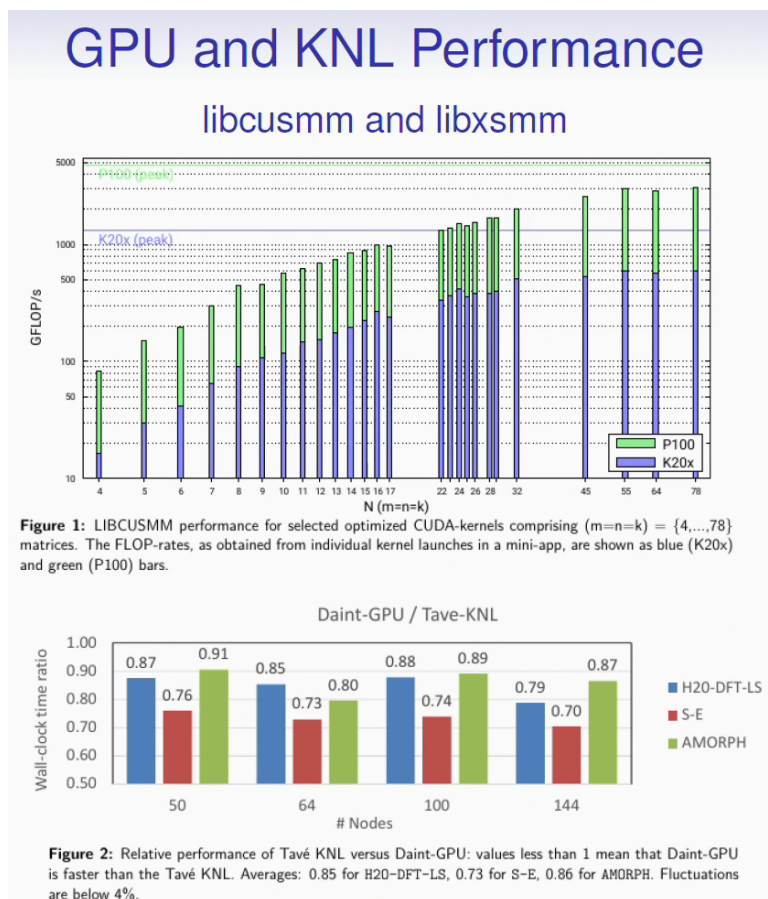


Figure 14: CP2K Presentation: GPU and KNL Performance

In future, this [DBCSPR library](#) can be generalised for sparse tensor algebra, meaning that tensor contractions can be mapped to matrix multiplications. This tensor library is still in development at the time of this report.

#### 4.3.2 “Scalability of Path Sampling Simulations” – David Swenson

Path sampling methods, such as transition path sampling and transition interface sampling, are Monte Carlo simulations in the space of trajectories. They provide powerful tools for studying topics such as chemical reactions, crystallization, binding processes, and conformational changes in biomolecules. Path sampling software can leverage the high-performance capabilities of existing tools for molecular dynamics simulation and analysis, as well as producing additional scalability by running multiple trajectories in parallel. As a part of E-CAM, the OpenPathSampling (OPS) software package has been extended to support widely-used and highly-scalable molecular dynamics engines (e.g. OpenMM, LAMPPS).

The talk gave an overview of using OPS to monitor the trajectories space in order to handle rare events. This is carried out using a Python library that interface with the simulation engine directly (for those with Python APIs such as OpenMM, LAMMPS) or indirectly (e.g. file-based interaction for Gromacs). In terms of performance, the overhead from OPS was shown to be less than the variance due to Gromacs’s performance tuning. In realistic simulations, the cost of the underlying dynamics dominates OPS’s performance, i.e. its scalability (towards exascale) is comparable to that of the underlying engine itself such as Gromacs. Finally, additional functionalities are being developed for the library, including support for running simultaneous trajectories (see Section 4.4.1).

