

**E-Infrastructures**

**H2020-EINFRA-2014-2015**

**EINFRA-4-2014: Pan-European High Performance Computing**

**Infrastructure and Services**

**PRACE-4IP**

**PRACE Fourth Implementation Phase Project**

**Grant Agreement Number:** **EINFRA-653838**

**D7.5**

**Application performance on accelerators**

***Draft***

Version: 0.1

Author(s): Victor Cameo Ponz, CINES

Date: 01.03.2016

Project and Deliverable Information Sheet

|  |  |  |
| --- | --- | --- |
| **PRACE Project** | **Project Ref. №: EINFRA-653838** | |
| **Project Title: PRACE Fourth Implementation Phase Project** | |
| **Project Web Site:** <http://www.prace-project.eu> | |
| **Deliverable ID:** < **D7.5**> | |
| **Deliverable Nature:** <DOC\_TYPE: Report / Other> | |
| **Dissemination Level:**  PU | **Contractual Date of Delivery:**  31 / 03 / 2017 |
| **Actual Date of Delivery:**  DD / Month / YYYY |
| **EC Project Officer: Leonardo Flores Añover** | |

\* - The dissemination level are indicated as follows: **PU** – Public, **CO** – Confidential, only for members of the consortium (including the Commission Services) **CL** – Classified, as referred to in Commission Decision 2991/844/EC.

Document Control Sheet

|  |  |  |
| --- | --- | --- |
| **Document** | **Title: Application performance on accelerators** | |
| **ID:** **D7.5** | |
| **Version:** <0.1> | **Status:** |
| **Available at:** <http://www.prace-project.eu> | |
| **Software Tool:** Microsoft Word 2010 | |
| **File(s):** d7.5\_4IP.docx | |
| **Authorship** | **Written by:** | Victor Cameo Ponz, CINES |
| **Contributors:** | Andrew Sunderland, STFC  Arno Proeme, EPCC  Charles Moulinec, STFC  Martti Louhivuori, CSC  Alan Grey, EPCC  Andrew emerson, CINECA  Dimitris Dellis, GRNET  Valeriu Codreanu, SURFSARA  Jacob Finkenrath, CyI  Janko Strassburg, BSC  Adem Tekin, ITU |
| **Reviewed by:** | Filip Stanek, IT4I  Thomas Eickermann, FZJ |
| **Approved by:** | MB/TB |

Document Status Sheet

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Status** | **Comments** |
| 0.1 | 13/03/2017 | Draft |  |
| 0.2 | DD/Month/YYYY |  |  |
| 1.0 | DD/Month/YYYY | Final version |  |

Document Keywords

|  |  |
| --- | --- |
| **Keywords:** | PRACE, HPC, Research Infrastructure, Accelerators, GPU, Xeon Phi, Benchmark suite |

**Disclaimer**

This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement n° . 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. Please note that even though all participants to the Project are members of PRACE AISBL, this deliverable has not been approved by the Council of PRACE AISBL and therefore does not emanate from it nor should it be considered to reflect PRACE AISBL’s individual opinion.

|  |
| --- |
| **Copyright notices**  © 2016 PRACE Consortium Partners. All rights reserved. This document is a project document of the PRACE project. All contents are reserved by default and may not be disclosed to third parties without the written consent of the PRACE partners, except as mandated by the European Commission contract for reviewing and dissemination purposes.  All trademarks and other rights on third party products mentioned in this document are acknowledged as own by the respective holders. |

Table of Contents

Project and Deliverable Information Sheet i

Document Control Sheet i

Document Status Sheet i

Document Keywords iii

Table of Contents iv

List of Figures v

List of Tables v

References and Applicable Documents vi

List of Acronyms and Abbreviations vi

List of Project Partner Acronyms viii

Executive Summary 1

1 Introduction 1

2 Targeted architectures 1

2.1 Co-processor description 1

2.2 Systems description 2

2.2.1 K40 cluster 2

2.2.2 Xeon Phi 5110P cluster 3

2.2.3 P100 cluster 3

2.2.4 Xeon Phi 7250 cluster 3

3 Benchmark suite description 3

3.1 Alya 3

3.1.1 Code desctiption 4

3.1.2 Test cases desctiption 4

3.2 Code Saturne 4

3.2.1 Code desctiption 5

3.2.2 Test cases desctiption 5

3.3 CP2K 5

3.3.1 Code desctiption 6

3.3.2 Test cases desctiption 6

3.4 GPAW 6

3.4.1 Code desctiption 6

3.4.2 Test cases desctiption 7

3.5 GROMACS 7

3.5.1 Code desctiption 7

3.5.2 Test cases desctiption 8

3.6 NAMD 8

3.6.1 Code desctiption 8

3.6.2 Test cases desctiption 8

3.7 PFARM 9

3.7.1 Code desctiption 9

3.7.2 Test cases desctiption 9

3.8 QCD 10

3.8.1 Code desctiption 10

3.8.2 Test cases desctiption 11

3.9 Quantum Espresso 11

3.9.1 Code desctiption 11

3.9.2 Test cases desctiption 12

3.10 Synthetic benchmarks -- SHOC 12

3.10.1 Code desctiption 12

3.10.2 Test cases description 13

3.11 SPECFEM3D 13

3.11.1 Test cases definition 13

4 Applications performances 14

4.1 ALYA 15

4.2 Code\_Saturne 16

4.3 CP2K 16

4.4 GPAW 17

4.5 GROMACS 18

4.6 NAMD 19

4.7 PFARM 20

4.8 QCD 21

4.8.1 First implementation 21

4.8.2 Second implementation 22

4.9 Quantum Espresso 29

4.10 Synthetic benchmarks (SHOC) 30

4.11 SPECFEM3D 30

5 Conclusion and future work 30

List of Figures

[Figure 1 ALYA benchmark results 15](#_Toc477154938)

[Figure 2 Code\_Saturne's performance on KNLs. AMG is used as a solver in V4.2.2. 16](#_Toc477154939)

[Figure 3 Parallel scaling of GPAW 17](#_Toc477154940)

[Figure 4 Parallel Scaling for GROMACS GPU 18](#_Toc477154941)

[Figure 5 Results and scalability for NAMD test case A 19](#_Toc477154942)

[Figure 6 Results and scalability for NAMD test case B 20](#_Toc477154943)

[Figure 7 PFARM results on Xeon Phi KNC 20](#_Toc477154944)

[Figure 8 PFARM results on GPUs 21](#_Toc477154945)

[Figure 9 Results on Titan 21](#_Toc477154946)

[Figure 10 New architecture results 22](#_Toc477154947)

[Figure 11 Result of second implementation of QCD on K40m GPU 23](#_Toc477154948)

[Figure 12 Result of second implementation of QCD on P100 GPU 24](#_Toc477154949)

[Figure 13 Result of second implementation of QCD on P100 GPU on larger test case 25](#_Toc477154950)

[Figure 14 Result of second implementation of QCD on KNC 26](#_Toc477154951)

[Figure 15 Result of second implementation of QCD on KNL 27](#_Toc477154952)

[Figure 16 Result of second implementation of QCD on KNL on a larger test case 28](#_Toc477154953)

[Figure 17 AUSURF performances on GPU 29](#_Toc477154954)

[Figure 18 CNT performances on GPU 29](#_Toc477154955)

List of Tables

Table 1 Main co-processors specifications 2

Table 2 Synthetic benchmarks relusts on K40 and KNC 30

Table 3 SPECFEM 3D GLOBE results 30

References and Applicable Documents

1. <http://www.prace-ri.eu>
2. The Unified European Application Benchmark Suite – <http://www.prace-ri.eu/ueabs/>
3. D7.4 Unified European Applications Benchmark Suite – Mark Bull et al. – 2013
4. <http://www.nvidia.com/object/quadro-design-and-manufacturing.html>

1. <https://userinfo.surfsara.nl/systems/cartesius/description>
2. MareNostrum III User’s Guide Barcelona Supercomputing Center – <https://www.bsc.es/support/MareNostrum3-ug.pdf>

1. <http://www.idris.fr/eng/ouessant/>
2. PFARM reference – <https://hpcforge.org/plugins/mediawiki/wiki/pracewp8/images/3/34/Pfarm_long_lug.pdf>
3. Solvent-Driven Preferential Association of Lignin with Regions of Crystalline Cellulose in Molecular Dynamics Simulation – Benjamin Lindner et al. – Biomacromolecules, 2013
4. NAMD website – <http://www.ks.uiuc.edu/Research/namd/>
5. SHOC source repository – <https://github.com/vetter/shoc>
6. Parallelizing the QUDA Library for Multi-GPU Calculations in Lattice Quantum Chromodynamics – R. Babbich, M. Clark and B. Joo – SC 10 (Supercomputing 2010)
7. Lattice QCD on Intel Xeon Phi – B. Joo, D. D. Kalamkar, K. Vaidyanathan, M. Smelyanskiy, K. Pamnany, V. W. Lee, P. Dubey, W. Watson III – International Supercomputing Conference (ISC’13), 2013

List of Acronyms and Abbreviations

aisbl Association International Sans But Lucratif   
 (legal form of the PRACE-RI)

BCO Benchmark Code Owner

CoE Center of Excellence

CPU Central Processing Unit

CUDA Compute Unified Device Architecture (NVIDIA)

DARPA Defense Advanced Research Projects Agency

DEISA Distributed European Infrastructure for Supercomputing Applications EU project by leading national HPC centres

DoA Description of Action (formerly known as DoW)

EC European Commission

EESI European Exascale Software Initiative

EoI Expression of Interest

ESFRI European Strategy Forum on Research Infrastructures

GB Giga (= 230 ~ 109) Bytes (= 8 bits), also GByte

Gb/s Giga (= 109) bits per second, also Gbit/s

GB/s Giga (= 109) Bytes (= 8 bits) per second, also GByte/s

GÉANT Collaboration between National Research and Education Networks to build a multi-gigabit pan-European network. The current EC-funded project as of 2015 is GN4.

GFlop/s Giga (= 109) Floating point operations (usually in 64-bit, i.e. DP) per second, also GF/s

GHz Giga (= 109) Hertz, frequency =109 periods or clock cycles per second

GPU Graphic Processing Unit

HET High Performance Computing in Europe Taskforce. Taskforce by representatives from European HPC community to shape the European HPC Research Infrastructure. Produced the scientific case and valuable groundwork for the PRACE project.

