H2020-FETHPC-01-2016

DEEP-EST

DEEP - Extreme Scale Technologies

Grant Agreement Number: 754304

D7.5
Case Study Booklet

Final

Version: 1.0

Author(s): X.Guo (UEDIN)

Contributor(s): M.Bull (UEDIN), W.Klijn (JUELICH), L.Kusch (Aix-Marseille University), M. van der Vlag (JUELICH), K.Sontheimer (JUELICH), S.Durr (JUELICH), I.Kabadshow (JUELICH), L.Morgenstern (JUELICH), O.Brown (UEDIN), M.Bareford (UEDIN), S.Krieg (JUELICH), E.Gregory (JUELICH), C.Alexan drou (The Cyprus Institute), L.Breitwieser (CERN and ETH Zurich)

Date: 29.01.2021

Project and Deliverable Information Sheet

DEEP-EST Project	**Project Ref. №:** 754304	
	Project Title: DEEP - Extreme Scale Technologies	
	Project Web Site: http://www.deep-projects.eu	
	Deliverable ID: D7.5	
	Deliverable Nature: Report	
	Deliverable Level: PU	**Contractual Date of Delivery:** 31 / January / 2021
		Actual Date of Delivery: 29 / January / 2021
	EC Project Officer: Juan Pelegrín	

* - The dissemination levels are indicated as follows: PU = Public, fully open, e.g. web; CO = Confidential, restricted under conditions set out in Model Grant Agreement; CI = Classified, information as referred to in Commission Decision 2001/844/EC.

Document Control Sheet

Document	**Title:** Case Study Booklet	
	ID: D7.5	
	Version: 1.0	**Status:** Final
	Available at: http://www.deep-projects.eu	
	Software Tool: Microsoft Word	
	File(s): DEEP-EST_D7.5_Case_Study_Booklet_v1.0	
Authorship	**Written by:**	X.Guo (UEDIN)
	Contributors:	M.Bull (UEDIN), W.Klijn (JUELICH), L.Kusch (Aix-Marseille University), M.Vlag (JUELICH), K.Sontheimer (JUELICH), S.Durr (JUELICH), I.Kabadshow (JUELICH), L.Morgenstern (JUELICH), O.Brown (UEDIN), M.Bareford (UEDIN), S.Krieg (JUELICH), E.Gregory (JUELICH), C.Alexandrou (The Cyprus Institute), L.Breitwieser (CERN and ETH Zurich)
	Reviewed by:	M. Petrova-ElSayed (JUELICH)
	Approved by:	PMT

Document Status Sheet

Version	Date	Status	Comments
1.0	29/01/2021	Final version	EC Submission

Document Keywords

Keywords:	DEEP-EST, HPC, Exascale, Early Access Programme

Table of Contents

List of Figures

List of Tables

Executive Summary

The Early Access Programme (EAP) within the EU-funded DEEP-EST project aimed to provide academic and industrial users with the opportunities to access and use the DEEP-EST prototype platform. Experienced HPC and data analytic/machine learning users were invited to port/benchmark applications and evaluate the DEEP-EST hardware and software architectures deployed on the DEEP-EST prototype platform via two types of access: Type1 – Selected Application Access (aka. Type1 Access) and Type2 – Any Application Access (aka. Type2 Access). Through the calls to Type1 Access and to Type2 Access, a total of 9 EAP proposals were received, reviewed and awarded. The awarded EAP projects were carried out on the DEEP-EST prototype platform from March to October 2020 and each completed EAP project provided a lightweight final report on their results.

This deliverable provides a brief introduction to the DEEP-EST Early Access Programme and presents the case studies collated from all the completed EAP projects, based on their final reports.

1 Introduction

The Early Access Programme (EAP) invited academic and industrial users to gain access to, and use, the DEEP-EST prototype platform. It aimed to extend and complement the testing and evaluating the DEEP-EST hardware and software architectures performed within the project. The DEEP-EST prototype platform, with the innovative Modular Supercomputer Architecture, was created by coupling various compute modules according to the building-block principle. Each module is tailored to the needs of a specific group of applications and all modules together behave as a single machine (H.Cornelius, 2019). Through the EAP access, all three modules on the DEEP-EST prototype platform, including Cluster Module (CM), Data Analytics Module (DAM) and Extreme Scale Booster (ESB), were available for usage and testing by the EAP users. The software stack and module environment including a number of compilers, libraries and tools installed on the DEEP-EST prototype platform was also available to all the EAP users.

There were two types of EAP access available for the applicants to apply through two calls:

- **Type1 – Selected Application Access** (aka. Type1 Access), was designed for the applicants who were interested in using any applications which had already been ported to the DEEP-EST prototype platform within the DEEP-EST projects. The call for Type1 Access was opened on the 2nd of January 2020 and was closed on the 15th of September 2020.
- **Type2 – Any Application Access** (aka. Type2 Access), was designed for the applicants who would like to use their own codes (which were excluded from Type1 Access) on the DEEP-EST prototype platform, to evaluate the usability user-friendliness and explore different modular configurations for porting their codes to the novel hardware. The call for Type2 Access was opened on the 2nd of January 2020 and closed on the 28th of February 2020.

Nine EAP proposals from five different countries/organisations were received in total, as shown in Table 1. All the EAP proposals were reviewed and awarded, with the access starting from March 2020 and finishing on the 1st of November 2020. Support and two online training sessions (DEEP-EST EAP online Tutorial1, First Steps Tutorial, n.d.) (DEEP-EST EAP online Tutorial 2, Using Tools on DEEP-EST Prototype for Application Analysis and Tuning, n.d.) were provided to the EAP users to assist their usage and code tuning on the DEEP-EST prototype.

Eight EAP projects were completed/partially completed as planned (1 was closed due to the PI moving to a different organisation). This document collates the case studies based on all the received EAP final reports. Section 2 presents the case studies, thus sharing the EAP users' experiences and feedbacks on porting, testing, code tuning, as well as their project results achieved on the DEEP-EST prototype.

