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

Final

Version: 1.0
Author(s): Victor Cameo Ponz, CINES
Date: 24.03.2017

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: 31 / 03 / 2017
	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: <1.0>	**Status:** ***Final***
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2010	
	File(s): D7.5	
Authorship	**Written by:**	Victor Cameo Ponz, CINES
	Contributors:	Adem Tekin, ITU Alan Grey, EPCC Andrew Emerson, CINECA Andrew Sunderland, STFC Arno Proeme, EPCC Charles Moulinec, STFC Dimitris Dellis, GRNET Fiona Reid, EPCC Gabriel Hautreux, INRIA Jacob Finkenrath, CyI James Clark, STFC Janko Strassburg, BSC Jorge Rodriguez, BSC Martti Louhivuori, CSC Philippe Segers, GENCI Valeriu Codreanu, SURFSARA
	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	First revision
0.2	15/03/2017	Draft	Include remark of the first review + new figures
1.0	24/03/2017	Final version	Improved the application performance section

Document Keywords

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

Table of Contents

List of Figures

List of Tables

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

[5] https://userinfo.surfsara.nl/systems/cartesius/description

[6] MareNostrum III User's Guide Barcelona Supercomputing Center – https://www.bsc.es/support/MareNostrum3-ug.pdf

[7] http://www.idris.fr/eng/ouessant/

[8] PFARM reference – https://hpcforge.org/plugins/mediawiki/wiki/pracewp8/images/3/34/Pfarm_long_lug.pdf

[9] Solvent-Driven Preferential Association of Lignin with Regions of Crystalline Cellulose in Molecular Dynamics Simulation – Benjamin Lindner et al. – Biomacromolecules, 2013

[10] NAMD website – http://www.ks.uiuc.edu/Research/namd/

[11] SHOC source repository – https://github.com/vetter/shoc

[12] Parallelizing the QUDA Library for Multi-GPU Calculations in Lattice Quantum Chromodynamics – R. Babbich, M. Clark and B. Joo – SC 10 (Supercomputing 2010)

[13] 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

[14] Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement – G. Houzeaux, R. Aubry, and M. Vázquez – Computers & Fluids, 44:297-313, 2011

[15] MIMD Lattice Computation (MILC) Collaboration – http://physics.indiana.edu/~sg/milc.html

[16] targetDP – https://ccpforge.cse.rl.ac.uk/svn/ludwig/trunk/targetDP/README

[17] QUDA: A library for QCD on GPU – https://lattice.github.io/quda/

[18] QPhiX, QCD for Intel Xeon Phi and Xeon processors – http://jeffersonlab.github.io/qphix/

[19] KNC MaxFlops issue (both SP and DP) – https://github.com/vetter/shoc/issues/37

[20] KNC SpMV issue – https://github.com/vetter/shoc/issues/24, https://github.com/vetter/shoc/issues/23.

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 (= 2^{30} ~ 10^9) Bytes (= 8 bits), also GByte
Gb/s	Giga (= 10^9) bits per second, also Gbit/s
GB/s	Giga (= 10^9) 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 (= 10^9) Floating point operations (usually in 64-bit, i.e. DP) per second, also GF/s
GHz	Giga (= 10^9) Hertz, frequency =10^9 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 (= 2^{10} ~10^3) Bytes (= 8 bits), also KByte
LINPACK	Software library for Linear Algebra
MB	Management Board (highest decision making body of the project)
MB	Mega (= 2^{20} ~ 10^6) Bytes (= 8 bits), also MByte
MB/s	Mega (= 10^6) Bytes (= 8 bits) per second, also MByte/s
MFlop/s	Mega (= 10^6) 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.

1 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 3 gives a description of each of the selected applications, together with the test case datasets while section 4 presents some sample results. Section 5 outlines further work on, and using, the suite.

2 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 4 machines, the benchmarks ran on.

2.1 Co-processor description

Scientific computing using co-processors has gained popularity in recent years. First the utility of GPU 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 CPU 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	possible	not possible	required	required
max number of thread/cuda cores	240	272	2880	3584

Table 1 Main co-processors specifications

2.2 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.

As for the future, the PRACE-3IP PCP is in its third and last phase and will be a good candidate to provide access to bigger machines. The following suppliers had been awarded with a contract: ATOS/Bull SAS (France), E4 Computer Engineering (Italy) and Maxeler Technologies (UK), providing pilots using Xeon Phi, OPENPower and FPGA technologies. During this final phase, which started in October 2016, the contractors will have to deploy pilot system with a compute capability of around 1 PFlop/s, to demonstrate technology readiness of the proposed solution and the progress in terms of energy efficiency, using high frequency monitoring designed for this purpose. These results will be evaluated on a subset of applications from UEABS (NEMO, SPECFEM3D, QuantumEspresso, BQCD). The access to these systems is foreseen to be open to PRACE partners, with a special interest for the 4IP-WP7 task on accelerated Benchmarks.

2.2.1 *Cartesius K40*

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) CPU/node
  - 2x NVIDIA Tesla K40m GPU/node

  - 96 GB/node, DDR3-1600 RAM

- Total theoretical peak performance (Ivy Bridge + K40m) 1,056 cores + 132 GPU: 210 TF/s

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

2.2.2 *MareNostrum KNC*

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 containing:
  - 1x Sandy-Bridge-EP (2 x 8 cores) host processors E5-2670
  - 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.

2.2.3 *Ouessant P100*

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 GPU, 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 InfiniBand CAPI interconnect network (1 interconnect per node)

2.2.4 *Frioul KNL*

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

- Compute nodes
  - 7250 KNL, 68 cores, 4 threads per cores
  - 192GB of DDR4 memory
  - 16GB of MCDRAM
- Interconnect:
  - A Mellanox EDR 4x InfiniBand

3 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 codes presented in this suite are included in the latter. Exceptions are PFARM which comes from PRACE-2IP [8] and SHOC [11] a synthetic benchmark suite.

	OpenMP	OpenCL	CUDA
Alya			
Code_Saturne			
CP2K			
GPAW			
GROMACS			
NAMD			
PFARM			
QCD			
Quantum Espresso			
SHOC			
SPECFEM3D			

Table 2 Codes and corresponding APIs available (in green)

Table 2 lists the codes that will be presented in the next sections as well as their implementations available. It should be noted that OpenMP can be used with the Intel Xeon Phi architecture while CUDA is used for NVidia GPU cards. OpenCL has been considered as a third alternative that can be used on both architectures. It has been available on the first generation of Xeon Phi (KNC) but has not been ported to the second one (KNL). SHOC is the only code that is impacted, this problem is addressed in section 4.10.

3.1 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.

3.1.1 *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(1) [14] 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 whether the momentum preserving or the continuity preserving algorithm is selected.

3.1.2 *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 GPU.

3.2 Code_Saturne

Code_Saturne is a 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 multi-physics problems (fluid-structure, fluid-conjugate heat transfer, for instance).

3.2.1 *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 handled by MPI. Hybrid parallelism using MPI/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. The matrices are very sparse. PETSc has 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 is used in this benchmark suite.

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

3.2.2 *Test cases description*

Two test cases are dealt with, the former with a mesh made of hexahedral cells and the latter with a mesh made of tetrahedral cells. Both configurations are meant for incompressible laminar flows. The first test case is run on KNL in order to test the performance of the code always completely filling up a node using 64 MPI tasks and then either 1, 2, 4 OpenMP threads, or 1, 2, 4 extra MPI tasks to investigate the effect of hyper-threading. In this case, the pressure is computed using the code's native Algebraic Multigrid (AMG) algorithm as a solver. The second test case is run on KNL and GPU. In this configuration, the pressure equation is solved using the conjugate gradient (CG) algorithm from the PETSc library (the version of PETSc is the developer's version which supports GPU) and tests are run on KNL as well as on CPU+GPU. PETSc is built with the CUSP library and the CUSP format is used.