HMM Hidden Markov Model

HPC High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing

HPL High Performance LINPACK

ISC International Supercomputing Conference; European equivalent to the US based SCxx conference. Held annually in Germany.

KB Kilo (= 210 ~103) Bytes (= 8 bits), also KByte

LINPACK Software library for Linear Algebra

MB Management Board (highest decision making body of the project)

MB Mega (= 220 ~ 106) Bytes (= 8 bits), also MByte

MB/s Mega (= 106) Bytes (= 8 bits) per second, also MByte/s

MFlop/s Mega (= 106) Floating point operations (usually in 64-bit, i.e. DP) per second, also MF/s

MooC Massively open online Course

MoU Memorandum of Understanding

MPI Message Passing Interface

NDA Non-Disclosure Agreement. Typically signed between vendors and customers working together on products prior to their general availability or announcement.

PA Preparatory Access (to PRACE resources)

PATC PRACE Advanced Training Centres

PRACE Partnership for Advanced Computing in Europe; Project Acronym

PRACE 2 The upcoming next phase of the PRACE Research Infrastructure following the initial five year period.

PRIDE Project Information and Dissemination Event

RI Research Infrastructure

TB Technical Board (group of Work Package leaders)

TB Tera (= 240 ~ 1012) Bytes (= 8 bits), also TByte

TCO Total Cost of Ownership. Includes recurring costs (e.g. personnel, power, cooling, maintenance) in addition to the purchase cost.

TDP Thermal Design Power

TFlop/s Tera (= 1012) Floating-point operations (usually in 64-bit, i.e. DP) per second, also TF/s

Tier-0 Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1

UNICORE Uniform Interface to Computing Resources. Grid software for seamless access to distributed resources.

List of Project Partner Acronyms

BADW-LRZ Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Germany (3rd Party to GCS)

BILKENT Bilkent University, Turkey (3rd Party to UYBHM)

BSC Barcelona Supercomputing Center - Centro Nacional de Supercomputacion, Spain

CaSToRC Computation-based Science and Technology Research Center, Cyprus

CCSAS Computing Centre of the Slovak Academy of Sciences, Slovakia

CEA Commissariat à l’Energie Atomique et aux Energies Alternatives, France (3 rd Party to GENCI)

CESGA Fundacion Publica Gallega Centro Tecnológico de Supercomputación de Galicia, Spain, (3rd Party to BSC)

CINECA CINECA Consorzio Interuniversitario, Italy

CINES Centre Informatique National de l’Enseignement Supérieur, France (3 rd Party to GENCI)

CNRS Centre National de la Recherche Scientifique, France (3 rd Party to GENCI)

CSC CSC Scientific Computing Ltd., Finland

CSIC Spanish Council for Scientific Research (3rd Party to BSC)

CYFRONET Academic Computing Centre CYFRONET AGH, Poland (3rd party to PNSC)

EPCC EPCC at The University of Edinburgh, UK

ETHZurich (CSCS) Eidgenössische Technische Hochschule Zürich – CSCS, Switzerland

FIS FACULTY OF INFORMATION STUDIES, Slovenia (3rd Party to ULFME)

GCS Gauss Centre for Supercomputing e.V.

GENCI Grand Equipement National de Calcul Intensiv, France

GRNET Greek Research and Technology Network, Greece

INRIA Institut National de Recherche en Informatique et Automatique, France (3 rd Party to GENCI)

IST Instituto Superior Técnico, Portugal (3rd Party to UC-LCA)

IUCC INTER UNIVERSITY COMPUTATION CENTRE, Israel

JKU Institut fuer Graphische und Parallele Datenverarbeitung der Johannes Kepler Universitaet Linz, Austria

JUELICH Forschungszentrum Juelich GmbH, Germany

KTH Royal Institute of Technology, Sweden (3 rd Party to SNIC)

LiU Linkoping University, Sweden (3 rd Party to SNIC)

NCSA NATIONAL CENTRE FOR SUPERCOMPUTING APPLICATIONS, Bulgaria

NIIF National Information Infrastructure Development Institute, Hungary

NTNU The Norwegian University of Science and Technology, Norway (3rd Party to SIGMA)

NUI-Galway National University of Ireland Galway, Ireland

PRACE Partnership for Advanced Computing in Europe aisbl, Belgium

PSNC Poznan Supercomputing and Networking Center, Poland

RISCSW RISC Software GmbH

RZG Max Planck Gesellschaft zur Förderung der Wissenschaften e.V., Germany (3 rd Party to GCS)

SIGMA2 UNINETT Sigma2 AS, Norway

SNIC Swedish National Infrastructure for Computing (within the Swedish Science Council), Sweden

STFC Science and Technology Facilities Council, UK (3rd Party to EPSRC)

SURFsara Dutch national high-performance computing and e-Science support center, part of the SURF cooperative, Netherlands

UC-LCA Universidade de Coimbra, Labotatório de Computação Avançada, Portugal

UCPH Københavns Universitet, Denmark

UHEM Istanbul Technical University, Ayazaga Campus, Turkey

UiO University of Oslo, Norway (3rd Party to SIGMA)

ULFME UNIVERZA V LJUBLJANI, Slovenia

UmU Umea University, Sweden (3 rd Party to SNIC)

UnivEvora Universidade de Évora, Portugal (3rd Party to UC-LCA)

UPC Universitat Politècnica de Catalunya, Spain (3rd Party to BSC)

UPM/CeSViMa Madrid Supercomputing and Visualization Center, Spain (3rd Party to BSC)

USTUTT-HLRS Universitaet Stuttgart – HLRS, Germany (3rd Party to GCS)

VSB-TUO VYSOKA SKOLA BANSKA - TECHNICKA UNIVERZITA OSTRAVA, Czech Republic

WCNS Politechnika Wroclawska, Poland (3rd party to PNSC)

Executive Summary

This document describes an accelerator benchmark suite, a set of 11 codes that includes 1 synthetic benchmark and 10 commonly used applications. The key focus of this task has been exploiting accelerators or co-processors to improve the performance of real applications. It aims at providing a set of scalable, currently relevant and publically available codes and datasets.

This work has been undertaken by Task7.2B "Accelerator Benchmarks" in the PRACE Fourth Implementation Phase (PRACE-4IP) project.

Most of the selected application are a subset of the Unified European Applications Benchmark Suite (UEABS) [2][3]. One application and a synthetic benchmark have been added.

As a result, selected codes are: ALYA, Code\_Saturne, CP2K, GROMACS, GPAW, NAMD, PFARM, QCD, Quantum Espresso, SHOC and SPECFEM3D.

For each code either two or more test case datasets have been selected. These are described in this document, along with a brief introduction to the application codes themselves. For each code, some sample results are presented, from first run on leading edge systems and prototypes.

## Introduction

The work produced within this task is an extension of the UEABS for accelerators. This document will cover each code, presenting the code as well as the test cases defined for the benchmarks and the first results that have been recorded on various accelerator systems.

As the UEABS, this suite aims to present results for many scientific fields that can use HPC accelerated resources. Hence, it will help the European scientific communities to decide in terms of infrastructures they could buy in a near future. We focus on Intel Xeon Phi coprocessors and NVIDIA GPU cards for benchmarking as they are the two most wide-spread accelerated resources available now.

Section 2 will present both types of accelerator systems, Xeon Phi and GPU card along with architecture examples. Section \ref{applications} gives a description of each of the selected applications, together with the test case datasets, and presents some sample results. Section \ref{conclusion} outlines further work on, and using, the suite.

## Targeted architectures

This suite is targeting accelerator cards, more specifically the Intel Xeon Phi and NVIDIA GPU architecture. This section will quickly describe them and will present the four machines, the benchmarks ran on.

### Co-processor description

Scientific computing using co-processors has gained popularity in recent years. First the utility of GPUs has been demonstrated and evaluated in several application domains [4]. As a response to NVIDIA’s supremacy in this field, Intel designed Xeon Phi cards.

Architectures and programming models of co-processors may differ from CPUs and vary among different co-processor types. The main challenges are the high-level parallelism ability required from software and the fact that code may have to be offloaded to the accelerator card.

The Table 1 enlightens this fact:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Intel Xeon Phi | | NVIDIA GPU | |
|  | 5110P (KNC) | 7250 (KNL) | K40m | P100 |
| public availability date | Nov-12 | Jun-16 | Jun-13 | May-16 |
| theoretical peak perf | 1,011 GF/s | 3,046 GF/s | 1,430 GF/s | 5,300 GF/s |
| offload required | yes | no | yes | yes |
| max number of thread/cuda cores | 240 | 272 | 2880 | 3584 |

Table 1 Main co-processors specifications

### Systems description

The benchmark suite has been officially granted access to 4 different machines hosted by PRACE partners. Most results presented in this paper were obtained on these machines but some of the simulation has run on similar ones. This section will cover specifications of the sub mentioned 4 official systems while the few other ones will be presented along with concerned results.

As it can be noticed on the previous section, leading edge architectures have been available quite recently and some code couldn't run on it yet. Results will be completed in a near future and will be delivered with an update of the benchmark suite. Still, presented performances are a good indicator about potential efficiency of codes on both Xeon Phi and NVIDIA GPU platforms.

#### K40 cluster

The SURFsara institute in The Netherlands granted access to Cartesius which has a GPU island (installed May 2014) with following specifications [5]:

* 66 Bullx B515 GPU accelerated nodes
  + 2x 8-core 2.5 GHz Intel Xeon E5-2450 v2 (Ivy Bridge) CPUs/node
  + 2x NVIDIA Tesla K40m GPGPUs/node
  + 96 GB/node DDR3-1600 RAM
* Total theoretical peak performance (Ivy Bridge + K40m) 1,056 cores + 132 GPUs: 210 TF/s

The interconnect has a fully non-blocking fat-tree topology. Every node has two ConnectX-3 InfiniBand FDR adapters: one per GPGPU.