#### 4.3.3 “PaPIM: A Code for (Quantum) Time Correlation Functions” – Momir Malis

The Phase Integration Method (PIM) is a novel approximate quantum dynamical technique developed for computing systems time dependent observables. PIM employs an algorithm in which the exact sampling of the quantum thermal Wigner density is combined with a linearized approximation of the quantum time propagators represented in the path integral formalism that reduces the evolution to classical dynamics.

The quantities of interest can then be computed by combining classical molecular dynamics algorithms with an original generalised Monte Carlo scheme for sampling the initial conditions and averaging them over all sampled points in phase space. Because all trajectories are mutually independent, the algorithm can be efficiently parallelised. The work within E-CAM have resulted in novel Fortran90 PaPIM code parallelised using MPI, or PaPIM. While PaPIM has shown good scalability up to several thousand cores, with other parts of the code also having further potential for parallelisation. These include computing the polymer chains in parallel, parallel approaches for individual potential energy calculations (already being examined by other groups) and the decoupling of classical trajectory propagation. Finally, since PIM requires a multitude of input parameters, a large-scale parallel search for optimal sets of PIM sampling parameters will be useful and are planned as future work for the project.

#### 4.3.4 “Porting DL\_MESO\_DPD on GPUs” – Jony Castagna

DL\_MESO is a general purpose mesoscale simulation package developed by Michael Seaton written in Fortran90 and C++ which supports both Lattice Boltzmann Equation (LBE) and Dissipative Particle Dynamics (DPD) methods. The talk gave an overview of some of the challenges in porting the relevant Fortran90 code to Nvidia GPUs.

One of the main problems first encountered was the random memory access pattern, caused by particle locations being stored in a contiguous manner, which can be fixed by re-organising the cell-linked array to ensure coalescent memory access. The GPU port, when executed on a Tesla P100 GPU, showed weak scaling of up to 4x speed-up over an Intel Ivy Bridge 12-core processor (See Fig.15); this is some way off the theoretical ideal speed-up of 12x due to clustering of particles and hence load imbalance amongst threads (which is to be addressed in future work).

One key recommendation from this talk, which may be applicable for GPU porting work in general, is to maintain two versions of the code, one codebase for the GPU while the other in plain Fortran/C, and that new physics should only be introduced into the latter before porting work to the GPU.

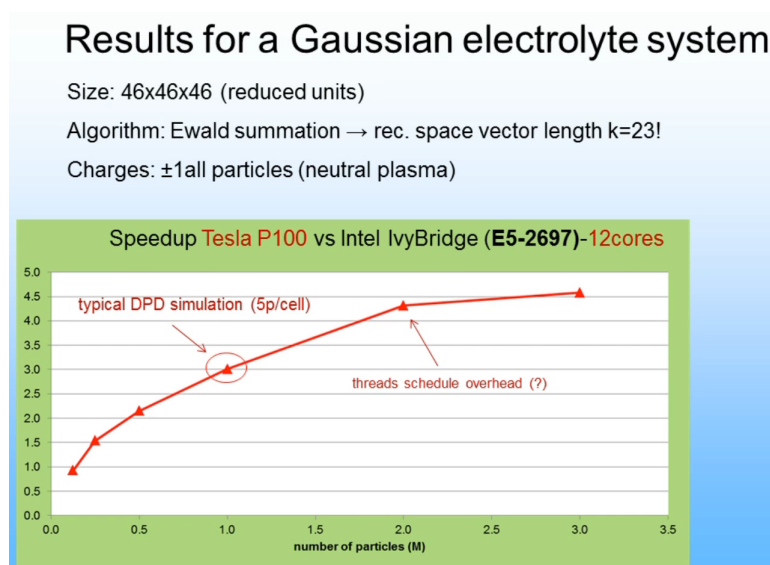


Figure 15: DL\_MESO: GPU porting results

#### 4.3.5 “MP2C: Multiple-particle collision dynamics on massively parallel computers” – Godehard Sutmann

The talk focuses the scalable implementation of MPC or MP2C for massively parallel computers. MP2C is a grid-free method for mesoscale simulation of hydrodynamic media mainly based on the concept of particles, bridging the gap between microscopic simulations on the atomistic level and macroscopic calculations on the continuum level by solving the Navier-Stokes equations in different types of discretisation. With a small set of parameters (particle density, scattering angle, mean free path of a particle), it is possible to reproduce hydrodynamic behaviour and map collective interactions to a local particle based algorithm.