Type & ID	Titles	Institu-tions	Countries	Applications & Prototype Modules
Type1_1	Multiscale co-simulation and streaming analysis of experimental results for neuroscience	JUELICH, Aix-Marseille Université	Germany, France	NEST (CM)
Type1_2[1]	Machine learning and deep learning for magnetic reconnection and solar wind	KU Leuven	Belgium	GMM (DAM), DLMOS (DAM)
Type2_1	Optimization of Lattice Quantum Chromodynamics codes	JUELICH	Germany	Simlab_su3 (DAM), qcdcafe (KNLs on the DEEP-ER prototype)
Type2_2	A lightweight tasking framework for heterogeneous HPC hardware.	JUELICH	Germany	Eventify & FMSolvr (DAM)
Type2_3	EPiGRAM-HS WP2: Networking	UEDIN	UK	ECMWF Spectral Transform Dwarf, FPGA tutorial exercise (DAM)
Type2_4	Modular Supercomputing in Nuclear & Particle Theory	JUELICH	Germany	Various applications used by SimLab Nuclear & Particle Physics (CM, ESB, DAM)
Type2_5	Modular lattice QCD simulations with Lyncs	The Cyprus Institute	Cyprus	Lyncs (CM, ESB)
Type2_6	Distributed BioDynaMo	CERN and ETH Zurich	Switzerland	BioDynaMo (CM, DAM)
Type2_7	Modular Supercomputing for functional hetero-interfaces in photovoltaic applications	JUELICH	Germany	1D-NEGF (CM, ESB)

Table 1: The awarded EAP projects

[1] The project was awarded but was uncompleted and closed due to the PI moving to a different organisation.

2 EAP Project Case Studies

2.1 Multiscale co-simulation and streaming analysis of experimental results for neuroscience

2.1.1 Project introduction

- Project ID: Type1_1
- PI: Wouter Klijn (Simlab Neuroscience, JUELICH)
- Co-authors: Kim Sontheimer (SimLab Neuroscience, JUELICH), Michiel van der Vlag (SimLab Neuroscience, JUELICH), Lionel Kusch (Aix-Marseille Université)
- Project length: 7 months
- Scientific area: Neuroscience
- Application(s) and module(s) used: NEST on Cluster Modules (CM)

A co-simulation system was created within the Human Brain Project with the aim of integrating a number of simulators for the human brain simulation at different scales and having different characteristics regarding time, space and complexity. This EAP Type1 access project tested and benchmarked the co-simulation workloads on the hybrid hardware of the DEEP-EST prototype. Two simulators were mainly used in this project: NEST, which is one of the applications ported to the DEEP-EST prototype within the DEEP-EST project, and The Virtual Brain (TVB). The NEST simulator abstracts the neuron's morphology into a single point, capable of sending and receiving information in the form of spikes. The Virtual Brain pushes this abstraction even further and whole brain areas are reduced to single sets of differential equations, which model their mean activity. The project also aimed to utilise the large capacity of the network connectivity between different modules to support the co-simulation system’s communication requirements.

2.1.2 Technical work completed and results

During this EAP project, the following three project components have been installed and, where possible, benchmarked:

- The TVB-NEST co-simulation was installed and deployed on the system.
- RateML TVB - GPU was installed and the experiments on GPU VS multi-node were performed.
- The co-simulation pivot component was installed allowing M to N MPI transport of neuroscience data. Multi-node scaling experiments were performed comparing to the JURECA cluster.

2.1.2.1 RateML TVB – GPU

Performance comparisons of the RateML parameter sweep model generation for CUDA on a GPU and Python on a multi-node MPI implementation were carried out. Large parameter sweeps performed on the GPU were compared to a multi-node CPU implementation to establish an estimate of how many nodes it would take to equal the GPU as well as how much energy would be used. 4 TVB models were explored. The Pycuda library was used to drive the GPU.

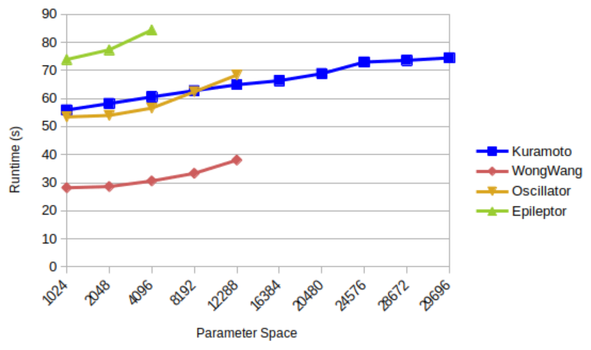

Figure 1: Runtime results for parameter scaling for TVB models on a single GPU

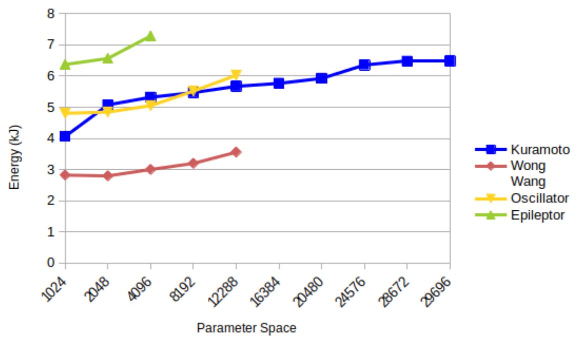

Figure 2: Energy results for parameter scaling for TVB models on a single GPU

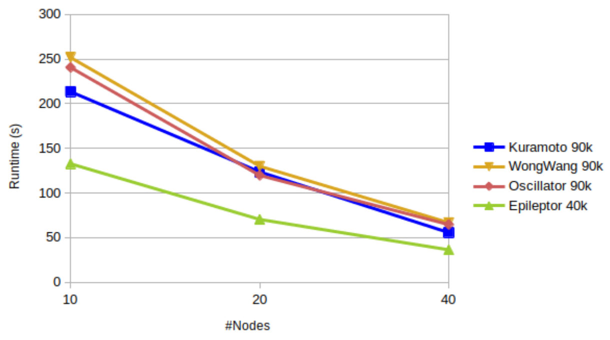

Figure 3: Runtime results for node scaling for TVB models on a multi CPU for fixed number of parameters