#### Xeon Phi 5110P cluster

The Barcelona Supercomputing Center (BSC) in Spain granted access to MareNostrum III which features KNC nodes (upgrade June 2013). Here's the description of this partition [6]:

* 42 hybrid nodes contain:
  + 8x 8G DDR3–1600 DIMMs (4GB/core), total: 64GB/node
* 2x Xeon Phi 5110P accelerators
* Interconnection networks:
  + Infiniband Mellanox FDR10: High bandwidth network used by parallel applications communications (MPI)
  + Gigabit Ethernet: 10GbitEthernet network used by the GPFS Filesystem.

#### P100 cluster

GENCI granted access to the Ouessant prototype at IDRIS in France (installed September 2016). It is composed of 12 IBM Minsky compute nodes with each containing [7]:

* Compute nodes
  + POWER8+ sockets, 10 cores, 8 threads per core (or 160 threads per node)
  + 128 GB of DDR4 memory (bandwidth > 9 GB/s per core)
  + 4 NVIDIA’s new generation Pascal P100 GPUs, 16 GB of HBM2 memory
* Interconnect
  + 4 NVLink interconnects (40GB/s of bi-directional bandwidth per interconnect); each GPU card is connected to a CPU with 2 NVLink interconnects and another GPU with 2 interconnects remaining
  + A Mellanox EDR IB CAPI interconnect network (1 interconnect per node)

#### Xeon Phi 7250 cluster

GENCI also granted access to the Frioul prototype at CINES in France (installed December 2016). It is composed of 48 Intel KNL compute nodes:

* Peak performance of 146 TF/s
* Interconnect: Infiniband IB 4x FDR
* File system: Lustre, more than 5 PB usable and a maximum bandwidth of 105 GB/s

## Benchmark suite description

This part will cover each code, presenting the interest for the scientific community as well as the test cases defined for the benchmarks.

As an extension to the EUABS, most of the codes presented in this suite are included in the later. Exceptions are PFARM which comes from PRACE-2IP [8] and SHOC a synthetic benchmark suite.

### Alya

Alya is a high performance computational mechanics code that can solve different coupled mechanics problems: incompressible/compressible flows, solid mechanics, chemistry, excitable media, heat transfer and Lagrangian particle transport. It is one single code. There are no particular parallel or individual platform versions. Modules, services and kernels can be compiled individually and used a la carte. The main discretisation technique employed in Alya is based on the variational multiscale finite element method to assemble the governing equations into Algebraic systems. These systems can be solved using solvers like GMRES, Deflated Conjugate Gradient, pipelined CG together with preconditioners like SSOR, Restricted Additive Schwarz, etc. The coupling between physics solved in different computational domains (like fluid-structure interactions) is carried out in a multi-code way, using different instances of the same executable. Asynchronous coupling can be achieved in the same way in order to transport Lagrangian particles.

#### Code description

The code is parallelised with MPI and OpenMP. Two OpenMP strategies are available, without and with a colouring strategy to avoid ATOMICs during the assembly step. A CUDA version is also available for the different solvers. Alya has been also compiled for MIC (Intel Xeon Phi).

Alya is written in Fortran 1995 and the incompressible fluid module, present in the benchmark suite, is freely available. This module solves the Navier-Stokes equations using an Orthomin \ref{} method for the pressure Schur complement. This method is an algebraic split strategy which converges to the monolithic solution. At each linearisation step, the momentum is solved twice and the continuity equation is solved once or twice depending on whether the momentum preserving or the continuity preserving algorithm is selected.

#### Test cases description

Cavity-hexaedra elements (10M elements)

This test is the classical lid-driven cavity. The problem geometry is a cube of dimensions 1x1x1. The fluid properties are density=1.0 and viscosity=0.01. Dirichlet boundary conditions are applied on all sides, with three no-slip walls and one moving wall with velocity equal to 1.0, which corresponds to a Reynolds number of 100. The Reynolds number is low so the regime is laminar and turbulence modelling is not necessary. The domain is discretised into 9800344 hexaedra elements. The solvers are the GMRES method for the momentum equations and the Deflated Conjugate Gradient to solve the continuity equation. This test case can be run using pure MPI parallelisation or the hybrid MPI/OpenMP strategy.

Cavity-hexaedra elements (30M elements)

This is the same cavity test as before but with 30M of elements. Note that a mesh multiplication strategy enables one to multiply the number of elements by powers of 8, by simply activating the corresponding option in the ker.dat file.

Cavity-hexaedra elements-GPU version (10M elements)

This is the same test as Test case 1, but using the pure MPI parallelisation strategy with acceleration of the algebraic solvers using GPUs.

### Code Saturne

Code Saturne is an open-source CFD software package developed by EDF R&D since 1997 and open-source since 2007. The Navier-Stokes equations are discretised following a finite volume method approach. The code can handle any type of mesh built with any type of cell/grid structure. Incompressible and compressible flows can be simulated, with or without heat transfer, and a range of turbulence models is available. The code can also be coupled with itself or other software to model some multiphysics problems (fluid-structure, fluid-conjugate heat transfer, for instance).

#### Code description

Parallelism is handled by distributing the domain over the processors (several partitioning tools are available, either internally, i.e. SFC Hilbert and Morton, or through external libraries, i.e. METIS Serial, ParMETIS, Scotch Serial, PT-SCOTCH. Communications between subdomains are performed through MPI. Hybrid parallelism using OpenMP has recently been optimised for improved multicore performance.

For incompressible simulations, most of the time is spent during the computation of the pressure through Poisson equations. PETSc and HYPRE have recently been linked to the code to offer alternatives to the internal solvers to compute the pressure. The developer’s version of PETSc supports CUDA and will be used in this benchmark suite.

Code Saturne is written in C, F95 and Python. It is freely available under the GPL license.

#### Test cases description

Two test cases are dealt with, the former with a mesh made of tetrahedral cells and the latter with a mesh made of hexahedral cells. Both configurations are meant for incompressible laminar flows. Note that both configurations will also be used in the regular UEABS.

Flow in a 3-D lid-driven cavity (tetrahedral cells)

The geometry is very simple, i.e. a cube, but the mesh is built using tetrahedral cells. The Reynolds number is set to 400, and symmetry boundary conditions are applied in the spanwise direction. The case is modular and the mesh size can easily been varied. The largest mesh has about 13 million cells.

This test case is expected to scale efficiently to 1000+ nodes.

3-D Taylor-Green vortex flow (hexahedral cells)

The Taylor-Green vortex flow is traditionally used to assess the accuracy of CFD code numerical schemes. Periodicity is used in the 3 directions. The total kinetic energy (integral of the velocity) and enstrophy (integral of the vorticity) evolutions as a function of the time are looked at. Code Saturne is set for 2nd order time and spatial schemes, and three meshes are considered, containing 1283, 2563 and 5123 cells, respectively.

This test case is expected to scale efficiently to 4000+ nodes for the largest mesh.

### CP2K

CP2K is a quantum chemistry and solid state physics software package that can perform atomistic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. It can perform molecular dynamics, metadynamics, Quantum Monte Carlo, Ehrenfest dynamics, vibrational analysis, core level spectroscopy, energy minimisation, and transition state optimisation using NEB or dimer method.

CP2K provides a general framework for different modelling methods such as DFT using the mixed Gaussian and plane waves approaches GPW and GAPW. Supported theory levels include DFTB, LDA, GGA, MP2, RPA, semi-empirical methods (AM1, PM3, PM6, RM1, MNDO, …), and classical force fields (AMBER, CHARMM, …).

#### Code description

Parallelisation is achieved using a combination of OpenMP-based multi-threading and MPI.

Offloading for accelerators is implemented through CUDA and OpenCL for GPGPUs and through OpenMP for MIC (Intel Xeon Phi).

CP2K is written in Fortran 2003 and freely available under the GPL license.

#### Test cases description

LiH-HFX

This is a single-point energy calculation for a particular configuration of a 216 atom Lithium Hydride crystal with 432 electrons in a 12.3 Å3 (Angstroms cubed) cell. The calculation is performed using a density functional theory (DFT) algorithm with Gaussian and Augmented Plane Waves (GAPW) under the hybrid Hartree-Fock exchange (HFX) approximation. These types of calculations are generally around one hundred times the computational cost of a standard local DFT calculation, although the cost of the latter can be reduced by using the Auxiliary Density Matrix Method (ADMM). Using OpenMP is of particular benefit here as the HFX implementation requires a large amount of memory to store partial integrals. By using several threads, fewer MPI processes share the available memory on the node and thus enough memory is available to avoid recomputing any integrals on-the-fly, improving performance.

This test case is expected to scale efficiently to 1000+ nodes.

H2O-DFT-LS

This is a single-point energy calculation for 2048 water molecules in a 39 Å3 box using linear-scaling DFT. A local-density approximation (LDA) functional is used to compute the Exchange-Correlation energy in combination with a DZVP MOLOPT basis set and a 300 Ry cut-off. For large systems the linear-scaling approach for solving Self-Consistent-Field equations should be much cheaper computationally than using standard DFT, and allow scaling up to 1 million atoms for simple systems. The linear scaling cost results from the fact that the algorithm is based on an iteration on the density matrix. The cubically-scaling orthogonalisation step of standard DFT is avoided and the key operations are sparse matrix-matrix multiplications, which have a number of non-zero entries that scale linearly with system size. These are implemented efficiently in CP2K's DBCSR library.

This test case is expected to scale efficiently to 4000+ nodes.

### GPAW

GPAW is a density-functional theory (DFT) program for ab-initio electronic structure calculations using the projector augmented wave method. It uses a uniform real-space grid representation of the electronic wavefunctions, that allows for excellent computational scalability and systematic converge properties.

#### Code description

GPAW is written mostly in Python, but includes also computational kernels written in C as well as leveraging external libraries such as NumPy, BLAS and ScaLAPACK. Parallelisation is based on message-passing using MPI with no threading. Development branches for GPGPUs and MICs include support for offloading to accelerators using either CUDA or pyMIC, respectively. GPAW is freely available under the GPL license.

#### Test cases description

Carbon Nanotube

A ground state calculation for a carbon nanotube in vacuum. By default uses a 6-6-10 nanotube with 240 atoms (freely adjustable) and serial LAPACK with an option to use ScaLAPACK.

This benchmark is aimed at smaller systems, with an intended scaling range of up to 10 nodes.