The talk covered the hybrid domain partitioning methods used by MP2C to distribute work to the processors, including typical communication patterns and load balancing issues. This enables a coupling between molecular dynamics (MD) simulations and a mesoscopic solvent, taking into account hydrodynamic interactions between solutes. Both MPC and MD computations have demonstrated strong scaling on the JUGENE system. It then went on to become one of the first codes to demonstrate scalability across the whole JUQUEEN system (458k processor cores compared



to 262k cores on the JUGENE, see Fig. 16) MP2C has also been adopted as a benchmark for parallel I/O using the SionLib I/O library, which have been able to achieve I/O reads of >100 GB/s and writes of 60 GB/s.

The final part of the talk discussed in more details regarding load balancing issues, caused by highly heterogeneous particle distribution, meaning inefficient use of the HPC resources (i.e. some cores will be busy while others left idle). The solutions include further distribution of workloads at the expense of more data traffic and communication, and adjustment of the volume of domains at the expense of more complex geometries and communication patterns. These approaches to address load imbalance play a pivotal role in the code's scalability.

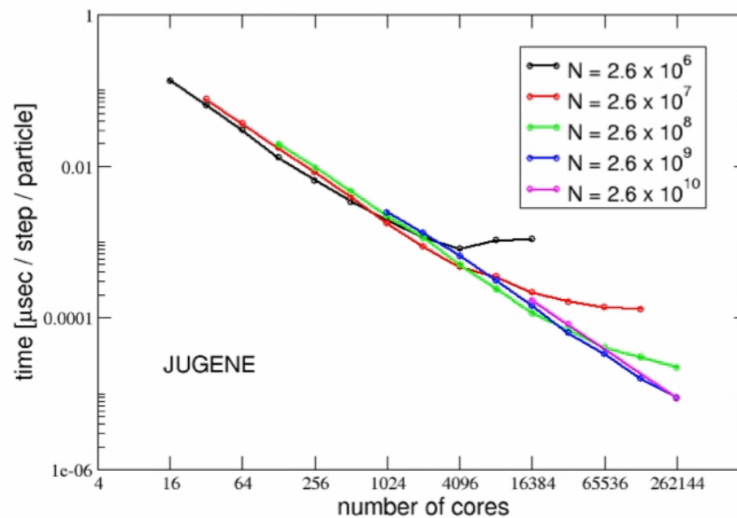


Figure 16: MP2C scaling on JUGENE

#### 4.3.6 “POP – understanding applications and how to prepare for exascale” – Jesus Labarta

The POP CoE projects aims to apply common analysis methodologies to a large base of customer codes to increase the awareness of fundamental issues that limit the scalability and performance of HPC applications. The project offers assessment of application performance and helps customers of the POP service to refactor applications in directions that maximize the performance and scaling revenue at minimum cost.

Due to increasingly complexity and variability of novel HPC architectures, there seems to be a growing divergence between the ways systems behave, and the way developers expect them to behave. Hence the promoted vision of POP is to decouple the underlying hardware system from the application (where the science happens) via different software layers (e.g. programming model layer and runtime layer, Fig.17). An example of activities already being undertaken at the programming model layer includes the StarSs paradigm (encompassing OmpSs that have developed features being incorporated into OpenMP). This is also discussed in Section 3.2.5.