Note that computing the pressure using a CG algorithm has always been slower than using the native AMG algorithm, when using Code_Saturne. The second test is then meant to compare the current results obtained on KNL and GPU using CG only, and not to compare CG and AMG time to solution.

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 only. The Reynolds number is set to 100, 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 and is used to get some first comparisons using Code_Saturne linked to the developer's PETSc library, in order to get use of the GPU.

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. The mesh size is 2563 cells.

3.3 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 density functional theory (DFT) using the mixed Gaussian and plane waves approaches (GPW) and Gaussian and Augmented Plane (GAPW). Supported theory levels include DFTB, LDA, GGA, MP2, RPA, semi-empirical methods (AM1, PM3, PM6, RM1, MNDO, …), and classical force fields (AMBER, CHARMM, …).

3.3.1 *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 GPU and through OpenMP for MIC (Intel Xeon Phi).

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

3.3.2 *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 $Å^3$ (Angstroms cubed) cell. The calculation is performed using a DFT algorithm with 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 $Å^3$ 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 cutoff. 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 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.

3.4 GPAW

GPAW is a 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.

3.4.1 *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 GPU and MICs include support for offloading to accelerators using either CUDA or pyMIC, respectively. GPAW is freely available under the GPL license.

3.4.2 *Test cases description*

Carbon Nanotube

This test case is a ground state calculation for a carbon nanotube in vacuum. By default, it 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

This test case is a ground state calculation for a copper filament in vacuum. By default, it 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).

3.5 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 and up, 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 both single and double precision, and support all different kinds of SIMD support found in x86-family (and other) processors.

3.5.1 *Code description*

Parallelisation is achieved using combined OpenMP and MPI.

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

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

3.5.2 *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.

3.6 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.

3.6.1 *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 GPU and MIC (Intel Xeon Phi).

3.6.2 *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.

STMV.28M

This is a 3x3x3 replication of the original STMV dataset from the official NAMD site. The system contains roughly 28 million atoms. This data set also scales efficiently up to 6000 x86 Ivy Bridge cores.

3.7 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.

3.7.1 *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 GPU accelerators. EXDIG uses combined MPI and OpenMP to distribute the scattering energy calculations on CPU efficiently both across and within Intel Xeon Phi co-processors.

3.7.2 *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.

Iron, 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 energy region' up to 30 Legendre functions may be applied leading to Hamiltonian matrices of up to 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 are used in the calculation.

Methane, CH4

The dataset is an electron-molecule calculation with 1361 channels. Hamiltonian dimensions are therefore estimated between 13610 and ~40000. A process in the code which splits the constituent channels according to spin can be used to approximately halve the Hamiltonian size (whilst doubling the overall number of Hamiltonian matrices). As eigensolvers generally require O(N3) operations, spin splitting leads to a saving in both memory requirements and operation count. The final radius of the external region required is relatively long, leading to more numerous sectors calculations (estimated to between 20 and 30). The calculation will require many thousands of scattering energies.

In the current model, parallelism in EXDIG is limited to the number of sector calculations, i.e a maximum of around 30 accelerator nodes.

Methane is a relatively new dataset which has not been calculated on novel technology platforms at the very large-scale to date, so this is somewhat a step into the unknown. We are also somewhat reliant on collaborative partners that are not associated with PRACE for continuing to develop and fine tune the accelerator-based EXAS program for this proposed work. Access to suitable hardware with throughput suited to development cycles is also a necessity if suitable progress is to be ensured.

3.8 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.

3.8.1 *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 [15].

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 CPU.

The implementation for accelerators has been achieved using the "targetDP" programming model [16], a lightweight abstraction layer designed to allow the same application source code to be able to target multiple architectures, e.g. NVIDIA GPU and multicore/manycore CPU, in a performance portable manner. The targetDP syntax maps, at compile time, to either NVIDIA CUDA (for execution on GPU) or OpenMP+vectorisation (for implementation on

multi/manycore CPU 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 optimize for running on NVIDIA GPU [17]. The QPhix library consists of routines which are optimize to use INTEL intrinsic functions of multiple vector length, including optimized routines for KNC and KNL's [18]. In both QUDA and QPhix, the benchmark kernel uses the conjugate gradient solvers implemented within the libraries.

3.8.2 *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. Test cases for the second implementation are given by a strong-scaling mode with a lattice size of 32x32x32x96 and 64x64x64x128 and a weak scaling mode with a local lattice size of 48x48x48x24.

3.9 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.

3.9.1 *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.

3.9.2 *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 GPU (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.

3.10 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 GPU 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).

The SHOC benchmark suite has been selected to evaluate the performance of accelerators on synthetic benchmarks, mostly because SHOC provides CUDA/OpenCL/Offload/OpenACC variants of the benchmarks. This allowed us to evaluate NVIDIA GPU (with CUDA/OpenCL/OpenACC), Intel Xeon Phi KNC (with both Offload and OpenCL), but also Intel host CPU (with OpenCL/OpenACC). However, on the latest Xeon Phi processor (codenamed KNL) none of these 4 models is supported. Thus, benchmarks on the KNL architecture can not be run at this point, and there aren't any news of Intel supporting OpenCL on the KNL. However, there is work in progress on the PGI compiler to support the KNL as a target. This support will be added during 2017. This will allow us to compile and run the OpenACC benchmarks for the KNL. Alternatively, the OpenACC benchmarks will be ported to OpenMP and executed on the KNL.

3.10.1 *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 GPU and through OpenMP for MIC (Intel Xeon Phi). For selected benchmarks OpenACC implementations are provided for GPU. Multi-node parallelisation is achieved using MPI.

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

3.10.2 *Test cases description*

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

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

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

3.11 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.

3.11.1 *Test cases definition*

Both test cases will use the same input data. A 3D shear-wave speed model (S362ANI) will be used to benchmark the code.

Here is an explanation of the simulation parameters that will be used to size the test case:

- *NCHUNKS,* number of face of the cubed sphere included in the simulation (will be always 6)
- *NPROC_XI,* number of slice along one chunk of the cubed sphere (will represents also the number of processors used for 1 chunk
- *NEX_XI*, number of spectral elements along one side of a chunk.
- *RECORD_LENGHT_IN_MINUTES,* length of the simulated seismograms. The time of the simulation should vary linearly with this parameter.

Small test case

It runs with 24 MPI tasks and has the following mesh characteristics:

- NCHUCKS=6
- NPROC_XI=2
- NEX_XI =80
- RECORD_LENGHT_IN_MINUTES =2.0

Bigger test case

It runs with 150 MPI tasks and has the following mesh characteristics:

- NCHUCKS=6
- NPROC_XI=5
- NEX_XI =80
- RECORD_LENGHT_IN_MINUTES =2.0

4 Applications performances

This section presents some sample results on targeted machines.

4.1 Alya

Alya has been compiled and run using test case A on three different types of compute nodes:

- BSC MinoTauro Westemere Partition (Intel E5649 12 core 2.53 GHz, 24 GB RAM, Infiniband)
- BSC MinoTauro Haswell + K80 Partition (Intel Xeon E5-2630 v3 16 core 2.4 GHz, 128 GB RAM, NVIDIA K80, Infiniband)
- KNL 7250 (68 core 1.40 GHz, 16 GB MCDRAM, 96BG DDR4 RAM, Ethernet)

Alya supports parallelism via different options, mainly MPI for problem decomposition, OpenMP within the matrix construction phase and CUDA parallelism for selected solvers. In general, the best distribution and performance can be achieved by using MPI. Running on KNL it has been proven optimal to use 4 OpenMP threads and 16 MPI processes for a total of 64 processes, each on its own physical core. The Xeon Phi processor shows slightly better performance in Alya configured in Quadrant/Cache when compared to Quadrant/Flat, although the difference is negligible. The application is not optimized for the first generation Xeon Phi KNC and does not support offloading.