Copper Filament

A ground state calculation for a copper filament in vacuum. By default uses a 2x2x3 FCC lattice with 71 atoms (freely adjustable) and ScaLAPACK for parallelisation.

This benchmark is aimed at larger systems, with an intended scaling range of up to 100 nodes. A lower limit on the number of nodes may be imposed by the amount of memory required, which can be adjusted to some extent with the run parameters (e.g. lattice size or grid spacing).

### GROMACS

GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles.

It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers.

GROMACS supports all the usual algorithms you expect from a modern molecular dynamics implementation, and some additional features:

GROMACS provides extremely high performance compared to all other programs. A lot of algorithmic optimisations have been introduced in the code; for instance the calculation of the virial has been extracted from the innermost loops over pairwise interactions, and we use our own software routines to calculate the inverse square root. In GROMACS 4.6, on almost all common computing platforms, the innermost loops are written in C using intrinsic functions that the compiler transforms to SIMD machine instructions, to utilise the available instruction-level parallelism. These kernels are available in either single and double precision, and support all different kinds of SIMD support found in x86-family processors available in January 2013.

#### Code description

Parallelisation is achieved using combined OpenMP and MPI.

Offloading for accelerators is implemented through CUDA for GPGPUs and through OpenMP for MIC (Intel Xeon Phi).

GROMACS is written in C/C++ and freely available under the GPL license.

#### Test cases description

GluCL Ion Channel

The ion channel system is the membrane protein GluCl, which is a pentameric chloride channel embedded in a lipid bilayer. The GluCl ion channel was embedded in a DOPC membrane and solvated in TIP3P water. This system contains 142k atoms, and is a quite challenging parallelisation case due to the small size. However, it is likely one of the most wanted target sizes for biomolecular simulations due to the importance of these proteins for pharmaceutical applications. It is particularly challenging due to a highly inhomogeneous and anisotropic environment in the membrane, which poses hard challenges for load balancing with domain decomposition.

This test case was used as the “Small” test case in previous 2IP and 3IP PRACE phases. It is included in the package's version 5.0 benchmark cases. It is reported to scale efficiently up to 1000+ cores on x86 based systems.

Lignocellulose

A model of cellulose and lignocellulosic biomass in an aqueous solution [9]. This system of 3.3 million atoms is inhomogeneous. This system uses reaction-field electrostatics instead of PME and therefore scales well on x86. This test case was used as the “Large” test case in previous PRACE 2IP and 3IP projects. It is reported in previous PRACE projects to scale efficiently up to 10000+ x86 cores.

### NAMD

NAMD is a widely used molecular dynamics application designed to simulate bio-molecular systems on a wide variety of compute platforms. NAMD is developed by the “Theoretical and Computational Biophysics Group” at the University of Illinois at Urbana Champaign. In the design of NAMD particular emphasis has been placed on scalability when utilising a large number of processors. The application can read a wide variety of different file formats, for example force fields, protein structures, which are commonly used in bio-molecular science. A NAMD license can be applied for on the developer’s website free of charge. Once the license has been obtained, binaries for a number of platforms and the source can be downloaded from the website. Deployment areas of NAMD include pharmaceutical research by academic and industrial users. NAMD is particularly suitable when the interaction between a number of proteins or between proteins and other chemical substances is of interest. Typical examples are vaccine research and transport processes through cell membrane proteins.

#### Code description

NAMD is written in C++ and parallelised using Charm++ parallel objects, which are implemented on top of MPI, supporting both pure MPI and hybrid parallelisation [10].

Offloading for accelerators is implemented for both GPGPUs and MIC (Intel Xeon Phi).

#### Test cases description

The datasets are based on the original "Satellite Tobacco Mosaic Virus (STMV)" dataset from the official NAMD site. The memory optimised build of the package and data sets are used in benchmarking. Data are converted to the appropriate binary format used by the memory optimised build.

STMV.1M

This is the original STMV dataset from the official NAMD site. The system contains roughly 1 million atoms. This data set scales efficiently up to 1000+ x86 Ivy Bridge cores.

STMV.8M

This is a 2x2x2 replication of the original STMV dataset from the official NAMD site. The system contains roughly 8 million atoms. This data set scales efficiently up to 6000 x86 Ivy Bridge cores.

### PFARM

PFARM is part of a suite of programs based on the ‘R-matrix’ ab-initio approach to the varitional solution of the many-electron Schrödinger equation for electron-atom and electron-ion scattering. The package has been used to calculate electron collision data for astrophysical applications (such as: the interstellar medium, planetary atmospheres) with, for example, various ions of Fe and Ni and neutral O, plus other applications such as data for plasma modelling and fusion reactor impurities. The code has recently been adapted to form a compatible interface with the UKRmol suite of codes for electron (positron) molecule collisions thus enabling large-scale parallel ‘outer-region’ calculations for molecular systems as well as atomic systems.

#### Code description

In order to enable efficient computation, the external region calculation takes place in two distinct stages, named EXDIG and EXAS, with intermediate files linking the two. EXDIG is dominated by the assembly of sector Hamiltonian matrices and their subsequent eigensolutions. EXAS uses a combined functional/domain decomposition approach where good load-balancing is essential to maintain efficient parallel performance. Each of the main stages in the calculation is written in Fortran 2003 (or Fortran 2003-compliant Fortran 95), is parallelised using MPI and is designed to take advantage of highly optimised, numerical library routines. Hybrid MPI / OpenMP parallelisation has also been introduced into the code via shared memory enabled numerical library kernels.

Accelerator-based implementations have been implemented for both EXDIG and EXAS. EXAS uses offloading via MAGMA (or MKL) for sector Hamiltonian diagonalisations on Intel Xeon Phi and GPGPU accelerators. EXDIG uses combined MPI and OpenMP to distribute the scattering energy calculations on CPUs efficiently both across and within Intel Xeon Phico-processors.

#### Test cases description

External region R-matrix propagations take place over the outer partition of configuration space, including the region where long-range potentials remain important. The radius of this region is determined from the user input and the program decides upon the best strategy for dividing this space into multiple sub-regions (or sectors). Generally, a choice of larger sector lengths requires the application of larger numbers of basis functions (and therefore larger Hamiltonian matrices) in order to maintain accuracy across the sector and vice-versa. Memory limits on the target hardware may determine the final preferred configuration for each test case.

FeIII

This is an electron-ion scattering case with 1181 channels. Hamiltonian assembly in the coarse region applies 10 Legendre functions leading to Hamiltonian matrix diagonalisations of order 11810. In the fine region up to 30 Legendre functions may be applied leading to Hamiltonian matrices of order 35430. The number of sector calculations is likely to range from about 15 to over 30 depending on the user specifications. Several thousand scattering energies will be used in the calculation.

In the current model, parallelism in EXDIG is limited to the number of sector calculations, i.e around 30 accelerator nodes. Parallelism in EXAS is limited by the number of scattering energies, so we would expect this to reach into the hundreds of nodes.

Methane

The dataset is an electron-molecule calculation with 1361 channels. Hamiltonian dimensions are therefore estimated between 13610 and ~40000. The length of the external region required is relatively long, leading to more numerous sectors calculations (estimated to between 25 and 50). The calculation will require many thousands of scattering energies.

EXDIG scaling expected up to 50 accelerator nodes. EXAS scaling expected on hundreds to low thousands of nodes.

### QCD

Matter consists of atoms, which in turn consist of nuclei and electrons. The nuclei consist of neutrons and protons, which comprise quarks bound together by gluons.

The theory of how quarks and gluons interact to form nucleons and other elementary particles is called Quantum Chromo Dynamics (QCD). For most problems of interest, it is not possible to solve QCD analytically, and instead numerical simulations must be performed. Such “Lattice QCD” calculations are very computationally intensive, and occupy a significant percentage of all HPC resources worldwide.

#### Code description

The QCD benchmark benefits of two different implementations described below.

First implementation

The MILC code is a freely-available suite for performing Lattice QCD simulations, developed over many years by a collaboration of researchers (physics.indiana.edu/~sg/milc.html).

The benchmark used here is derived from the MILC code (v6), and consists of a full conjugate gradient solution using Wilson fermions. The benchmark is consistent with “QCD kernel E” in the full UAEBS, and has been adapted so that it can efficiently use accelerators as well as traditional CPUs.

The implementation for accelerators has been achieved using the “targetDP” programming model [http://ccpforge.cse.rl.ac.uk/svn/ludwig/trunk/targetDP/README], a lightweight abstraction layer designed to allow the same application source code to be able to target multiple architectures, e.g. NVidia GPUs and multicore/manycore CPUs, in a performance portable manner. The targetDP syntax maps, at compile time, to either NVIDIA CUDA (for execution on GPUs) or OpenMP+vectorisation (for implementation on multi/manycore CPUs including Intel Xeon Phi). The base language of the benchmark is C and MPI is used for node-level parallelism.

Second implementation

The QCD Accelerator Benchmark suite Part 2 consists of two kernels, the QUDA [12] and the QPhix [13] library. The library QUDA is based on CUDA and optimized for running on NVIDIA GPUs (https://lattice.github.io/quda/). The library QPhix consists of routines which are optimized for various of INTEL INTRINSICS, which contains optimized routines for KNC and KNL's (http://jeffersonlab.github.io/qphix/).

The benchmark code is using the provided Conjugated Gradient benchmark functions of the libraries.

#### Test cases description

Lattice QCD involves discretisation of space-time into a lattice of points, where the extent of the lattice in each of the 3 spatial and 1 temporal dimensions can be chosen. This means that the benchmark is very flexible, where the size of the lattice can be varied with the size of the computing system in use (weak scaling) or can be fixed (strong scaling). For testing on a single node, then 64x64x32x8 is a reasonable size, since this fits on a single Intel Xeon Phi or a single GPU. For larger numbers of nodes, the lattice extents can be increased accordingly, keeping the geometric shape roughly similar.