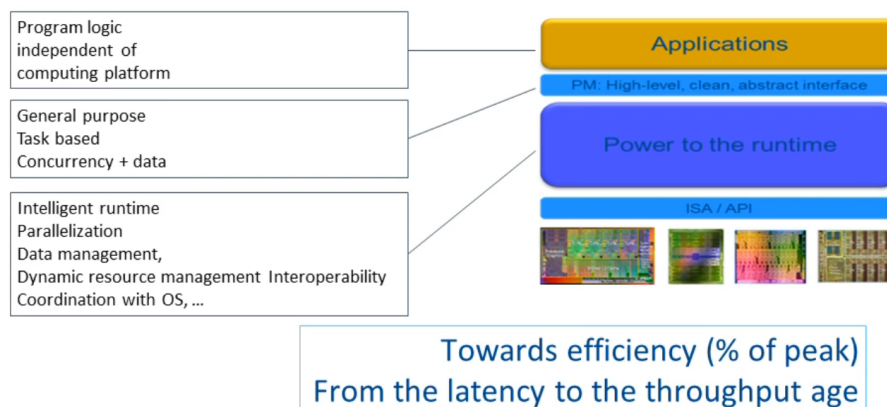


Figure 17: POP: Different software layers

The POP CoE offers three levels of services: application performance audit, application performance plan and proof-of-concept software demonstrators. Its target customers include code developers (looking to assess performance

and behaviour), application users (looking for potential improvements for specific conditions/set-ups), infrastructure operators (looking for information for allocation purposes or training of support staff) and vendors (looking to carry out benchmarking or system design guidance). Based on a hierarchy of fundamental performance factors (Fig. 18), the presentation provided a number of case studies where POP has improved the efficiency and scalability of applications it has evaluated. In total, it has analysed well over 50 codes.

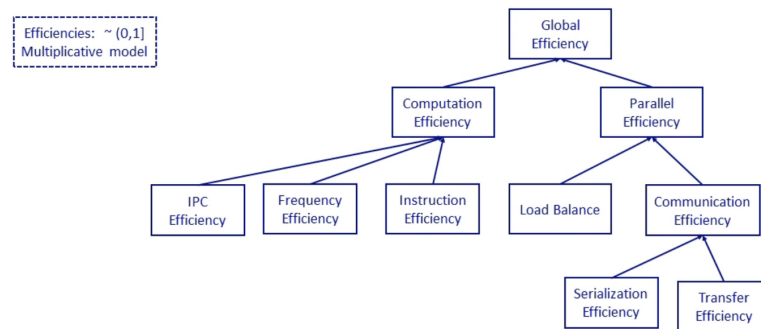


Figure 18: Fundamental performance factors in evaluating performance

#### 4.3.7 “Exascale challenges: the view from MaX” – Carlo Cavazzoni

For the [MaX CoE](#) (HPC for materials research and innovation), the goal of enabling the exascale transition involves the modernisation of community codes, making them ready to exploit future exascale system for material science simulations. The presentation initially focused on some of the exascale challenges. With Dennard scaling – constant power density despite transistors getting smaller – compute core frequency and performance no longer follow Moore’s Law. A key issue for compute performance is how it can be increased with the least amount of power. In order to achieve this, chips that have been designed for maximum performance for all possible workloads (i.e. conventional CPUs) are increasingly being supplemented by new chips designed for maximum performance in a small set of workloads (e.g. many-core technologies).

It was shown that future exascale systems (with current technologies) will necessarily have  $10^9$  Floating Point Units (FPUs). This may be configured as  $10^5$  FPUs in  $10^4$  nodes, or  $10^4$  FPUs in  $10^5$  nodes. In either case, the use of MPI to distribute code to  $10^4$  or  $10^5$  nodes may not be the main issue, but how applications make use of the  $10^4$  or  $10^5$  FPUs, i.e. many-core technologies, within single nodes. Furthermore, exascale is not only about scalability and flops performance, but moving data in and out of  $10^9$  FPUs poses tremendous challenges based on current technology. In addition, effective exploitation of exascale will have to contend with heterogeneous architectures and deep memory hierarchies compared to today, and considerably more attention paid to workloads and workflows.

As an example of the CoE’s activities on applications, some of the core modules from the Quantum Espresso software package have been factored out and transformed into “mini-apps”, self-contained ready-for-deployment mini applications that can be used to guide development of future hardware. The advantage in developing these mini-apps, although time and effort consuming, is that it makes it much easier for designers of future hardware to study key aspects of scientific applications without having to deal with typically very large codebases. MaX is also actively engaged in profiling efforts (in collaboration with POP), and the implementation of novel OpenMP and MPI features into some of its key applications.