Overall speedups have been compared to a one node CPU run on the Haswell partition of MinoTauro. As the application is heavily optimized for traditional computation the best and almost linear scaling is observed on the CPU only runs. Some calculations benefit from the accelerators, GPU yielding from 3.6x to 6.5x speedup for one to three nodes. The KNL runs are limited by the OpenMP scalability and too many MPI tasks on these processors lead to suboptimal scaling. Speedups in this case range from 0.9x to 1.6x and can be further optimized by introducing more threading parallelism. The communication overhead when running with many MPI tasks on KNL is noticeable and further limited by the ethernet connection on multinode runs. High-performance fabrics such as Omni-Path or Infiniband promise to provide significant enhancement for these cases. The results are compared in Figure 3.

It can be seen that the best performance is gained on the most recent standard Xeon CPU in conjunction with GPU. This is expected as Alya has been heavily optimized for traditional HPC scalability using mainly MPI and makes good use of available cores. The addition of GPU enabled solvers provides a noticeable boost to the overall performance. To fully exploit the KNL further optimizations are ongoing and additional OpenMP parallelism will need to be employed.

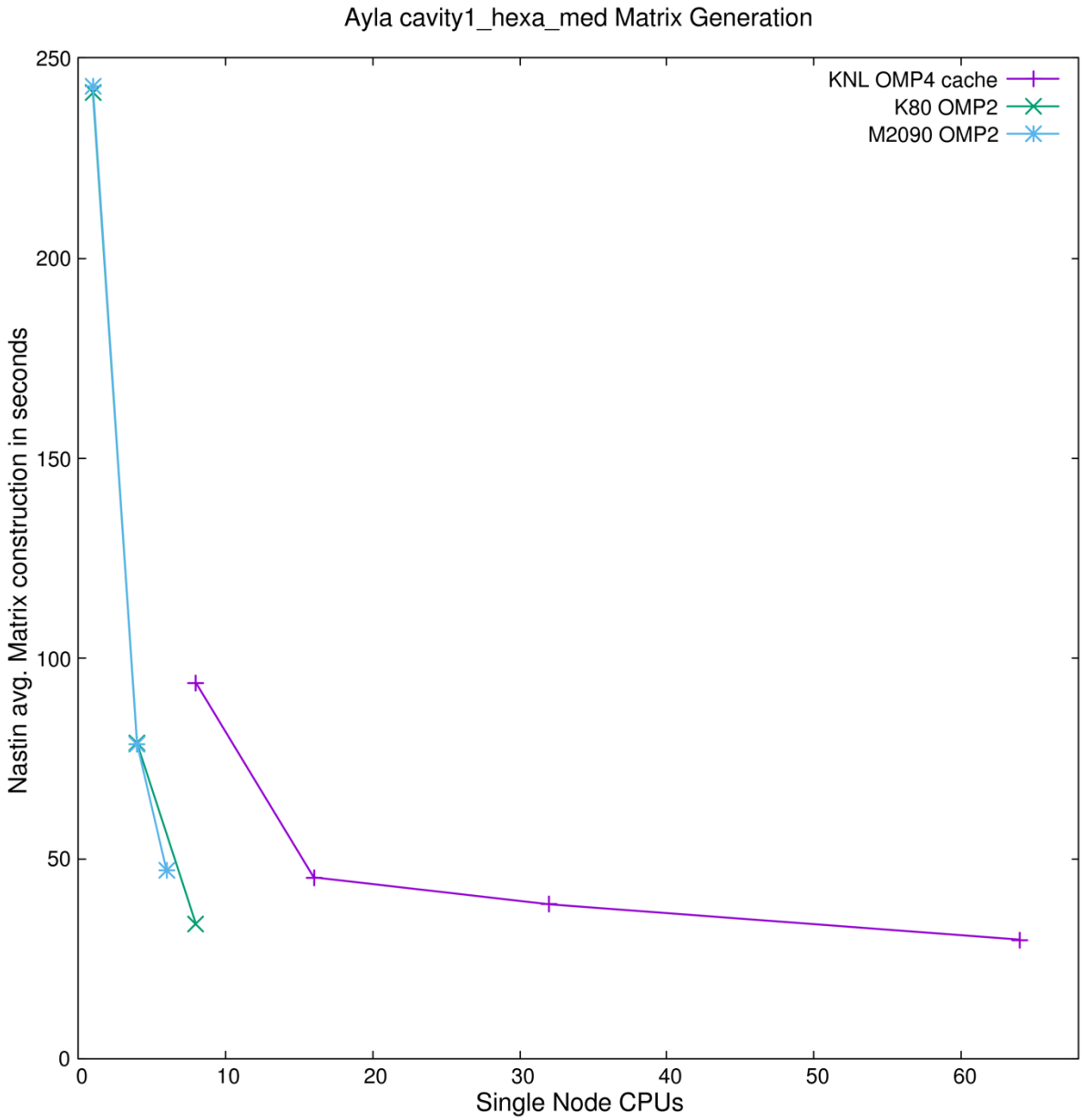

Figure 1 Shows the matrix construction part of Alya that is parallelised with OpenMP and benefits significantly from the many cores available on KNL.