### Quantum Espresso

QUANTUM ESPRESSO is an integrated suite of computer codes for electronic-structure calculations and materials modelling, based on density-functional theory, plane waves, and pseudopotentials (norm-conserving, ultrasoft, and projector-augmented wave). QUANTUM ESPRESSO stands for *opEn Source Package for Research in Electronic Structure, Simulation, and Optimisation*. It is freely available to researchers around the world under the terms of the GNU General Public License. QUANTUM ESPRESSO builds upon newly restructured electronic-structure codes that have been developed and tested by some of the original authors of novel electronic-structure algorithms and applied in the last twenty years by some of the leading materials modelling groups worldwide. Innovation and efficiency are still its main focus, with special attention paid to massively parallel architectures, and a great effort being devoted to user friendliness. QUANTUM ESPRESSO is evolving towards a distribution of independent and inter-operable codes in the spirit of an open-source project, where researchers active in the field of electronic-structure calculations are encouraged to participate in the project by contributing their own codes or by implementing their own ideas into existing codes.

QUANTUM ESPRESSO is written mostly in Fortran90, and parallelised using MPI and OpenMP and is released under a GPL license.

#### Code description

During 2011 a GPU-enabled version of Quantum ESPRESSO was publicly released. The code is currently developed and maintained by Filippo Spiga at the High Performance Computing Service - University of Cambridge (United Kingdom) and Ivan Girotto at the International Centre for Theoretical Physics (Italy). The initial work has been supported by the EC-funded PRACE and a SFI (Science Foundation Ireland, grant 08/HEC/I1450). At the time of writing, the project is self-sustained thanks to the dedication of the people involved and thanks to NVIDIA support in providing hardware and expertise in GPU programming.

The current public version of QE-GPU is 14.10.0 as it is the last version maintained as plug-in working on all QE 5.x versions. QE-GPU utilised phiGEMM (external) for CPU+GPU GEMM computation, MAGMA (external) to accelerate eigen-solvers and explicit CUDA kernel to accelerate compute-intensive routines. FFT capabilities on GPU are available only for serial computation due to the hard challenges posed in managing accelerators in the parallel distributed 3D-FFT portion of the code where communication is the dominant element that limits excellent scalability beyond hundreds of MPI ranks.

A version for Intel Xeon Phi (MIC) accelerators is not currently available.

#### Test cases description

PW-IRMOF\_M11

Full SCF calculation of a Zn-based isoreticular metal–organic framework (total 130 atoms) over 1 K point. Benchmarks run in 2012 demonstrated speedups due to GPUs (NVIDIA K20s, with respect to non-accelerated nodes) in the range 1.37 – 1.87, according to node count (maximum number of accelerators=8). Runs with current hardware technology and an updated version of the code are expected to exhibit higher speedups (probably 2-3x) and scale up to a couple hundred nodes.

PW-SiGe432

This is a SCF calculation of a Silicon-Germanium crystal with 430 atoms. Being a fairly large system, parallel scalability up to several hundred, perhaps a 1000 nodes is expected, with accelerated speed-ups likely to be of 2-3x.

### Synthetic benchmarks – SHOC

The Accelerator Benchmark Suite will also include a series of synthetic benchmarks. For this purpose, we choose the Scalable HeterOgeneous Computing (SHOC) benchmark suite, augmented with a series of benchmark examples developed internally. SHOC is a collection of benchmark programs testing the performance and stability of systems using computing devices with non-traditional architectures for general purpose computing. Its initial focus is on systems containing GPUs and multi-core processors, and on the OpenCL programming standard, but CUDA and OpenACC versions were added. Moreover, a subset of the benchmarks is optimised for the Intel Xeon Phi coprocessor. SHOC can be used on clusters as well as individual hosts.

The SHOC benchmark suite currently contains benchmark programs categorised by complexity. Some measure low-level 'feeds and speeds' behaviour (Level 0), some measure the performance of a higher-level operation such as a Fast Fourier Transform (FFT) (Level 1), and the others measure real application kernels (Level 2).

#### Code description

All benchmarks are MPI-enabled. Some will report aggregate metrics over all MPI ranks, others will only perform work for specific ranks.

Offloading for accelerators is implemented through CUDA and OpenCL for GPGPUs and through OpenMP for MIC (Intel Xeon Phi). For selected benchmarks OpenACC implementations are provided for GPGPUs. Multi-node parallelisation is achieved using MPI.

SHOC is written in C++ and is open-source and freely available.

#### Test cases description

The benchmarks contained in SHOC currently feature 4 different sizes for increasingly large systems. The size convention is as follows:

1. CPUs / debugging
2. Mobile/integrated GPUs
3. Discrete GPUs (e.g. GeForce or Radeon series)
4. HPC-focused or large memory GPUs (e.g. Tesla or Firestream Series)

In order to go even larger scale we plan to add a 5th level for massive supercomputers.

### SPECFEM3D

The software package SPECFEM3D simulates three-dimensional global and regional seismic wave propagation based upon the spectral-element method (SEM). All SPECFEM3D\_GLOBE software is written in Fortran90 with full portability in mind, and conforms strictly to the Fortran95 standard. It uses no obsolete or obsolescent features of Fortran77. The package uses parallel programming based upon the Message Passing Interface (MPI).

The SEM was originally developed in computational fluid dynamics and has been successfully adapted to address problems in seismic wave propagation. It is a continuous Galerkin technique, which can easily be made discontinuous; it is then close to a particular case of the discontinuous Galerkin technique, with optimised efficiency because of its tensorised basis functions. In particular, it can accurately handle very distorted mesh elements. It has very good accuracy and convergence properties. The spectral element approach admits spectral rates of convergence and allows exploiting hp-convergence schemes. It is also very well suited to parallel implementation on very large supercomputers as well as on clusters of GPU accelerating graphics cards. Tensor products inside each element can be optimised to reach very high efficiency, and mesh point and element numbering can be optimised to reduce processor cache misses and improve cache reuse. The SEM can also handle triangular (in 2D) or tetrahedral (3D) elements as well as mixed meshes, although with increased cost and reduced accuracy in these elements, as in the discontinuous Galerkin method.

In many geological models in the context of seismic wave propagation studies (except for instance for fault dynamic rupture studies, in which very high frequencies of supershear rupture need to be modelled near the fault) a continuous formulation is sufficient because material property contrasts are not drastic and thus conforming mesh doubling bricks can efficiently handle mesh size variations. This is particularly true at the scale of the full earth. Effects due to lateral variations in compressional-wave speed, shear-wave speed, density, a 3D crustal model, ellipticity, topography and bathyletry, the oceans, rotation, and self-gravitation are included. The package can accommodate full 21-parameter anisotropy as well as lateral variations in attenuation. Adjoint capabilities and finite-frequency kernel simulations are also included.

#### Test cases description

The small test case runs with 24 MPI tasks and has the following characteristics: NCHUNCK=6 NPROC=2 NEX=80 RECORD\_LENGHT=2.0. The large one runs with 150 MPI tasks and has the following characteristics: NCHUNCK=6 NPROC=5 NEX=80 RECORD\_LENGHT=2.0.

## Applications performances

This section presents results on targeted machines.

### ALYA



Figure 1 ALYA benchmark results

### Code Saturne

First test case has been run on ARCHER KNLs and the performance on KNLs has been looked at for several configurations, each of them using 64 MPI tasks per node and either 1, 2 or 4 hyper-threads (more MPI tasks) or threads (openMP) have been added for testing. The results are compared to ARCHER CPUs, in this case IvyBridge CPUs.

Figure 1 shows that the best configuration to run Code Saturne on KNLs is using 64 MPI tasks and 2 openMP threads per task.



Figure 2 Code Saturne's performance on KNLs. AMG is used as a solver in V4.2.2.