#### 4.3.8 “Exascale challenges: the view from NoMAD” – Claudia Draxl

The <https://nomad-coe.eu/> presentation focused on some of the big data challenges for materials research towards the exascale era, where the amount of data produced on workstations, compute clusters and supercomputers is growing exponentially but most are thrown away. An overview of materials data and their structure was shown (Fig.19), and an example was provided where the calculation for a single thermoelectric figure (at level IV) involves the generation of large amounts of intermediate data (at levels I thru III). In order to allow researchers to better exploit such data, the NoMAD repository is being established to host, organise and share materials data. It accepts input and output files from all major codes, and contains over 5 million entries at the time of this report. This has facilitated, for example, studies in DFT calculations of solids, where results from different electronic calculation software can be compared in order to assess data quality.

Apart from the repository, a NoMAD Encyclopedia has been developed that provides an easy way for researchers to explore properties of given materials that have been previously computed, complemented by visualisation tools. Going further, the NoMAD Analytics Toolkit allows researchers to study correlations and structure in the data repository,

Level	Properties	Methods	Size
I	Atomic positions and nuclear charges, properties of free atoms, symmetry, temperature, pressure	Input: definition of material <i>gene</i>	10 kB - 10 MB
II	Total energy, electron density, potential, wavefunctions, atomic forces, optimized geometry, elastic constants, etc.	Density-functional theory (DFT) and <i>ab initio</i> molecular dynamics (MD)	10 MB - 10 TB
III	Excitation energies, dielectric screening, matrix elements of Coulomb interaction, etc. optical spectra, electrical conductivity, phonon spectra, thermal conductivity, etc.	Many-body perturbation theory (MBPT), DF perturbation theory, <i>ab initio</i> MD	1 GB - 1 TB
IV	Efficiency of solar cell, thermoelectric figure of merit, turn-over frequency of catalyst, etc. as a function of temperature and pressure	Modeling, output derived from levels I-III <i>phenotype</i>	10 kB - 1 MB

Figure 19: Overview of materials data and their structure

enabling scientists and engineers to identify materials for potential use in novel products. In summary, the approach towards exascale is very much data-oriented, i.e. how to improve on ways to handle and better exploit the exponential increase in data quantity and complexity.

#### 4.3.9 “Exascale challenges: the view from EoCoE” – Matthieu Haefele

EoCoE is an energy oriented Centre of Excellence with four main thematic pillars: meteorology, materials, water and fusion all supported by a transversal group focused on advanced maths linear algebra and HPC tools and techniques. Some of their contributions towards exascale include design of numerical schemes, software package improvements and some efforts on porting applications to new architectures. They also provide support for performance evaluations for the different communities.

With regards to exascale software stack improvements, the CoE has in-house expertise across different areas such as numerical method improvements, linear algebra packages (e.g. parallel sparse direct solvers, hybrid solvers, including promotion of European linear algebra technologies), parallel I/O (e.g. FTI, XIOS, SIONlib, PDI). On application performance, EoCoE has defined 28 performance metrics (Fig.20), and automated and reproducible methods to collect them using JUBE.

	Metric name	03/01/2016
	Test-case	case1
Global	Total Time (s)	43.2
	Time IO (s)	0.3
	Time MPI (s)	12.4
	Memory vs Compute Bound	1.1
	Load Imbalance MPI	24.8
IO	IO Volume (MB)	35.8
	Calls (nb)	384000
	Throughput (MB/s)	105.0
	Individual IO Access (kB)	0.1
MPI	P2P Calls (nb)	0
	P2P Calls (s)	0.0
	P2P Message Size (kB)	0.0
	Collective Calls (nb)	2721
	Collective Calls (s)	0.1
	Collective Message Size (kB)	908.4
	Synchro / Wait MPI (s)	11.7
	Ratio Synchro / Wait MPI	94.8
Node	Time OpenMP (s)	0.0
	Ratio OpenMP	0.0
	Time Synchro / Wait OpenMP	0.0
	Ratio Synchro / Wait OpenMP	0.0
Mem	Memory Footprint (B)	66 mB
	Cache Usage Intensity	N.A.
Core	IPC	N.A.
	Runtime without vectorisation (s)	46.5
	Vectorisation efficiency	1.1
	Runtime without FMA (s)	44.6
	FMA efficiency	1.0