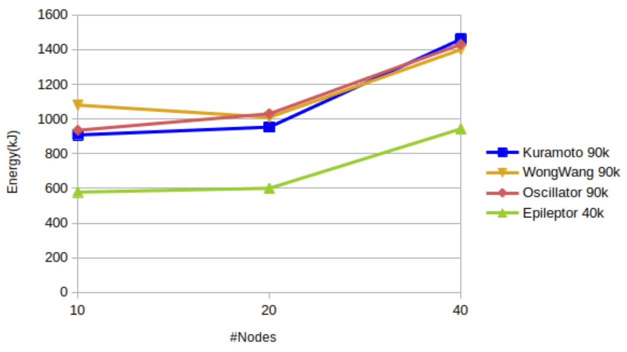

Figure 4: Energy results for node scaling for TVB models on a multi CPU for fixed number of parameters

Validated models	Runtime	Energy
• Kuramoto (K) • 2D oscillator (O) • Reduced WongWang (W) • Epileptor (E) • Simulated for 40s biological time on 68 brain nodes	• K does ~30k params max in ~ 70s on 1 GPU • W & O do 12k params max in ~40 & 70s on 1 GPU • E does 4k params in ~85s on 1 GPU • K, W, & O scale similarly increasing the number of nodes	• Measured by registering energy usuage on slurm nodes • 30k params (K) take ~6.5kJ in ~1 min. on 1 GPU • 12k params (W, O) take ~3.5kJ in ~1 min. on 1 GPU • 4k params (E) take ~7kJ in ~1 min. on 1 GPU • 90k params (K, W, O) and 40k params (E) take ~1.4MJ & ~1MJ in ~1 min. on 40 nodes with 12 CPUs

Table 2: Further info on RateML TVB scaling runtime and energy results

2.1.2.2 The cosimulation pivot

Scaling comparison was implemented on JURECA and the DEEP-EST prototype.

- Simulation of 1000 neurons for 100 seconds: 4761 packages, 10000 events per data package.
- Measurement of transfer rate: number of packages sent by simulation / number of packages receives by analysis (transfer rate of >1 no loss of packages).

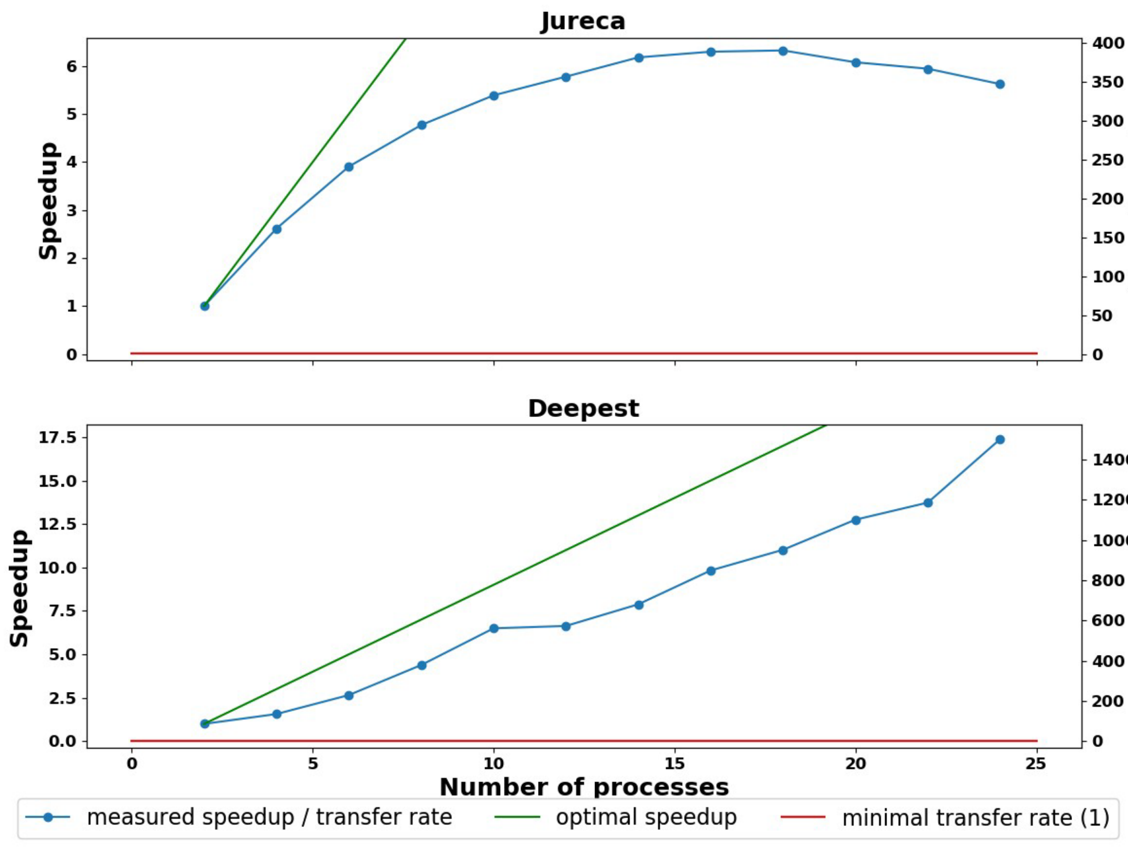

Figure 5: Scaling comparison on JURECA and the DEEP-EST prototype.

2.1.3 Benefit from EAP and DEEP-EST prototype

Energy is a vital dimension when it comes to benchmarking. A major benefit from the DEEP-EST prototype usage is that the energy registration was enabled and could be collected through slurm per process, which was not the case for JUWELS or JUSUF. Furthermore, the access to the hybrid system with modern network interface allowed the testing of data transport.

2.1.4 Future plan

Only single hardware system work has been performed within the EAP project period. The next step will be the completion of the development work on the co-simulation workflows followed by the detailed full system experiments and benchmarking on the hybrid resources. Continued access to the DEEP-EST Prototype platform has been applied and granted for the next step work.