### CP2K

### GPAW



Figure 3 Parallel scaling of GPAW

### GROMACS

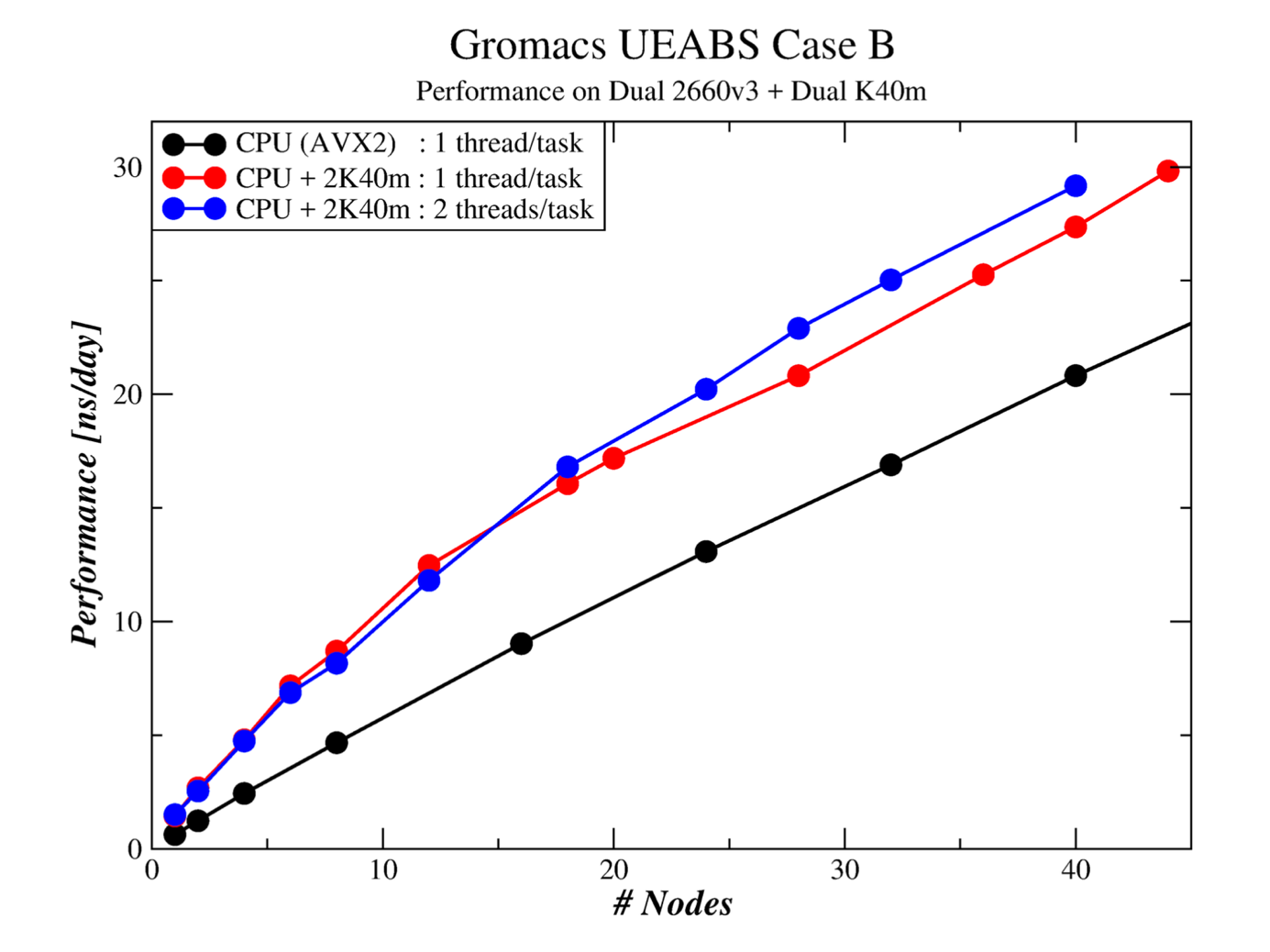


Figure 4 Parallel Scaling for GROMACS GPU

### NAMD

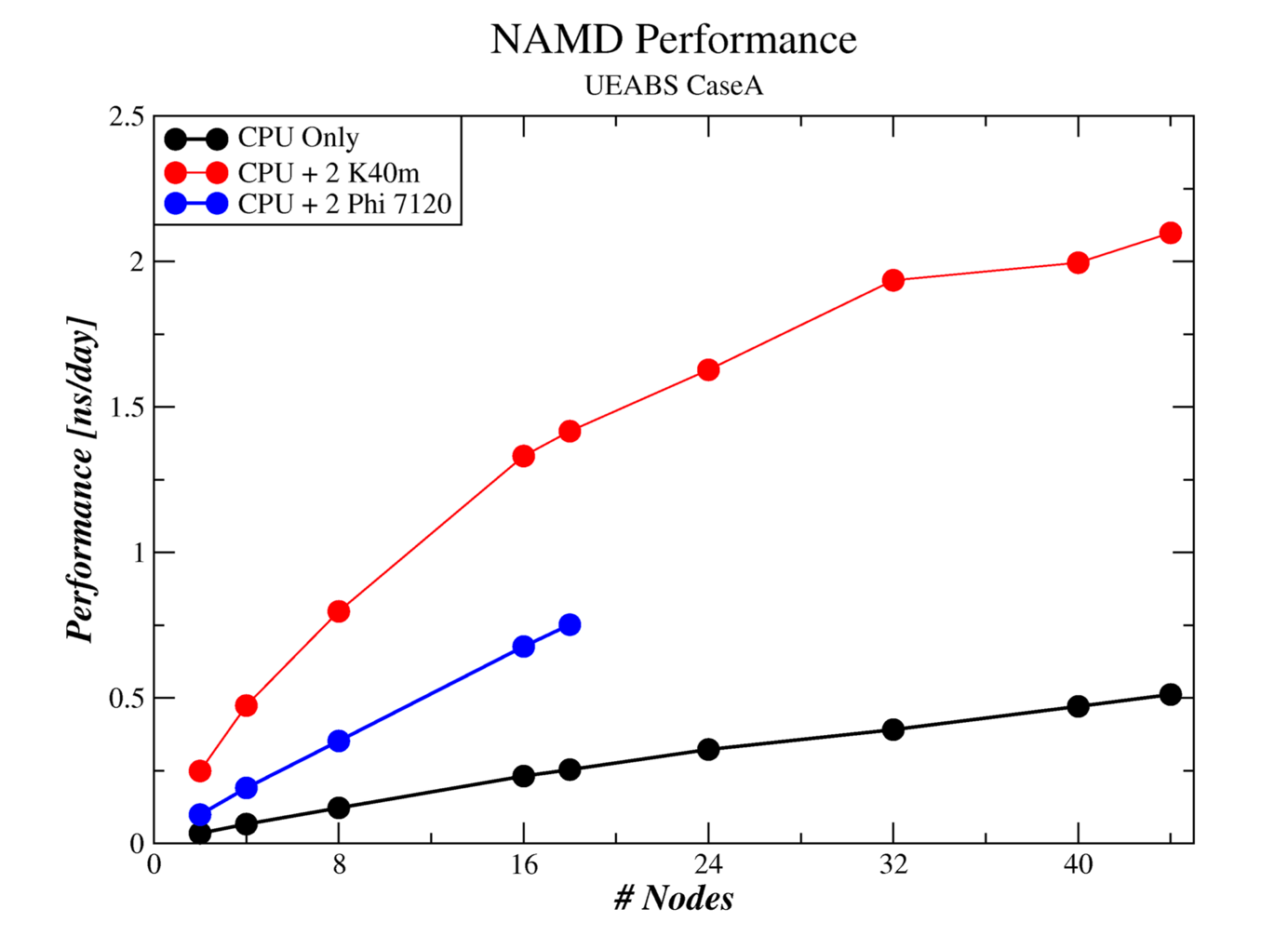


Figure 5 Results and scalability for NAMD test case A

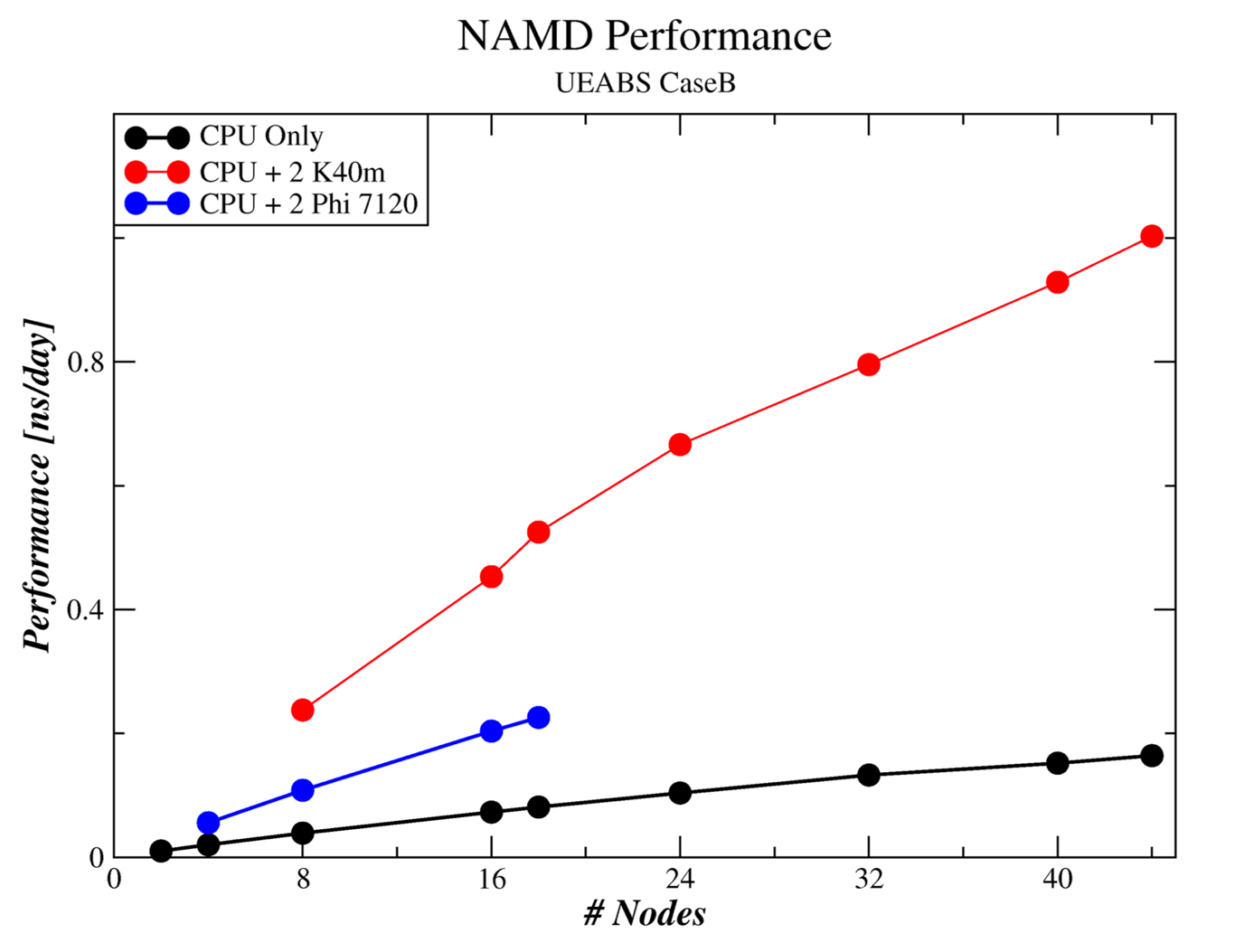


Figure 6 Results and scalability for NAMD test case B

### PFARM

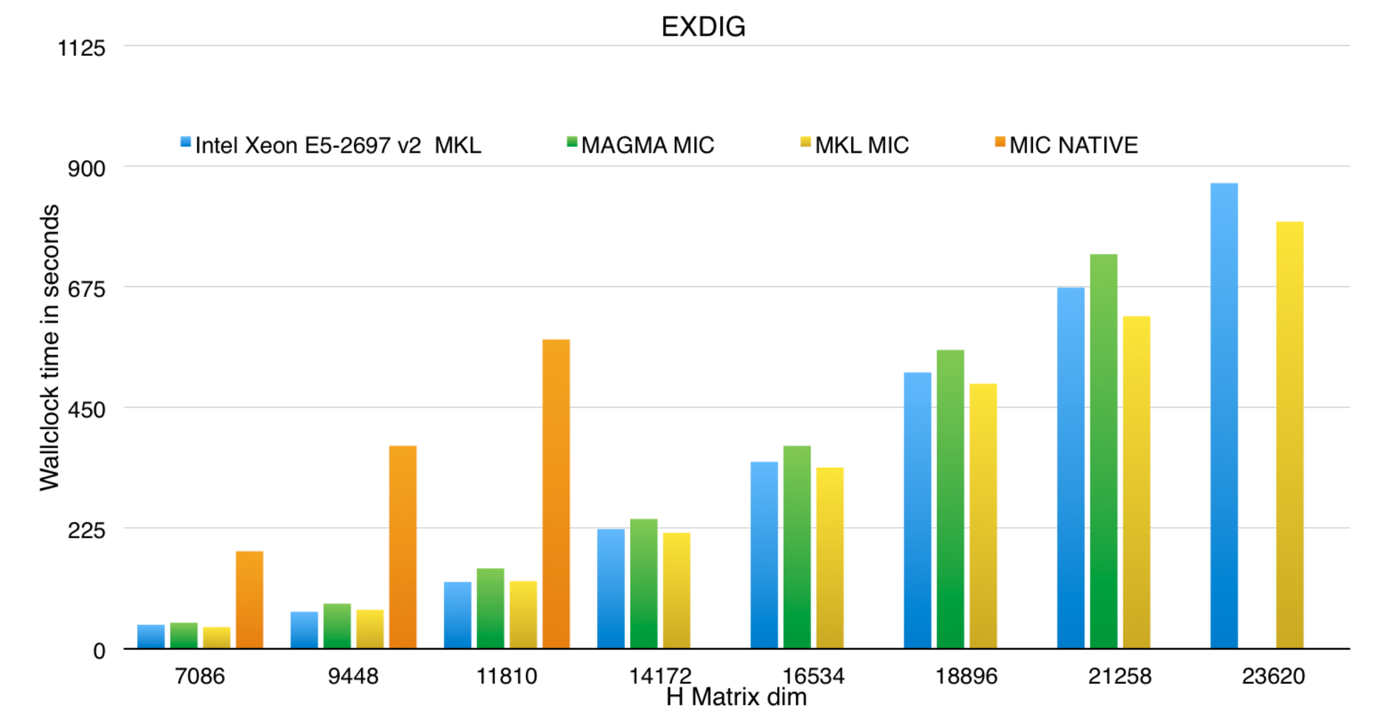


Figure 7 PFARM results on Xeon Phi KNC

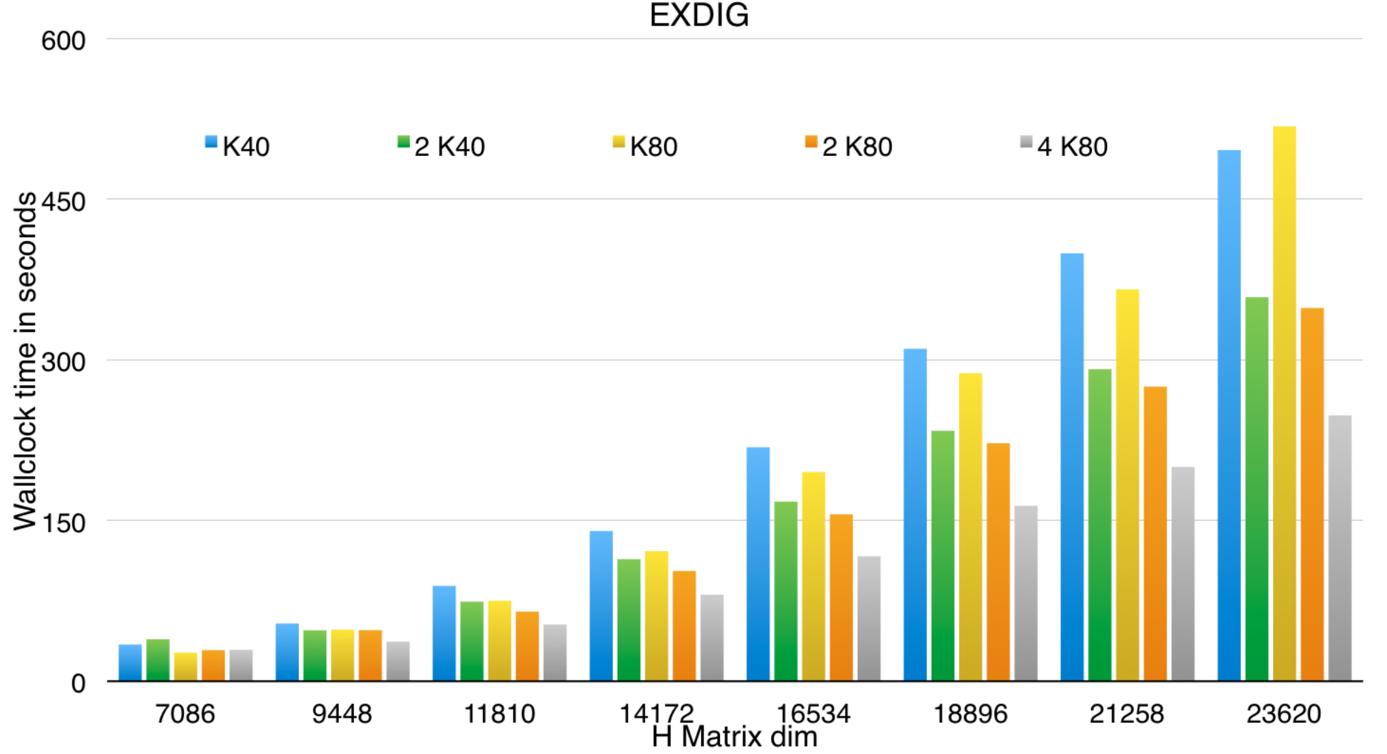


Figure 8 PFARM results on GPUs

### QCD

As stated in the description, QCD benchmark has two implementations.

#### First implementation



Figure 9 Results on Titan

The strong scaling, on Titan and ARCHER, for small (Figure 9, left) and large (Figure 9, right) problem sizes. For ARCHER, both CPUs are used per node. For Titan, we include results with and without GPU utilization



Figure 10 New architecture results

Figure 10 shows the time taken by the full MILC 64x64x64x8 test cases on traditional CPU, Intel Knights Landing Xeon Phi and NVIDIA P100 (Pascal) GPU architectures.

#### Second implementation

GPU results

Here are shown the benchmark results on PizDaint located in Switzerland at CSCS and the GPGPU-partition of Cartesius at Surfsara based in Netherland, Amsterdam. The runs are performed by using the provided bash-scripts. PizDaint has one P100 Pascal-GPU per node and two different test-cases are shown, the "Strong-Scaling mode with a random lattice configuration of size 32x32x32x96 and a "Weak-Scaling" mode with a configuration of local lattice size 48x48x48x24. The GPGPU nodes of Cartesius has two Kepler-GPU K40m per node and the "Strong-Scaling" test is shown for the case that one card per node and two cards per node are used. Benchmarks are done by using the Conjugated Gradient solver which solve a linear equation, D \* x = b, for the unknown solution "x" based on the clover improved Wilson Dirac operator "D" and a known right hand side "b".



Figure 11 Result of second implementation of QCD on K40m GPU

The figure shows strong scaling of the conjugate gradient solver on K40m GPUs on Cartesius. The lattice size is given by 32x32x32x96, which corresponds to a moderate lattice size nowadays. The test is perform with an mixed precision CG in double-double mode (red) and half-double mode (blue). The run is done on one GPU per node (filled) and two GPU nodes per node (non-filled).



Figure 12 Result of second implementation of QCD on P100 GPU

The figure shows strong scaling of the conjugate gradient solver on P100 GPUs on PizDaint. The lattice size is given by 32x32x32x96 similar to the strong scaling run on the K40m on Cartesius. The test is performed with mixed precision CG in double-double mode (red) and half-double mode (blue).