Figure 20: EoCoE Performance metrics table

In collaboration with POP, EoCoE holds performance evaluation workshops that brings both code developers and HPC experts together. During the workshop, both groups work together to carry out the performance analyses, generate the performance metrics and producing the performance report. This allows the code developers to gain knowledge on performance tools and the capability to re-run the performance evaluation. Through this type of approach, it has been shown that the resulting performance improvements have saved over 40 million core hours for one study alone

(2.5× speed-up), while another case where a 12× speed-up has paved the way for simulations of larger systems.

However, the author (not necessarily the view across EoCoE) has raised some concerns about the “mini-app” route to exascale that are being undertaken by some. The argument is that while the mini-app itself may be factored out of an application and adapted for exascale systems, re-integration of these mini-app back into the applications can often be problematic (e.g. the physics may need to change when executed on new architectures, breaking the original relationship to the code factored out into the mini-app). The proposed approach, then, is to concentrate resources on a small number of applications, centred on key scientific projects, and start already the port of relevant parts of these applications (holistically) on new architectures.

#### 4.4 Outcomes and key recommendations for E-CAM

The discussion sessions of the meetings were intense and probing. Repeatedly raised was the potential conflict of interest between the pilot projects (where industrial partners are interested in software developments that facilitate new science, including the development of new algorithms) and the funding agency desire to focus software developments exclusively on HPC goals. It was recognised that E-CAM needs to develop niche activities that can provide mutual benefit overlapping area between these two objectives.

To this end, E-CAM has highlighted the possibility to provide a reliable software delivery mechanism between academic software developers and industrial partners. In addition, it has sought to influence the workflows of the scientific developer community so that it aligns more with the expectations of software in the industrial space (in terms of documentation, usability and testing).

Increasing the E-CAM community exposure to HPC technologies is also a key goal that is to be enhanced by a set of HPC pilot projects (see Section 4.4.1) and the introduction of transversal components to our ESDW events (exposing people to new technologies and best practices).

Finally, to ensure that E-CAM can achieve it's own internal goals, a number of additional steps are to be taken:

- Deeper collaboration with projects such as EoCoE, POP, PRACE to collaborate on scaling and optimisation work for codes with potential for exascale systems,
- Leverage transversal ESDW workshops to create more synergy between the Work Package (WP)s and bring the E-CAM developers to the same level in terms of parallel code development skills,
- Increase the visibility and impact of the modules developed by E-CAM community by publishing pilot project webpages and advertising them via Twitter and other means.

##### 4.4.1 WP7 Pilot Projects

To engage at a practical level with the challenges we have exposed (both in this Section and Section 3), E-CAM has chosen to engage in two pilot projects that filter these challenges in two particular directions:

- An HTC project that recognises the role of ensemble calculations in the E-CAM community and provides an adaptive framework that efficiently maps it to extreme-scale resources. This project will be carried out in collaboration with PRACE and will be implemented as a Python module.
- A runtime system project that adapts an existing C++ mini-app (a "Minimal Espresso++" package (MESSPP)) to the HPX runtime system. The goal is to quantify the complexity of such a task, the effort required for it and the tuning required at development and runtime to make it efficient.

At the time of writing, these projects were still in the design phase and the resources to implement them were being sourced within the project.