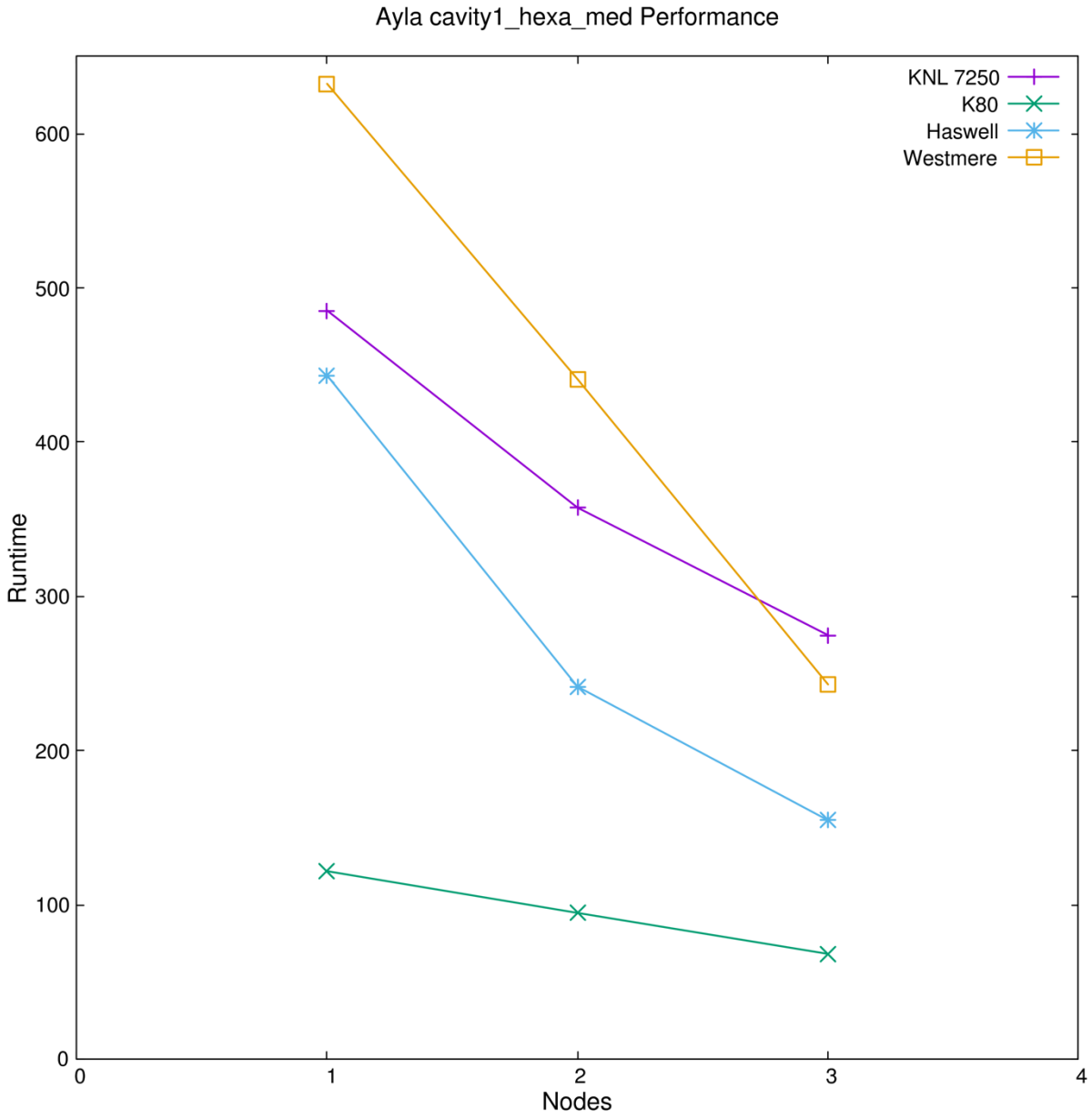

Figure 2 Demonstrates the scalability of the code. As expected Haswell cores with K80 GPU are high-performing while the KNL port is currently being optimized further.

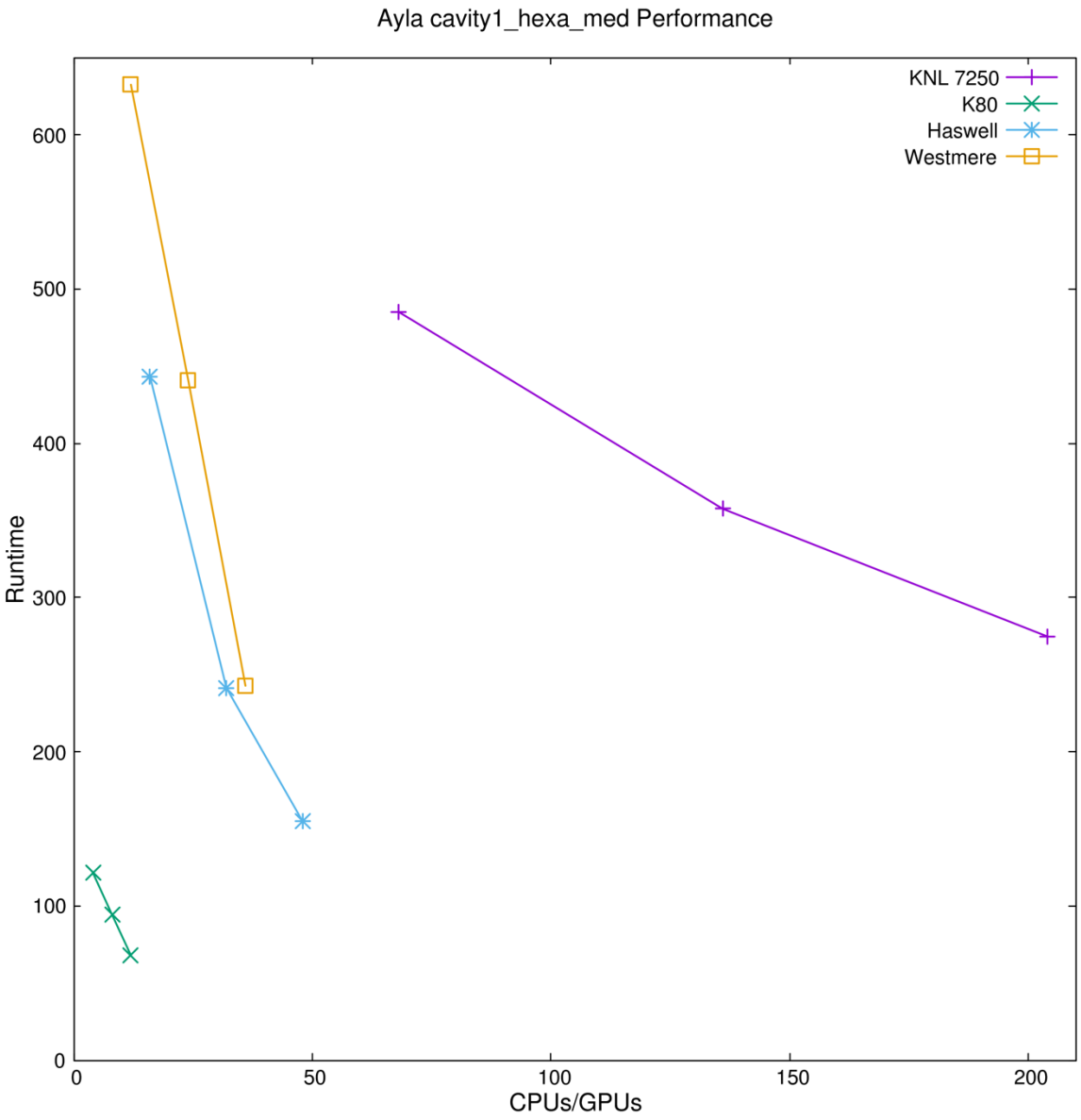

Figure 3 Best performance is achieved with GPU in combination with powerful CPU cores. Single thread performance has a big impact on the speedup, both threading and vectorization are employed for additional performance.

4.2 Code_Saturne

Description runtime architecture:

- KNL: ARCHER (model 7210) - The following environment is used, i.e. ENV_6.0.3. The INTEL compiler's version is 17.0.0.098.
- GPU: 2 POWER8 nodes, i.e. S822LC (2x P8 10-cores + 2x K80 (2 G210 per K80)) and S824L (2x P8 12-cores + 2x K40 (1 G180 per K40)) - The compiler is at/8.0, the MPI distribution openmpi/1.8.8 and the CUDA compiler's version is 7.5.

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

The first test case has been run on ARCHER KNL and the performance has been investigated for several configurations, each of them using 64 MPI tasks per node and either 1, 2 or 4 hyper-threads (extra MPI tasks) or OpenMP threads have been added for testing. The results are

compared to ARCHER CPU, in this case IvyBridge CPU. Up to 8 nodes are used for comparison.