Figure 13 Result of second implementation of QCD on P100 GPU on larger test case

The figure shows strong scaling of the conjugate gradient solver on P100 GPUs on PizDaint. The lattice size is increase to 64x64x64x128, which is a commonly used large lattice nowadays. By increasing the lattice the scaling tests shows that the conjugate gradient solver has a very good strong scaling up to 64 GPUs.

Xeon Phi results

The benchmark results for the XeonPhi benchmark suite are performed on Frioul, a test cluster at CINES, and the hybrid partition on MareNostrum III at BSC. Frioul has one KNL-card per node while the hybrid partition of MareNostrum III is equipped with two KNCs per node. The data on Frioul are generated by using the bash-scripts provided by the QCD-Accelerator Benchmarksute Part 2 and are done for the two test cases "Strong-Scaling" with a lattice size of 32x32x32x96 and "Weak-scaling" with a local lattice size of 48x48x48x24 per card. In case of the data generated at MareNostrum, data for the "Strong-Scaling" mode on a 32x32x32x96 lattice are shown. The Benchmark is using a random gauge configuration and uses the Conjugated Gradient solver to solve a linear equation involving the clover Wilson Dirac operator.



Figure 14 Result of second implementation of QCD on KNC

The figure shows strong scaling of the conjugate gradient solver on KNC's on the hybrid partition on MareNostrum III. The lattice size is given by 32x32x32x96, which corresponds to a moderate lattice size nowadays. The test is performed with a conjugate gradient solver in single precision by using the native mode and 60 openMP tasks per MPI processes. The run is done on one KNC per node (filled) and two KNCs node per node (non-filled).



Figure 15 Result of second implementation of QCD on KNL

The figure shows strong scaling of the conjugate gradient solver on KNL's on Frioul. The lattice size is given by 32x32x32x96 similar to the strong scaling run on the KNCs on MareNostrum III. The run is performed in quadrantic cache mode with 68 openMP processes per KNLs. The test is performed with a conjugate gradient solver in single precision.



Figure 16 Result of second implementation of QCD on KNL on a larger test case

The figure shows strong scaling of the conjugate gradient solver on KNL's GPUs on PizDaint. The lattice size is increased to 64x64x64x128, which is a commonly used large lattice nowadays. By increasing the lattice the scaling tests shows that the conjugate gradient solver has a very good strong scaling up to 64 GPUs.