## References

### Acronyms Used

**CECAM** Centre Européen de Calcul Atomique et Moléculaire

**HPC** High Performance Computing

**HTC** High Throughput Computing

**PRACE** Partnership for Advanced Computing in Europe

**ESDW** Extended Software Development Workshop

**WP** Work Package

**EsD** Extreme-scale Demonstrators

**MPI** Message Passing Interface

**NVM** Non-volatile Memory

**PGAS** Partitioned Global Address Space

**PATC** PRACE Advanced Training Centre

**CoE** Center of Excellence

**FPU**s Floating Point Units

**BSC** Barcelona Supercomputing Center

### URLs referenced

#### Page ii

<https://www.e-cam2020.eu> ... <https://www.e-cam2020.eu>  
<https://www.e-cam2020.eu/deliverables> ... <https://www.e-cam2020.eu/deliverables>  
 Internal Project Management Link ... <https://redmine.e-cam2020.eu/issues/44>  
[liang.liang@idris.fr](mailto:liang.liang@idris.fr) ... <mailto:liang.liang@idris.fr>  
<http://creativecommons.org/licenses/by/4.0> ... <http://creativecommons.org/licenses/by/4.0>

#### Page 2

EsD ... <http://www.etp4hpc.eu/en/esds.html>  
 ETP4HPC ... <http://www.etp4hpc.eu/>

#### Page 3

Eurolab-4-HPC Long-Term Vision on High-Performance Computing ... <https://www.eurolab4hpc.eu/static/deliverables/D2-2--final-vision.3718a25ff0dd.pdf>

#### Page 5

Intel Knights Landing ... <https://ark.intel.com/products/codename/48999/Knights-Landing>  
 DEEP ... [http://www.deep-project.eu/deep-project/EN/Home/home\\_node.html](http://www.deep-project.eu/deep-project/EN/Home/home_node.html)  
 SC14 ... <http://sc14.supercomputing.org/>  
 Some Surprises in the 2018 DoE Budget for Supercomputing ... <https://www.nextplatform.com/2017/05/23/surprises-2018-doe-budget-supercomputing/>  
 Knights Mill ... <https://www.servethehome.com/intel-knights-mill-for-machine-learning>  
 Intel Advisor ... <https://software.intel.com/en-us/intel-advisor-2017-user-guide-linux>  
 OpenMP ... <http://www.openmp.org/>  
 VTune ... <https://software.intel.com/en-us/intel-vtune-amplifier-xe>  
 Intel Advisor ... <https://software.intel.com/en-us/intel-advisor-xe-support/training>  
 Vtune ... <https://software.intel.com/en-us/intel-vtune-amplifier-xe-support/training>  
 Tesla V100 ... <https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/>

#### Page 6

NVIDIA Deep Learning SDK ... <https://developer.nvidia.com/deep-learning-software>  
 CUDA 9.0 ... <http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>  
 OpenMP 4.5 ... <http://www.openmp.org/updates/openmp-4-5-specs-released/>  
 OpenACC open standard ... <https://www.openacc.org/specification>

Unified Virtual Addressing... <https://devblogs.nvidia.com/parallelforall/beyond-gpu-memory-limits-unified-memory/>  
 Tensor Core ... <https://devblogs.nvidia.com/parallelforall/cuda-9-features-revealed>

**Page 7**

Unified Memory... <https://devblogs.nvidia.com/parallelforall/beyond-gpu-memory-limits-unified-memory/>  
 NVidia education site ... <https://developer.nvidia.com/cuda-education>  
 presentation on performance of the Clang OpenMP 4.5 implementation on NVIDIA gpus... <http://on-demand.gputechconf.com/gtc/2016/presentation/s6510-jeff-larkin-targeting-gpus-openmp.pdf>  
 FPGAs... <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>  
 ARM+FPGA Exascale Project... <https://www.hpcwire.com/2017/09/07/eu-funds-20-million-euro-armfpga-exascale/>  
 ExaNeSt... <http://www.exanest.eu/>  
 EcoScale... <http://www.ecoscale.eu/>  
 ExaNoDe... <http://www.exanode.eu/>  
 OpenCL for FPGA... <https://www.altera.com/support/training/course/oopncl100.html>  
 PRACE calls... <http://www.prace-ri.eu/call-announcements/>

**Page 8**

DECI calls... <http://www.prace-ri.eu/deci-13-call/>

**Page 9**

Strategic Research Agenda... <http://www.etp4hpc.eu/en/news/18-strategic-research-agenda-update.html>