2.2 Optimization of Lattice Quantum Chromodynamics codes

2.2.1 Project introduction

- Project ID: Type2_1
- PI: Stephan Durr (University of Wuppertal and JUELICH)
- Project length: 6 months
- Scientific area: Lattice Quantum Chromodynamics
- Application(s) and module(s) used: simlab_su3 on Data Analytics Module (DAM), qcdcafe (DEEP-ER test platform, KNL nodes)

This EAP Type2 access project ported simlab_su3, an application that simulates pure Yang-Mills theory with N_c=3 colours, to the DAM of the DEEP-EST prototype with only a small amount of effort. Minor changes were applied to its functionality to enable the porting, which was mainly adding COLLAPSE statements to use all available threads.

This project also ported and optimised qcdcafe, an application which simulates Quantum Chromodynamics with a variety of fermion discretisations (Wilson, Brillouin, staggered, Adams), on the KNL nodes of the DEEP-ER test platform. The details of the development work and the results achieved based on the KNL nodes are not included in this deliverable.

2.3 A lightweight tasking framework for heterogeneous HPC hardware

2.3.1 Project introduction

- Project ID: Type2_2
- PI: Ivo Kabadshow (JUELICH)
- Co-author: Laura Morgenstern (JUELICH)
- Project length: 6 Months
- Scientific area: Molecular Dynamics esp. GROMACS
- Application(s) and module(s) used: Eventify & FMSolvr on Data Analytics Module (DAM, NVIDA 100)

A tasking framework including Eventify and FMSolvr were being developed for Fast Multipole (FMM) Library. It aimed to solve the N-body problem in O(N) for a user-defined accuracy and provide a library capable of abstracting HW-features away. This EAP Type2 access project focused on using and experimenting with the CPUs+GPUs modular hardware and software on the DEEP-EST prototype platform for the one codebase development for both the architectures.

2.3.2 Technical work completed and results

The most vital parts of porting the CPU tasking framework to GPUs with CUDA had already been done before this EAP project, thus the access to the DEEP-EST prototype was used for the first performance optimisations on GPUs. The NVIDIA V100 GPUs of the DAM were used to conduct a comparative performance analysis of the following locking mechanisms:

- FMSolvr Mutex: spin-lock via atomicCAS(), atomicExch() and memory fencing
- Locking mechanisms provided by lib freestanding based on libcu++: mutex, sem_mutex, ticket_mutex

Porting the code from CPUs to GPUs took large amounts of effort and has not yet finished by the end of this EAP project. Major obstacles for the portability were the lack of a C++ STL for GPUs as well as the data management between CPU and GPU in object-oriented contexts. The performance of the latter also played a key role when porting to modular architecture since the application was already latency and synchronisation-critical on a single multicore-node.

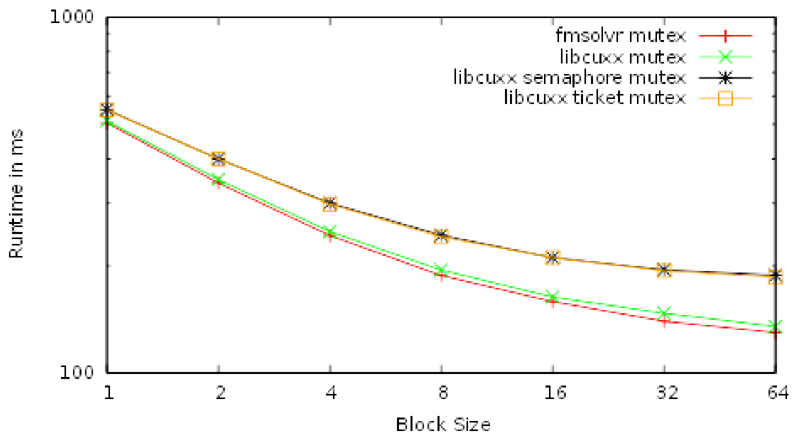

Figure 6: Eventify-based Fast Monopole Method, depth d=6, grid size g=320, on Nvidia V100 on DEEP-EST prototype

2.3.3 Benefit from EAP and DEEP-EST prototype

The EAP access enabled the project to examine the consequences of the independent thread scheduling for the application code via providing the access to Nvidia’s V100 GPU. Nvidia’s V100 GPU is the first GPU to support independent thread scheduling. It is a vital feature when aiming at performance portability on CPU-GPU systems for synchronisation-based applications.

2.3.4 Future plan

As useful, but unexpected, results were obtained via this EAP project, a deeper performance analysis will be carried out.

2.4 EPiGRAM-HS WP2: Networking

2.4.1 Project introduction

- Project ID: Type2_3
- PI: Oliver Thomson Brown (on behalf of EPiGRAM-HS (EPiGRAM-HS, n.d.) WP2) (EPCC, UEDIN)
- Co-author: Michael Bareford (EPCC, UEDIN)
- Project length: 8 Months
- Scientific area: Weather and Climate Modelling
- Application(s) and module(s): ECMWF Spectral Transform Dwarf and FPGA tutorial examples on Data Analytics Module (DAM, FPGA)

ECMWF Spectral Transform Dwarf is a proxy application (mini-app) that implements the spectral transform method – one of the building blocks of many global weather and climate models, such as the ECMWF Integrated Forecast System (IFS). The original plan for this EAP Type2 access project was to investigate the implementation difficulty and performance of MPI collectives on the FPGA of the DEEP-EST DAM, since MPI is critical to the performance of this proxy application. However, due to the technical issues met, the project was changed to using the FPGA tutorial examples to experiment with and evaluate the FPGA usage on the DEEP-EST prototype.

2.4.2 Technical work completed and results

The main evaluation was completed via working through the FPGA tutorial exercises (Intel OpenCL Vector Addition Design Example, , n.d.) on the FPGAs of the DEEP-EST DAM. It was fairly straightforward to build some of the kernels associated with the FPGA tutorial exercises in the emulation mode. However, attempting to build these kernels for the actual FPGA device resulted in the compiling error output “Compiler Error, not able to generate hardware”.