### Quantum Espresso

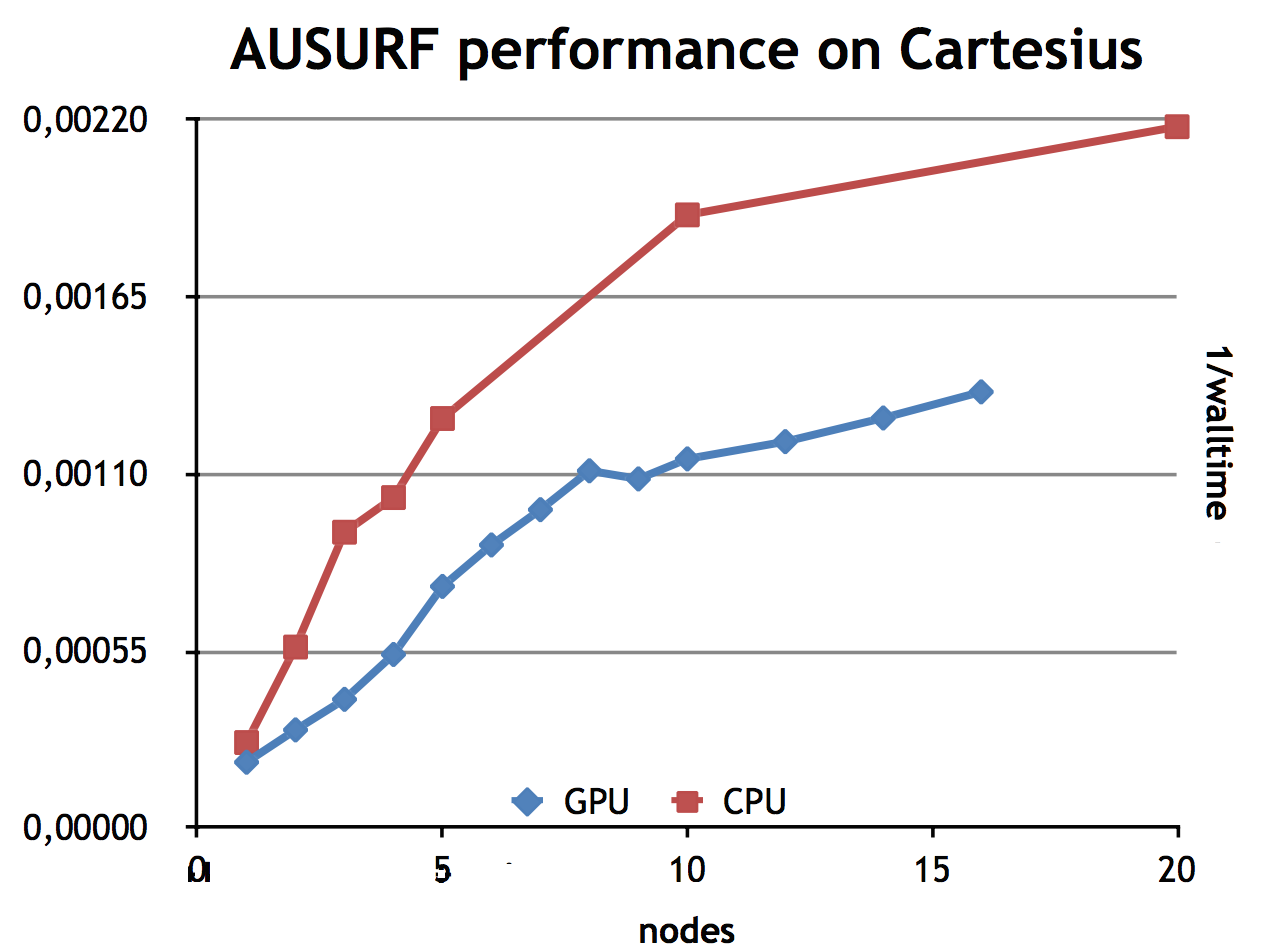


Figure 17 AUSURF performances on GPU

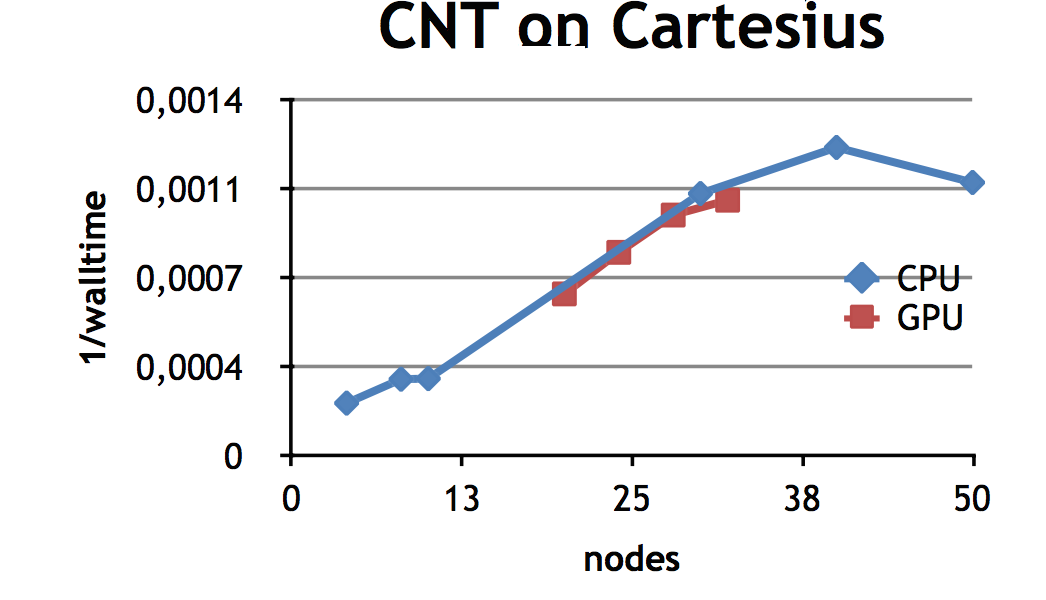


Figure 18 CNT performances on GPU

### Synthetic benchmarks (SHOC)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | NVIDIA GPU | | Intel Xeon Phi | | |
|  | K40 CUDA | K40 OpenCL | KNC Offload | KNC OpenCL | Haswell OpenCL |
| BusSpeedDownload | 10.55 GB/s | 10.56 GB/s | 6.6 GB/s | 6.8 GB/s | 11.3 GB/s |
| BusSpeedReadback | 10.56 GB/s | 10.56 GB/s | 6.7 GB/s | 6.8 GB/s | 10.4 GB/s |
| maxspflops | 3710 GF/s | 3658 GF/s | 21581 GF/s | 2314 GF/s | 1427 GF/s |
| maxdpflops | 1411 GF/s | 1411 GF/s | 16017 GF/s | 2318 GF/s | 742 GF/s |
| gmem\_readbw | 177 GB/s | 179 GB/s | 170 GB/s | 49.7 GB/s | 18.5 GB/s |
| gmem\_readbw\_strided | 18 GB/s | 20 GB/s | N/A | 35 GB/s | 156 GB/s |
| gmem\_writebw | 175 GB/s | 188 GB/s | 72 GB/s | 41 GB/s | 13.6 GB/s |
| gmem\_writebw\_strided | 7 GB/s | 7 GB/s | N/A | 25 GB/s | 163 GB/s |
| lmem\_readbw | 1164 GB/s | 1156 GB/s | N/A | 442 GB/s | 200 GB/s |
| lmem\_writebw | 1192 GB/s | 1162 GB/s | N/A | 477 GB/s | 248 GB/s |
| BFS | 51,816,100 Edges/s | 42,088,000 Edges/s | N/A | 1,635,330 Edges/s | 12,034,300 Edges/s |
| FFT\_sp | 523 GF/s | 146 GF/s | 135 GF/s | 71 GF/s | 70 GF/s |
| FFT\_dp | 261 GF/s | 61 GF/s | 69.5 GF/s | 31 GF/s | 50 GF/s |
| SGEMM | 2900-2990 GF/s | 694/761 GF/s | 640/645 GF/s | 179/217 GF/s | 348-503 GF/s |
| DGEMM | 1025-1083 GF/s | 411/433 GF/s | 179/190 GF/s | 76/100 GF/s | 174-175 GF/s |
| MD (SP) | 180 GF/s | 91 GF/s | 28 GF/s | 33 GF/s | 106 GF/s |
| MD5Hash | 3.42 GH/s | 3.36 GH/s | N/A | 1.7 GH/s | 1.12 GH/s |
| ReducWon | 137 GB/s | 150 GB/s | 99 GB/s | 10 GB/s | 51 GB/s |
| Scan | 47 GB/s | 39 GB/s | 11 GB/s | 4.5 GB/s | 15 GB/s |
| Sort | 3.08 GB/s | 0.54 GB/s | N/A | 0.11 GB/s | 0.3 GB/s |
| Spmv | 4-23 GF/s | 3-17 GF/s | 1-17944 GF/s | N/A | 1-8 GF/s |
| Stencil2D | 123 GF/s | 135 GF/s | 89 GF/s | 8.95 GF/s | 29 GF/s |
| Stencil2D\_dp | 57 GF/s | 67 GF/s | 16 GF/s | 7.92 GF/s | 26 GF/s |
| Triad | 13.5 GB/s | 9.9 GB/s | 5.76 GB/s | 5.57 GB/s | 8 GB/s |
| S3D (level2) | 93 GF/s | 91 GF/s | 109 GF/s | 18 GF/s | 25 GF/s |

Table 2 Synthetic benchmarks relusts on K40 and KNC

Measures marked red are not relevant and should not be considered.

### SPECFEM3D

So far it has only been possible to run on one fixed core count for each test case, so scaling curves are not available. Test case A ran on 4 KNL and 4 P100. Test case B ran on 10 KNL and 4 P100.

|  |  |  |
| --- | --- | --- |
|  | KNL | P100 |
| Test case A | 66 | 105 |
| Test case B | 21.4 | 68 |

Table 3 SPECFEM 3D GLOBE results

## Conclusion and future work

The work presented here stand as a first sight for application benchmarking on accelerators. Most codes have been selected among the main Unified European Application Benchmark Suite. This paper describes each of them as well as implementation, relevance to European science community and test cases. We have presented results on leading edge systems

The suite will be publicly available on the PRACE web site [1] where links to download sources and test cases will be published along with compilation and run instructions.

Task 7.2B in PRACE 4IP started to design a benchmark suite for accelerators. This work has been done aiming at integrating it to the main UEABS one so that both can be maintained and evolve together. As PCP (PRACE-3IP) machines will soon be available, it will be very interesting to run the benchmark suite on them. First because these machines will be larger, but also because they will feature energy consumption probes.