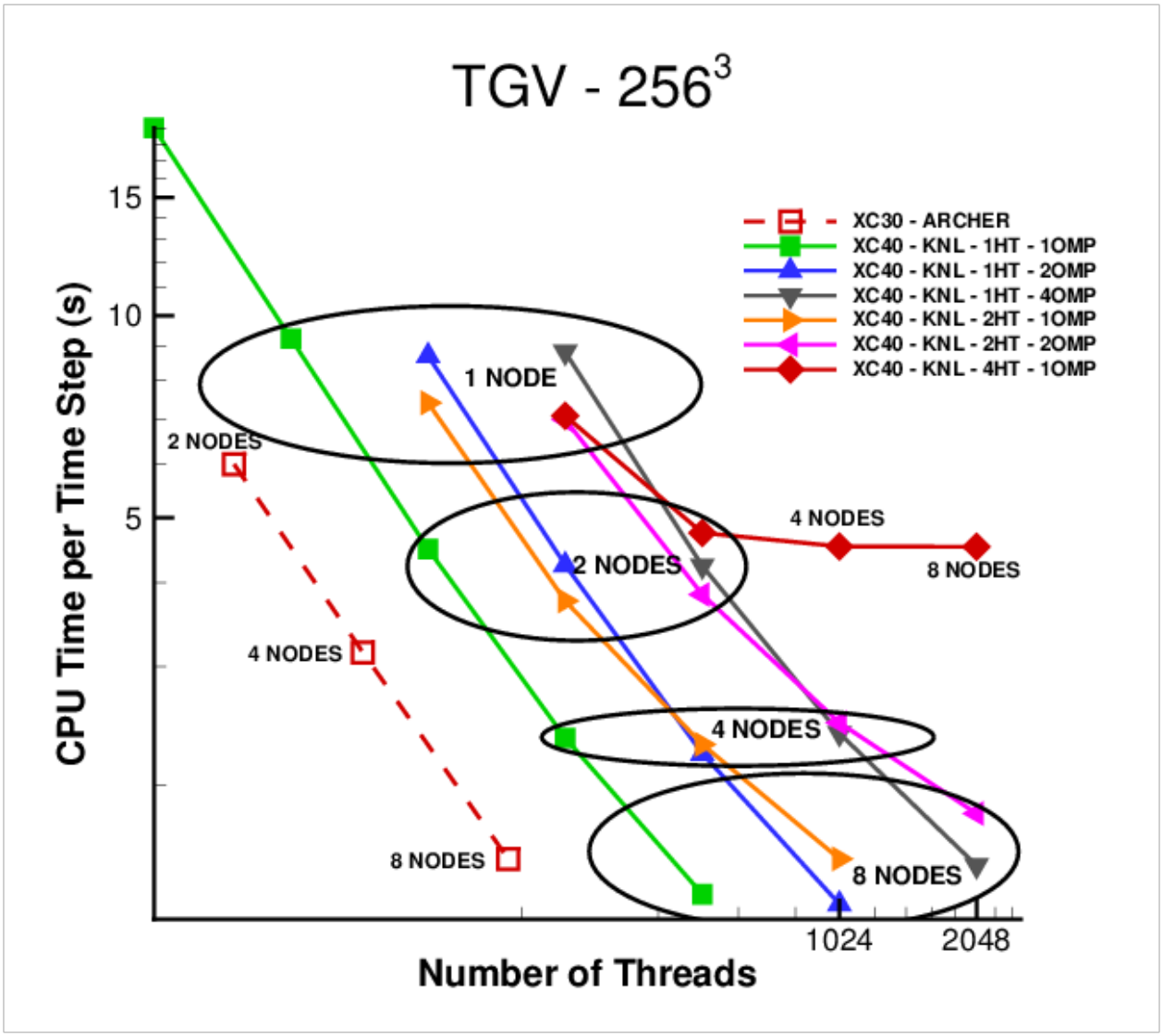

Figure 4 Code_Saturne's performance on KNL. AMG is used as a solver in V4.2.2.

Figure 4 shows the CPU time per time step as a function of the number threads/MPI tasks. For all the cases, the time to solution decreases when the number of threads increases. For the case using MPI only and no hyper-threading (green line) only, a simulation is run on half a node as well to investigate the speedup going from half a node to a node, which is about 2 as seen on the figure. The ellipses help comparing the time to solution per node, and finally, a comparison is carried out with simulations run on ARCHER without KNL, using Ivybridge processors. When using 8 nodes, the best configuration for Code_Saturne to run on KNL is for 64 MPI tasks and 2 OpenMP threads per task (blue line on the figure), which is about 15 to 20% faster than running on the Ivybridge nodes, using the same number of nodes.

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

The following options are used for PETSc: -

- -CPU: -ksp_type = cg and -pc_type = jacobi

- -GPU: -ksp_type = cg and -vec_type = cusp and -mat_type = aijcusp and -pc_type = jacobi

1 node IBM POWER 8 (S822LC) with 2x K80s				1 node IBM POWER 8 (S824L) with 2x K40s			
	CPU	CPU/GPU	CPU vs CPU/GPU		CPU	CPUGPU	CPU vs CPU/GPU
#MPI tasks	T (s)	T (s)	speedup	#MPI tasks	T (s)	T (s)	speedup
1	1022.18	630.54	1.62	1	1087.75	637.86	1.71
2	621.20	337.12	1.84	2	659.71	327.18	2.01
4	263.61	173.95	1.51	4	267.82	189.04	1.42
20	76.38	109.75	0.70	24	57.81	112.82	0.51

Table 3 Performance of Code_Saturne + PETSc on 1 node of the POWER8 clusters. Comparison between 2 different nodes, using different types of CPU and GPU. PETSc is built on LAPACK. The speedup is computed at the ratio between the time to solution on the CPU for a given number of MPI tasks and the time to solution on the CPU/GPU for the same number of MPI tasks.

1 node Intel Xeon Phi - Knights Landing			
All the simulations are run using 64 MPI tasks per node			
#Threads	T (s)	#MPI tasks	T (s)
1	95.12	64	95.12
2	127.37	128 (2 hyperthreads)	78.93

Table 4 Performance of Code_Saturne and PETSc on 1 node of KNL. PETSc is built on the MKL library

Table 3 and Table 4 show the results obtained using POWER8 CPU and CPU/GPU, and KNL, respectively. Focusing on the results on the POWER8 nodes first, a speedup is observed on each node of the POWER8, when using the same number of MPI tasks and of GPU. However, when the nodes are fully populated (20 and 24 MPI tasks, respectively), it is cheaper to run on the CPU only than using CPU/GPU. This could be explained by the fact that the same overall amount of data is transferred but the system administration costs, latency costs, asynchronicity of transfer in 20 (S822LC) or 24 (S824L) slices might be prohibitive.

4.3 CP2K

Times shown in the ARCHER KNL (model 7210, 1.30GHz, 96GB memory DDR) vs Ivy Bridge (E5-2697 v2, 2.7 GHz, 64GB) plot are for those CP2K threading configurations that give the best performance in each case. The shorthand for naming threading configurations is:

- MPI: pure MPI
- X_TH: X OpenMP threads per MPI rank

Whilst single-threaded pure MPI or 2 OpenMP threads is often fastest on conventional processors, on the KNL multithreading is more likely to be beneficial, especially in problems such as the LiH-HFX benchmark in which having fewer MPI ranks means more memory is available to each rank, allowing partial results to be stored in memory instead of expensively recomputed on the fly.

Hyperthreads were left disabled (equivalent to the aprun option –j 1), as no significant performance benefit was observed using hyperthreading.

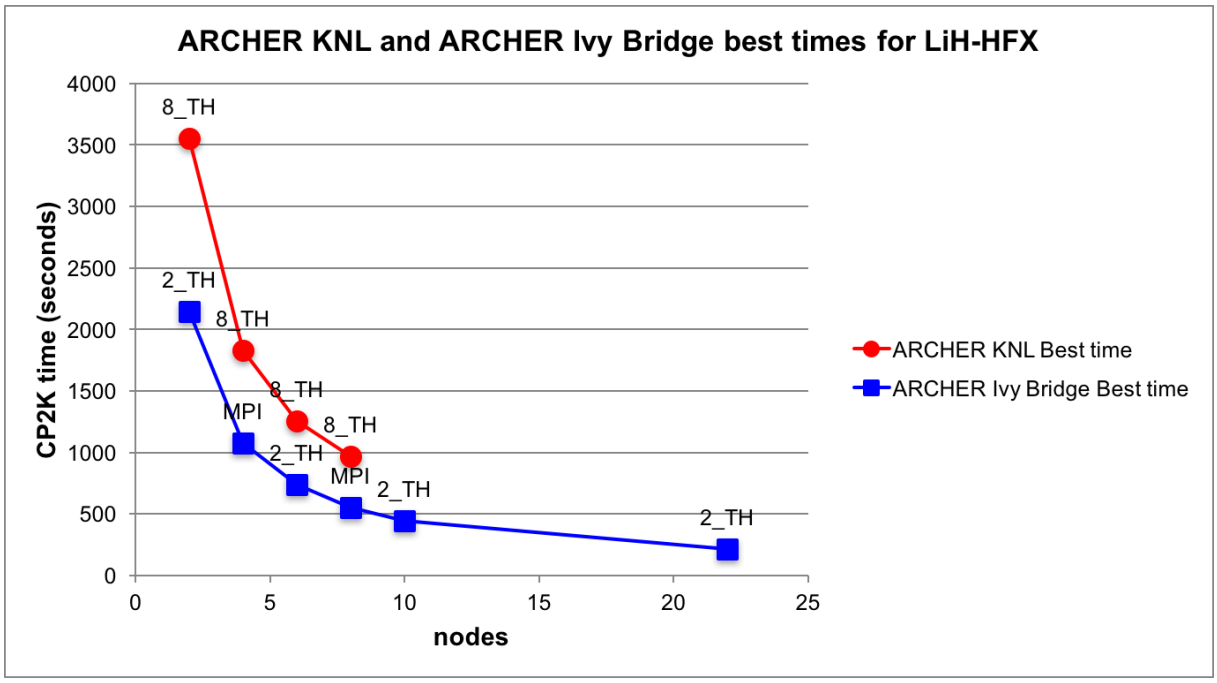

Figure 5 Test case 1 of CP2K on the ARCHER cluster

The node based comparison shows (Figure 5) that the runtimes on KNL nodes are roughly 1.7 times slower than runtimes on 2-socket IvyBridge nodes.