With the LD_LIBRARY_PATH extended (e.g. liblto_plugin), compiling the host and (emulated) kernel on the login node was successful and some basic tutorial examples such as helloworld and vectoradd could be run. However, when building for the device from a compute node, the kernel compiler (aoc) appeared to hang and gave no output. After the investigation by the DEEP-EST project, it was confirmed in late July 2020 that the aoc compiler was not really hanging when trying to compile a kernel on the compute node, but was actually taking a very long time to output an error. The issues with the file system (BeeGFS) were preventing the aoc compiler from locating its own Quartus installation.

The issues with the file system were noticed to be fixed in late Sep/early Oct 2020, and a helloworld kernel was eventually compiled explicitly for the FPGA device. However, running the device-built kernel failed with "ERROR: CL_DEVICE_NOT_FOUND". It was noticed that compiling was failing due to changes (i.e., deletions) having been made to the underlying software stack. The shell commands in "/usr/local/fpga/FPGA_init.sh" were no longer compatible with the existing file hierarchy. Due to the time limit for the EAP projects, there had been no confirmation for this issue raised by the time of the project ending.

From the EAP project experience on FPGA, it was also noted that the supporting software available on the DEEP-EST prototype may affect the building and usage on the FPGAs. The main software components available appeared to be "Intel Quartus Prime Design Suite 18.1.2.277" and "Intel FPGA SDK for OpenCL 18.1.2", but there were also newer versions of the FPGA SDK, 19.1, 19.2 and 20.0, available on the prototype. The versions used for this project were Quartus 19.2 and SDK 19.1, i.e. the tools/libraries used to build the kernels and hosts as mentioned above were provided by the Intel Quartus Software Suite v19.2.0 b57, featuring Intel FPGA SDK for OpenCL (v1.0) v19.1. The FPGA device was indicated by the "-board=pac_s10_dc" compile option.

2.4.3 Benefit from EAP and DEEP-EST prototype

FPGA access and experience on the DEEP-EST prototype.

2.4.4 Future plan

It will be good to extend the work on the DEEP-EST system if developing FPGA kernels is demonstrated to be viable on the DEEP-EST DAM.

2.5 Modular Supercomputing in Nuclear & Particle Theory

2.5.1 Project introduction

- Project ID: Type2_4
- PI: Stefan Krieg (JUELICH)
- Co-author: Eric Gregory (JUELICH)
- Project length: 8 Months
- Scientific area: Nuclear & Particle Physics/High Energy Physics/HPC
- Application(s) and module(s) used: Various applications used by the SimLab Nuclear & Particle Physics or associated users, such as chroma (Edwards & Joo, 2005) or grid (Boyle, Yamaguchi, Cossu, & Portelli, 2015), on Cluster Module (CM), Data Analytics Module (DAM) and Extreme Scale Booster (ESB)

This EAP Type2 access project aimed to develop, port and test some applications which implement the Hybrid Monte Carlo method for use in nuclear/particle physics, on a working modular system.

2.5.2 Technical work completed and results

2.5.2.1 QMod

Much of the work in this EAP project focused on the development of a library geared towards the Modular Supercomputing Architecture, called QMod (Gregory, 2019).

This library allows a single lattice QCD simulation to run simultaneously across separate hardware groups, which may contain heterogeneous architectures. LQCD calculations lend themselves well to data parallelism – the data and kernel operations are extremely uniform. However, in a given workflow it may be possible for different functional elements of a simulation to be executed in parallel if they can be assigned to separate hardware groups. For example, in a calculation requiring many propagators, contractions of completed propagators can commence while the solver continues to produce propagators. Alternatively, I/O operations could often be executed in parallel with other workflow elements.

As well as unlocking additional concurrency, this type of functional parallelism allows workflow elements to be mapped to hardware most suitable to the task with different numbers of nodes for each element. In the case of a modular supercomputer, workflow elements may also be assigned to the most suitable architecture, for example, accelerator nodes, large memory nodes, or I/O nodes.

The QMod library acts in conjunction with the USQCD software stack (USQCD, 1999), which is a set of libraries handling generic LQCD operations in a uniform way and includes architecture-specific solvers, such as QPhiX for Intel multi-core architectures or QUDA for Nvidia GPUs. Many open-source community LQCD codes are compiled with this stack, allowing portability across a variety of architectures. Figure 7 illustrates where the two components of the QMod library must be compiled into the USQCD software stack.

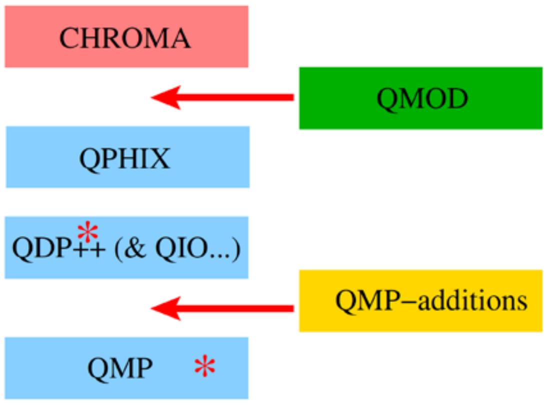

Figure 7: Two components of QMod are compiled into the USQCD software stack. Small changes are required of the USQCD libraries marked with "*".

Upon initiation, QMod splits the default MPI communicator into the different hardware groups, retaining the original as an "inter-communicator" to pass data objects between them. It includes a set of functions that mimic I/O operations for lattice data fields such as propagators, eigenvectors, or gauge fields. Instead of writing a lattice field to disk, a QMod function sends the field to a separate hardware group, and analogous to the reading a field from disk, a QMod operation enables receiving a field from a remote partition. There is some complexity here as the sending and receiving groups may contain differing numbers of nodes and have different lattice layout schemes.

In addition to the advantages of increased concurrency, QMod can reduce or eliminate I/O by keeping data in flight between different workflow elements. An important example is the calculation of light nuclei correlators. In this case the propagators corresponding to the valence quarks must be contracted in a large number of combinations. The number of contractions can run in the hundreds or thousands for even small nuclei.