**Page 10**

E-CAM Survey of Application Software... <https://docs.google.com/forms/d/e/1FAIpQLSc9i-vb8VNLm-KqC7Wy4i6d1m/viewform?c=0&w=1>  
 CECAM website... <https://www.cecarn.org>

**Page 12**

here... <https://web.archive.org/web/20170325025026/http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4660.pdf>  
 Standard Template Library... [https://en.wikipedia.org/wiki/Standard\\_Template\\_Library](https://en.wikipedia.org/wiki/Standard_Template_Library)  
 multi-threading programming model for C++... [https://en.wikipedia.org/wiki/C%2B%2B11#Multithreading\\_memory\\_model](https://en.wikipedia.org/wiki/C%2B%2B11#Multithreading_memory_model)  
 Fortran 2015... <http://fortranwiki.org/fortran/show/Fortran+2015>  
 coarray... [https://en.wikipedia.org/wiki/Coarray\\_Fortran](https://en.wikipedia.org/wiki/Coarray_Fortran)  
 Software Engineering and Parallel Programming in Modern Fortran... <https://www.cranfield.ac.uk/courses/short/aerospace/software-engineering-and-parallel-programming-in-modern-fortran>  
 compiler support for the latest Fortran standards... <http://fortranwiki.org/fortran/show/Compiler+Support+for+Modern+Fortran>

**Page 13**

Python 2 will stop being developed in 2020... <https://pythonclock.org/>  
 GASNet... <https://gasnet.lbl.gov/>  
 MPI-3.1... <http://mpi-forum.org/docs/mpl-3.1/mpl31-report.pdf>  
 scope of past and future updates to the MPI standard... [https://www.lrz.de/services/compute/courses/x\\_lecturenotes/Parallel\\_Programming\\_Languages\\_Workshop/MPI.pdf](https://www.lrz.de/services/compute/courses/x_lecturenotes/Parallel_Programming_Languages_Workshop/MPI.pdf)  
 level of OpenMP support among other compilers... <http://www.openmp.org/resources/openmp-compilers/>  
 OpenACC... <https://www.openacc.org/>  
 nvcc compiler... <http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html>  
 Vulkan... [https://en.wikipedia.org/wiki/Vulkan\\_\(API\)](https://en.wikipedia.org/wiki/Vulkan_(API))

**Page 14**

HPX... <https://github.com/STELLAR-GROUP/hpx>  
 ParalleX... <http://stellar.cct.lsu.edu/pubs/icpp09.pdf>  
 Kokkos... <https://github.com/kokkos/kokkos>  
 OmpSs... <https://pm.bsc.es/ompss>

**Page 15**

E-CAM Extreme-Scale State-of-the-Art Workshop... <https://www.cecarn.org/workshop-2-1512.html>  
 European HPC Technology Strategic Research Agenda (SRA)... <http://www.etp4hpc.eu/en/news/18-strategic-research-agenda.html>  
 EasyBuild... <http://easybuild.readthedocs.io/en/latest/>

**Page 19**



DBCSR library ... <https://dbcsr.cp2k.org/>

**Page 21**

POP CoE ... <https://pop-coe.eu/>

**Page 22**

MaX CoE ... <http://www.max-centre.eu/>

**Page 23**

EoCoE ... <http://www.eocoe.eu/>

**Page 24**

MESSPP ... <https://gitlab.e-cam2020.eu/vargas/MinimalMD-w-HPX-Kokkos-Charm>

## Citations

- [1] D. Borgis, L. Liang, L. Petit, M. Lysaght, and A. O’Cais, “Hardware Developments I - A Survey of State-of-the- art Hardware and Software,” Jul. 2016. [Online]. Available: <https://doi.org/10.5281/zenodo.929533>
- [2] R. Friedman, “C++ Parallel STL Introduced in Intel Parallel Studio XE 2018 Beta,” May 2017. [Online]. Available: <https://insidehpc.com/2017/05/parallel-stl/>