4.4 GPAW

The performance of GPAW using both benchmarks was measured with a range of parallel job sizes on several architectures; with the architectures designated in the following tables, figures, and text as:

- CPU: x86 Haswell CPU (Intel Xeon E5-2690v3) in a dual-socket node
- KNC: Knights Corner MIC (Intel Xeon Phi 7120P) with a x86 Haswell host CPU (Intel Xeon E5-2680v3) in a dual-socket node
- KNL: Knights Landing MIC (Intel Xeon Phi 7210) in a single-socket node
- K40: K40 GPU (NVIDIA Tesla K40) with a x86 Ivy Bridge host CPU (Intel Xeon E5-2620-v2) in a dual-socket node
- K80: K80 GPU (NVIDIA Tesla K80) with a x86 Haswell host CPU (Intel Xeon E5-2680v3) in a quad-socket node

Only time spent in the main SCF-cycle was used as the runtime in the comparison (Table 5 and Table 6) to exclude any differences in the initialisation overheads.

n	CPU	KNC	KNL	K40	K80
1	517.8		319.9	237.6	249.4
2	253.9	281.2	206.6		
4	136.2	164.9	141.3		
8	80.7	102.9	101.3		
16	55.6	77.0			
32	41.7	63.8			

Table 5 GPAW runtimes (in seconds) for the smaller benchmark (Carbon Nanotube) measured on several architectures when using n sockets (i.e. processors or accelerators).

n	CPU	KNC	KNL	K40	K80
1	802.2		323.4	386.8*	344.4*
2	405.8		172.3		
4	195.2		127.0		
8	95.4	102.6	80.0		
16	60.3	69.4			
32	33.7	42.8			
64	19.3	31.7			

Table 6 GPAW runtimes (in seconds) for the larger benchmark (Copper Filament) measured on several architectures when using n sockets (i.e. processors or accelerators). *Due to memory limitations on the GPU the grid spacing was increased from 0.22 to 0.28 to have a sparser grid. To account for this in the comparison, the K40 and K80 runtimes have been scaled up using a corresponding CPU runtime as a yardstick (scaling factor q=2.1132).

As can been seen from Table 2 and Table 3, in both benchmarks a single KNL or K40/K80 was faster than a single CPU. But when using multiple KNL, the performance does not seem to scale as well as for CPU. In the smaller benchmark (Carbon Nanotube), CPU outperform KNL when using more than 2 processors. In the larger benchmark (Copper Filament), KNL still outperform CPU with 8 processors but it seems likely that the CPU will overtake KNL when using an even larger number of processors.

In contrast to KNL, the older KNC are slower than Haswell CPU across the board. Nevertheless, as can been seen from Figure 4, the scaling of KNC is to some extend comparable to CPU but with a lower scaling limit. It is therefore likely that, on systems with considerably slower host CPU than Haswells (e.g. Ivy Bridges), KNC may also give a performance boost over the host CPU.

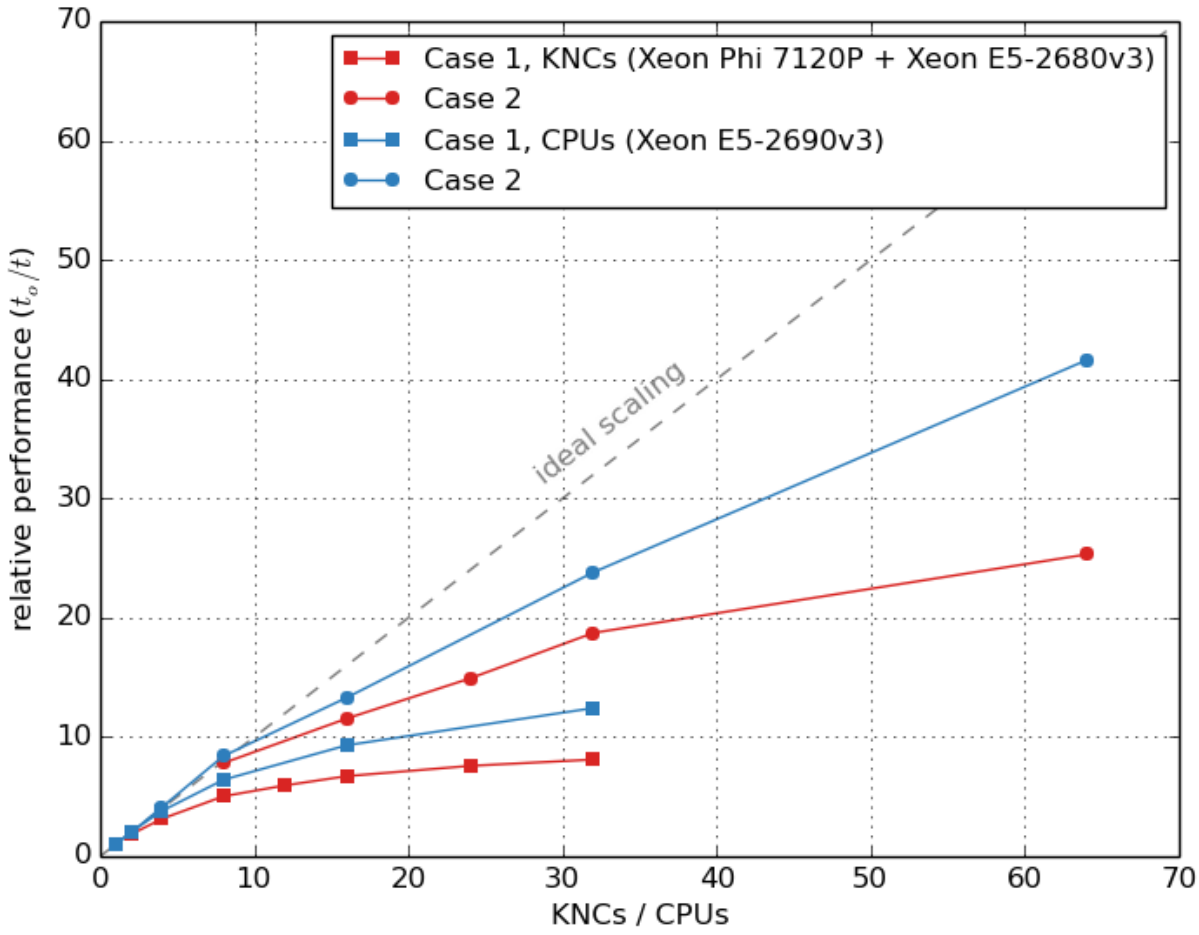

Figure 6 Relative performance (to / t) of GPAW is shown for parallel jobs using an increasing number of CPU (blue) or Xeon Phi KNC (red). Single CPU SCF-cycle runtime (to) was used as the baseline for the normalisation. Ideal scaling is shown as a linear dashed line for comparison. Case 1 (Carbon Nanotube) is shown with square markers and Case 2 (Copper Filament) is shown with round markers.