Standard methods involve saving the propagators to disk and subsequently re-loading them many times to complete the required contraction combinations. Alternately, one could assign a group of accelerators hardware nodes to calculate the propagators and send them through the network to groups of nodes to complete the contractions without saving propagators to disk.

Initial tests of QMod have proven the feasibility of launching a single compute job which solves for propagators on the booster and completes contractions on the cluster nodes. Further development will improve communication efficiency and test load balancing schemes for specific problems.

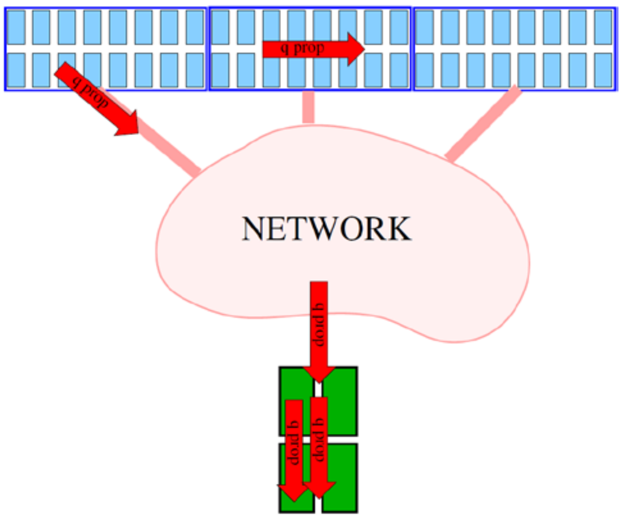

Figure 8: Accelerated booster nodes generating propagators and sending them to cluster nodes to contract into hadron correlators

2.5.2.2 Carbon Nano-Structures

Carbon Nano-Structures (CNS) provide an arena in which rich and complex phenomena occur. Because of their amazing mechanical and electrical properties, they are not only interesting to the condensed matter and atomic physics communities but play a major role in industrial research and development. Because of the small dimensions intrinsic to CNSs, strong correlations are prevalent and non-perturbative methods are essential for quantifying their electronic properties. Many of the technological advancements made in lattice QCD can be adapted to CNS simulations (Krieg, Luu, Ostmeyer, Papaphilippou, & Urbach, 2019). The necessary tests for the development of these methods were carried out on the DEEP-EST hardware. These methods were used in a first fully controlled calculation of the semimetal-Mott insulator quantum phase transition (Ostmeyer, et al., 2020), where parts of the calculation again were run on the system (see acknowledgements for both papers).

2.5.3 Benefit from EAP and DEEP-EST prototype

New software was developed for the Modular Supercomputer Architecture (Gregory, 2019; Krieg, Luu, Ostmeyer, Papaphilippou, & Urbach, 2019) and run for the first simulations (Ostmeyer, et al., 2020).

2.6 Lyncs, a Python APU for Lattice QCD

2.6.1 Project introduction

- Project ID: Type2_5
- PI: Constantia Alexandrou (The Cyprus Institute)
- Project length: 7 Months
- Scientific area: Lattice QCD
- Application(s) and module(s) used: Lyncs on Cluster Module (CM) and Extreme Scale Booster (ESB)

This EAP Type2 access project tested and further developed the Lyncs API based on the DEEP-EST prototype. Lyncs (Lyncs-API, n.d.) is a Python API for Lattice QCD currently under development and it aims to bring several widely used libraries for Lattice QCD under a common framework. Lyncs flexibly links to libraries for CPUs and GPUs in a way that can accommodate additional computing architectures as these arise, achieving the best performance for the calculations while maintaining the same high-level workflow.

Lyncs distributes calculations using Dask (htt), with bindings to the libraries performed automatically via cppyy (htt1). Multiple distributed tasks can be executed in parallel and different computing units can be used at the same time to fully exploit the machine allocation. While Lyncs is designed to allow linking to multiple libraries, a set of targeted packages are focused including c-lime (htt2), DDalphaAMG (htt3), tmLQCD (htt4) and quda (htt5).

The DEEP-EST prototype was considered to be a very suitable platform for this EAP project since one of the main purposes of the Lyncs API is to enable multiple distributed calculations simultaneously, which fully aligns with the capabilities of the modular architecture of the DEEP-EST prototype platform.

2.6.2 Technical work completed and results

The standard computational approach in Lattice QCD (LQCD) is domain decomposition. With such an approach, the lattice is divided in blocks and each block is assigned to a process. In practice, only data parallelism is exploited. With Lyncs, it was aimed to bridge the gap between data and task parallelism, as well as enabling modular computing. The plan was to do this by bringing together several LQCD libraries optimized for various architectures under a common framework. For that a high-level Python interface was used for managing various computing partitions and distributing the workload between them.

The task of parallelism and modular computing was enabled using Dask. Currently the application supports a few CPU and GPU libraries. The EAP project has been able to successfully run tasks in parallel using both the CPU and GPU partitions on the DEEP-EST prototype. A lot of work is still to be done for supporting such features seamlessly in a fully-fledged application.

The application performance was also analysed. Since the external HPC libraries employed in the application have already been optimised for various architectures, in the following analysis these libraries are extended. The main performance issues to be investigated included: (i) the overhead introduced by calling functions of these libraries from Python, and (ii) the overhead introduced by Dask when managing task parallelism. Regarding the first issue, the Lyncs project performed automatic binding to C/C++ libraries using cppyy (htt1). This module has been developed in an HPC-wise manner and minimal overheads were introduced, especially after a function has been called once the following overheads were in the order of microseconds (in the first call, some compilation might be involved). On the other hand, the overheads introduced by Dask (htt), were more significant. Dask uses a server approach where various workers are managed by a client. Therefore, the computational tasks are submitted from the client to the server and then a scheduler is responsible to make the workers execute these tasks. According to the achieved experience, the Dask workflow had an overhead oin the order of milliseconds between the time a task was submitted and the time when it was executed. This was a significant overhead for very small tasks. This overhead was overcome by joining many small tasks within a bigger one and executing tasks that take order of seconds.