4.5 GROMACS

Gromacs was successfully compiled and ran on the following systems:

- GRNET ARIS: Thin nodes (E5-2680v2), GPU nodes (Dual E5-2660v3+ Dual K40m), all with FDR14 Infiniband, Single node KNL 7210.
- CINES Frioul KNL 7230
- IDRIS Ouessant: IBM Power 8 + Dual P100

On KNL machines the runs were performed using Quadrant processor and both Cache / Flat memory configuration. On GRNET's single node KNL more configurations were tested.

As it is expected the Quandrant/Cache mode gives the best performance for all cases. The performance dependence on the MPI Tasks/OpenMP threads combination was also explored. In most cases 66 tasks/per node using 2 or 4 threads/task gives the best performance on KNL 7230.

In all accelerated runs a speed up of 2-2.6x with respect CPU only was achieved with GPU. Gromacs does not support offload on KNC.

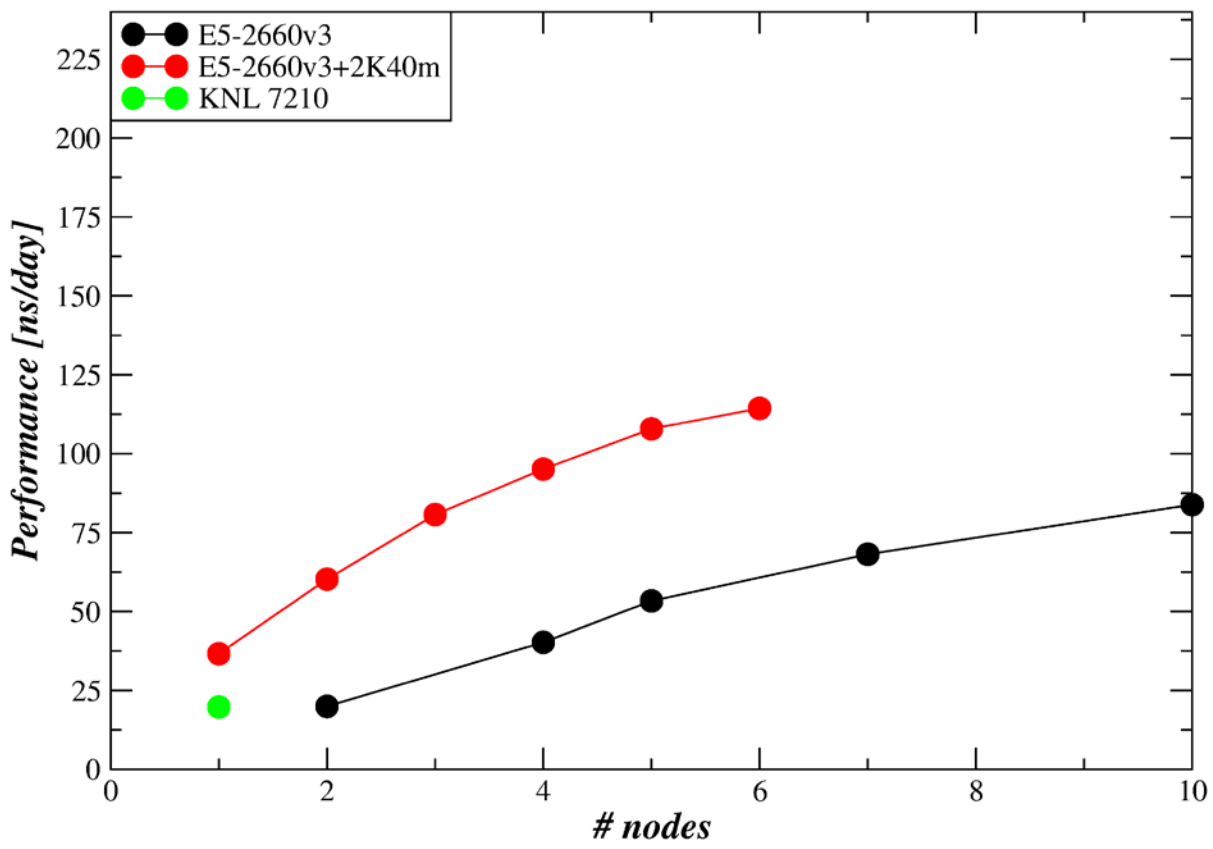

Figure 7 Scalability for GROMACS test case GluCL Ion Channel

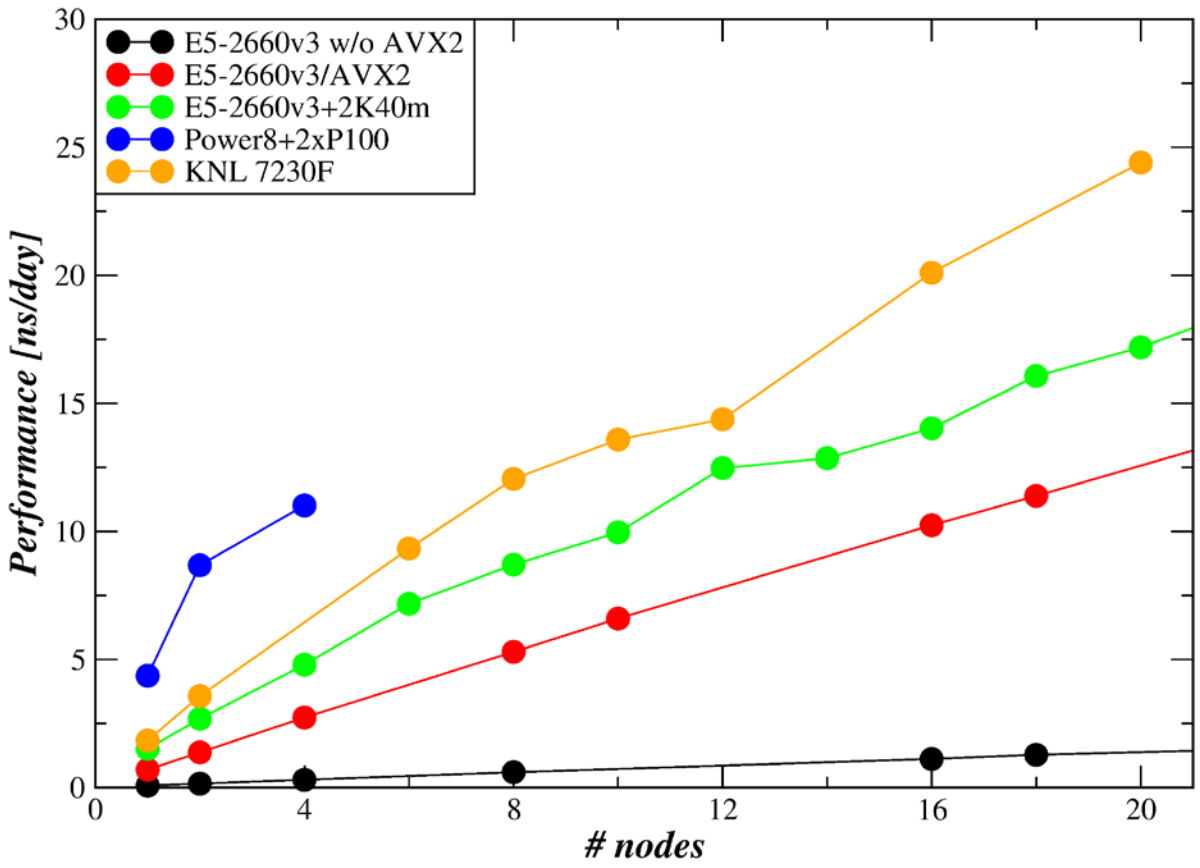

Figure 8 Scalability for GROMACS test case Lignocellulose

4.6 NAMD

NAMD was successfully compiled and ran on the following systems:

- GRNET ARIS : Thin nodes (E5-2680v2), GPU nodes (Dual E5-2660v3+ Dual K40m), KNC Nodes (Dual E5-2660v2+Dual KNC 7120P), all with FDR14 Infiniband, Single node KNL 7210.
- Cines Frioul : KNL 7230
- Cines Ouessant : IBM Power 8 + Dual P100

On KNL machines the runs were performed using Quadrant processor and both Cache / Flat memory configuration. On GRNET's single node KNL more configurations were tested.