2.6.3 Benefit from EAP and DEEP-EST prototype

The work required to be able to benefit from the modularity and heterogeneous configuration of DEEP-EST turned out to be larger than what was originally estimated. However, the software design decision was influenced by the modular design of the machine and it is a major feature that the users of this EAP project want to promote with the Lyncs API.

2.6.4 Future plan

The code was not particularly optimised for the DEEP-EST prototype during the EAP project as the development hadn't reached an advanced stage of the application. A feature to be supported in the Lyncs API, specific for systems with a configuration similar to the DEEP-EST prototype, is a seamless utilisation of the in-node memory.

Following this EAP project, in order to reach production readiness of the software, the users plan to further strengthen the pillars of the Lyncs API, advertise it to their community and open it to external contributions and deployment.

2.7 BioDynaMo

2.7.1 Project introduction

- Project ID: Type2_6
- PI: Lukas Breitwieser (CERN and ETH Zurich)
- Project length: 7 Months
- Scientific area: Large-scale agent-based biological simulations
- Application(s) and module(s) used: BioDynaMo (www) on Cluster Module (CM) and Data Analytic Module (DAM)

BioDynaMo is a novel general platform for agent-based computer simulations of biological tissue dynamics. It allows to simulate multi-scale biological systems in a 3D physical system, accounting for mechanical interactions, diffusion of substances, and biological behaviours. This EAP Type2 access project ported and benchmarked BioDynaMo on the DEEP-EST prototype. The details of the project work and results are confidential at the current stage and therefore not included in this deliverable.

2.8 Modular Supercomputing for functional hetero-interfaces in photovoltaic applications

2.8.1 Project introduction

- Project ID: Type2_7
- PI: Sebastian Achilles (JUELICH)
- Project length: 7 Months
- Scientific area: Non-Equilibrium Green's Functions for transport phenomena
- Application(s) and module(s) used: 1D-NEGF on Cluster Module (CM) and Extreme Scale Booster (ESB)

The 1D-NEGF software is developed within the Non-Equilibrium Green's Functions (NEGF) framework, which is an advanced approach that allows for treatment of out-of-equilibrium transport phenomena. The NEGF framework provides a quantum-statistical mechanics picture of many-body systems far from equilibrium. The model, describing such physical system, includes open boundary conditions: a device is coupled through a boundary self-energy to an out-of-equilibrium environment where electrons move due to externally applied voltages. Due to two nested self-consistent loops and non-local dependencies of the Green's functions on the spectral energy and momentum variables, NEGF based simulations are computationally intensive.

This EAP Type2 access project ported and tested the application 1D-NEGF on the DEEP-EST prototype. The details of the project work and results are confidential and therefore not included in this deliverable.

3 User Feedback

Beyond the technical reporting, the EAP team also invited each project to provide feed-backs/comments where possible, based on their experience of using the DEEP-EST prototype via the EAP access. Users were also invited to rate their user experience from 5 - Very Good to 1 - Bad in the following 5 aspects.

It is gratifying to see that most of the EAP users rated their experience of the EAP access and using the DEEP-EST prototype as 5 - Very Good or 4 – Good: Documentation (75%), Tutorials (67%), Support (83%), System usability (88%), General experience (71%). The following Table 3 shows the percentages of each score received in each user experience rating category. Several users sent comments and considered the support were very helpful. Users from 5 EAP projects would recommend the DEEP-EST Access if there is any future programme available, while the other completed 3 EAP projects did not explicitly answer this feedback question in the final report.

There were also a number of valuable comments received for the future improvement and considerations. The comments are mainly relevant to the available modules on the prototype, documentation improvement, and FPGA porting/usage challenge for HPC applications.

	5-Very Good	**4-Good**	**3-Average**	**2-Fair**	**1-Bad**
Documentation	57%	29%	14%		
Tutorials	67%		33%		
Support	83%		17%		
System usability	50%	38%		13%	
General experience	43%	29%		14%	14%

Table 3: Percentages of each score in the user experience rating categories.

The following are the quotes from part of the received user comments:

- *“Given its experimental status, the system usability is very good, it is just now available on a 24-7 basis like a production machine. Any non-standard behaviour is swiftly addressed by JSC staff. The DEEP-EST access was enormously useful, and I recommend it to anyone interested in achieving good performance on modular mid-sized machines.”*

- *“Yes, we would absolutely recommend the system!”*

- *“A general comment that we would like to make is that modular computing is a very interesting approach but also very challenging for current HPC applications. The first step that needs to be done by our community for exploiting it, is to develop applications that support task parallelism, as well as libraries that have the flexibility to run multiple instances of themselves within the same processor without creating conflicts. Also FPGAs are a very challenging technology as they require dedicated software and competences. It is difficult, though, to estimate if it is worth the effort of developing software for them and also if they will be supported in a future by HPC systems as it is happening nowadays with GPUs for instance.”*

- *"When our application will support the modularity offered by DEEP-EST we will advertise it as one of the supported systems."*

- *"Support …was well organised. There were some problems with the modules but the support team solves it quite fast. It would be better to have a link to the modules (https://apps.fz-juelich.de/jsc/llview/deep_modules/) on their website (https://www.deep-projects.eu/)."*

Reference(n.d.). Retrieved from https://dask.org

(n.d.). Retrieved from https://cppyy.readthedocs.io/en/latest/

(n.d.). Retrieved from https://github.com/usqcd-software/c-lime

(n.d.). Retrieved from https://github.com/sbacchio/DDalphaAMG

(n.d.). Retrieved from https://github.com/etmc/tmLQCD

(n.d.). Retrieved from https://github.com/lattice/quda

(n.d.). Retrieved from www.biodynamo.org

Boyle, P., Yamaguchi, A., Cossu, G., & Portelli, A. (2015, 12). Grid: A next generation data parallel C++ QCD library.

DEEP-EST EAP online Tutorial 2, Using Tools on DEEP-EST Prototype for Application Analysis and Tuning. (n.d.). Retrieved from https://deeptrac.zam.kfa-juelich.de:8443/trac/wiki/Public/User_Guide/Tutorial2

DEEP-EST EAP online Tutorial1, First Steps Tutorial. (n.d.). Retrieved from https://deeptrac.zam.kfa-juelich.de:8443/trac/wiki/Public/User_Guide/Tutorial1

Edwards, R. G., & Joo, B. (2005). The Chroma software system for lattice QCD. (G. T. Bodwin, D. K. Sinclair, E. Eichten, D. Holmgren, A. S. Kronfeld, P. Mackenzie, . . . A. X. El-Khadra, Eds.) *Nucl. Phys. B Proc. Suppl., 140*, 832. doi:10.1016/j.nuclphysbps.2004.11.254

EPiGRAM-HS. (n.d.). Retrieved from https://epigram-hs.eu/

Gregory, E. (2019). QCD on the Modular Supercomputer. *PoS, LATTICE2019*, 205. doi:10.22323/1.363.0205

H.Cornelius, H.-C. H. (2019). *Deliverable D3.2 -- Update: High-Level System Design.* Juelich: DEEP-EST Project Consortium.

Intel OpenCL Vector Addition Design Example, . (n.d.). Retrieved from https://www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/design-software/opencl/vector-addition.html?wapkw=vector%20addition

Krieg, S., Luu, T., Ostmeyer, J., Papaphilippou, P., & Urbach, C. (2019). Accelerating Hybrid Monte Carlo simulations of the Hubbard model on the hexagonal lattice. *Comput. Phys. Commun., 236*, 15-25. doi:10.1016/j.cpc.2018.10.008

Lyncs-API. (n.d.). Retrieved from https://github.com/Lyncs-API

Ostmeyer, J., Berkowitz, E., Krieg, S., Lähde, T. A., Luu, T., & Urbach, C. (2020). The Semimetal-Mott Insulator Quantum Phase Transition of the Hubbard Model on the Honeycomb Lattice. *Phys. Rev. B, 102*, 245105. doi:10.1103/PhysRevB.102.245105

USQCD. (1999). *USQCD Software stack*. Retrieved from https://www.usqcd.org/usqcd-software/

List of Acronyms and Abbreviations

A

API:	Application Programming Interface
APU:	Accelerated processing unit
AVX:	Advanced Vector Extensions
AVX-512:	Intel 512-bit SIMD instructions

B

BeeGFS:	The Fraunhofer Parallel Cluster File System (previously acronym FhGFS). A high-performance parallel file system.

C

CERN:	European Organisation for Nuclear Research / Organisation Européenne pour la Recherche Nucléaire, International organisation
CM:	Cluster Module: with its Cluster Nodes (CN) containing high-end general-purpose processors and a relatively large amount of memory per core
CNS:	Carbon Nano-Structures
CPU:	Central Processing Unit

D

D:	Deliverable, followed by a number, term to designate a deliverable (document) in the DEEP-EST project
DAM:	Data Analytics Module: with nodes (DN) based on general-purpose processors, a huge amount of (non-volatile) memory per core, and support for the specific requirements of data-intensive applications
DEEP:	Dynamical Exascale Entry Platform (project FP7-ICT-287530)
DEEP-ER:	DEEP - Extended Reach (project FP7-ICT-610476)
DEEP-EST:	DEEP - Extreme Scale Technologies
DoW:	Description of Work

E

EAP:	Early Access Programme to provide academic and industrial users with the opportunity to access and use the DEEP-EST prototype platform
EC:	European Commission
EC-GA:	Grant Agreement
ECMWF:	European Centre for Medium-Range Weather Forecasts
EoCoE:	Energy Oriented Centre of Excellence

EPCC:	A supercomputing centre based at the University of Edinburgh, UK, founded in 1990
EPiGRAM-HS:	A European Commission Funded project (Horizon 2020, Grant agreement ID: 801039) with the goal of designing and delivering a programming environment for Exascale heterogeneous systems in order to support the execution of large-scale applications
ESB:	Extreme Scale Booster: with highly energy-efficient many-core processors as Booster Nodes (BN), but a reduced amount of memory per core at high bandwidth
ETH Zurich:	Swiss Federal Institute of Technology in Zurich (German: Eidgenössische Technische Hochschule Zürich), a public research university in the city of Zürich, Switzerland
EU:	European Union
Exascale:	Computer systems or Applications, which are able to run with a performance above 10^{18} Floating point operations per second

F

Flop/s:	Floating point operation per second
FPGA:	Field-Programmable Gate Array, Integrated circuit to be configured by the customer or designer after manufacturing

G

GPU:	Graphics Processing Unit
GROMACS:	A toolbox for molecular dynamics calculations providing a rich set of calculation types, preparation and analysis tools

H

H2020:	Horizon 2020
HPC:	High Performance Computing

I

Intel:	Intel Germany GmbH, Feldkirchen, Germany
I/O:	Input/Output. May describe the respective logical function of a computer system or a certain physical instantiation

J

JSC:	Jülich Supercomputer Center, Forschungszentrum Jülich
JUELICH:	Forschungszentrum Jülich GmbH, Jülich, Germany
JURECA:	Jülich Research on Exascale Cluster Architectures
JUSUF:	Jülich Support for Fenix
JUWELS:	Jülich Wizard for European Leadership Science

K

KNL:	Knights Landing, second generation of Intel® Xeon Phi^{TM}
KU Leuven:	Katholieke Universiteit Leuven, Belgium

L

LQCD:	Lattice quantum chromodynamics

M

MPI:	Message Passing Interface, API specification typically used in parallel programs that allows processes to communicate with one another by sending and receiving messages
MPICH:	MPI implementation maintained by Argonne National Laboratory

N

NEST:	Widely-used, publically available simulation software for spiking neural network models developed by NMBU.
NUMA:	Non-Uniform Memory Access

O

OpenCL:	Open Computing Language, framework for writing programs that execute across heterogeneous platforms
OpenMP:	Open Multi-Processing, Application programming interface that support multiplatform shared memory multiprocessing
Open MPI:	MPI implementation maintained by the Open MPI Project

P

PI:	Principal Investigator

Q

QCD:	Quantum chromodynamics

S

SLURM:	Job scheduler that will be used and extended in the DEEP-EST prototype

U

UEDIN:	University of Edinburgh, UK

W

WP:	Work package