As it is expected the Quandrant/Cache mode gives the best performance for all cases. The performance dependence on the MPI Tasks/OpenMP threads combination was also explored.

In most cases 66 tasks per node using 4 threads/task or 4 tasks per node/64 threads per task gives the best performance on KNL 7230.

In all accelerated runs a speed up of 5-6x with respect CPU only runs was achieved with GPU.

On KNC the speed up with respect CPU only is in the range 2-3.5 in all cases.

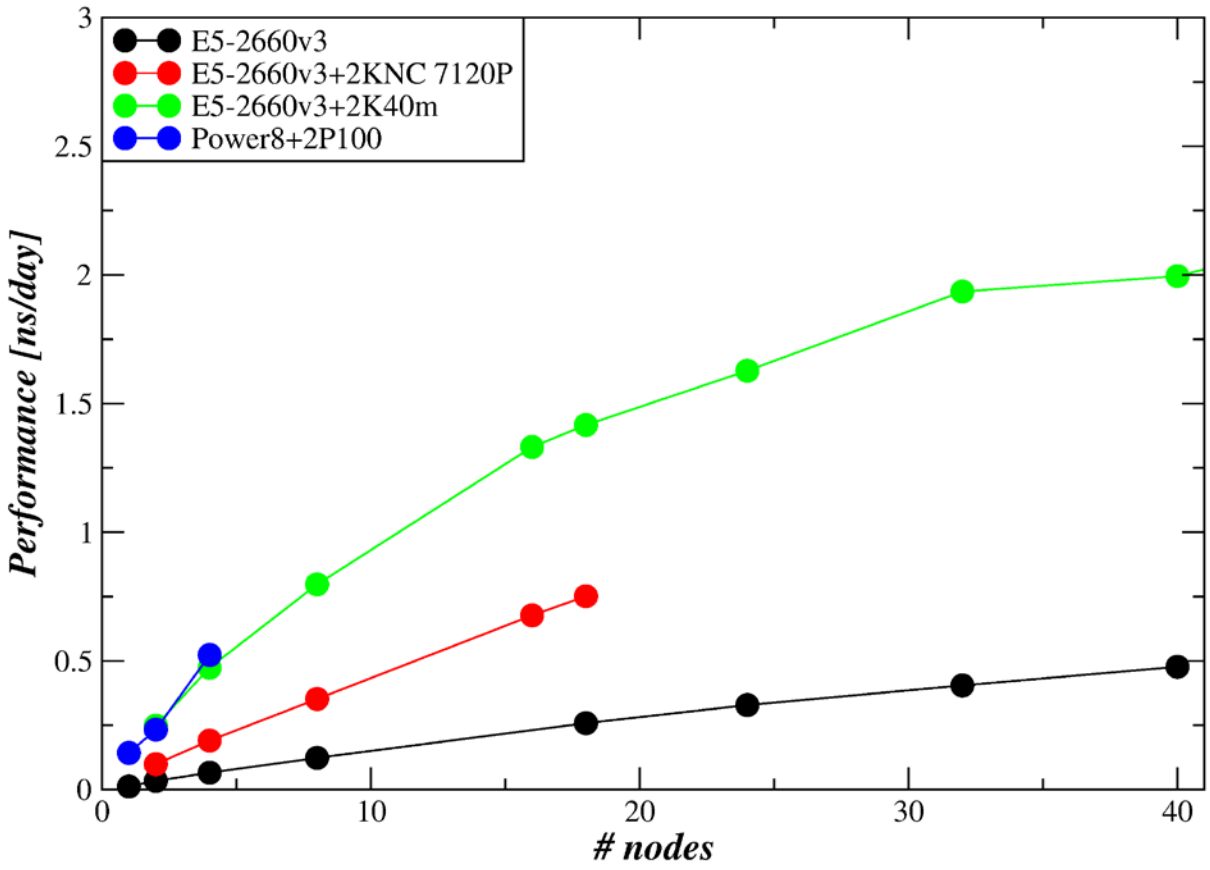

Figure 9 Scalability for NAMD test case STMV.8M

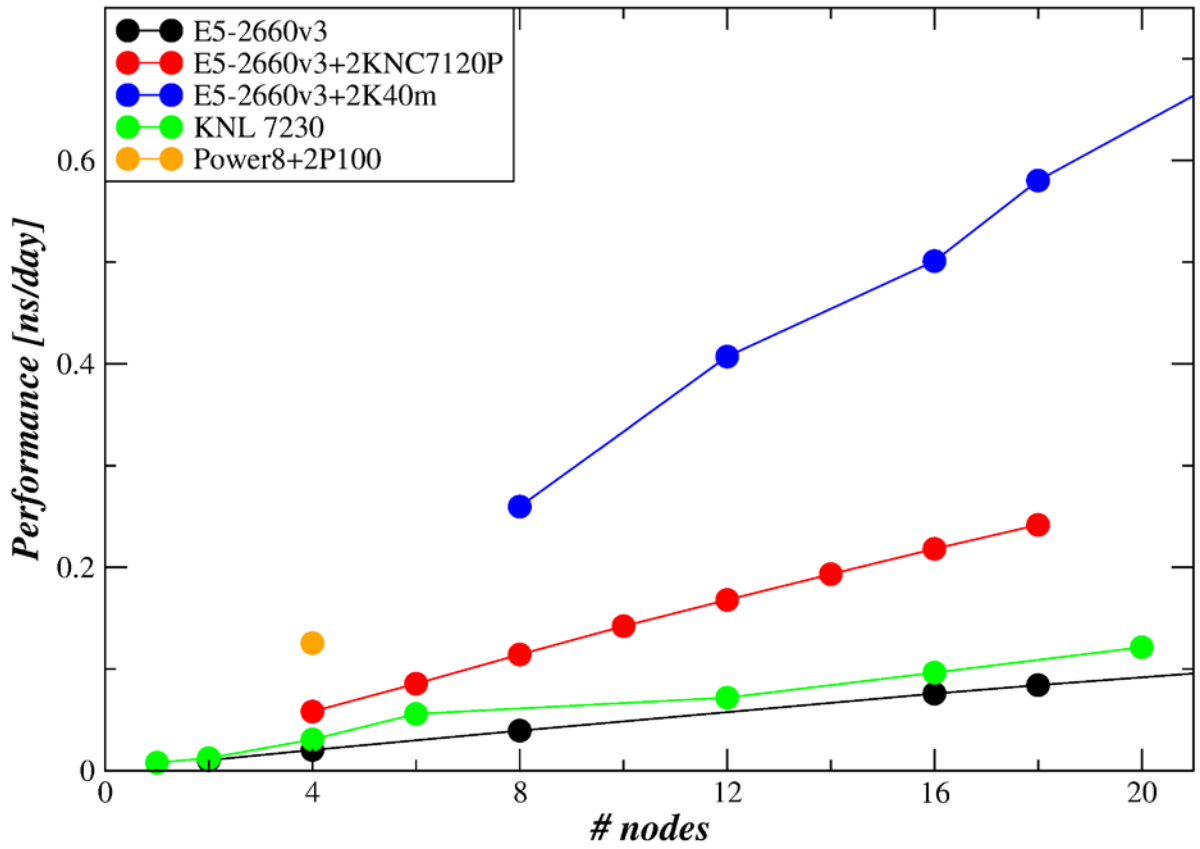

Figure 10 Scalability for NAMD test case STMV.28M

4.7 PFARM

The code has been tested and timed on several architectures, designated in the following figures, tables and text as:

- CPU: node contains two 2.7 GHz, 12-core E5-2697 v2 (Ivy Bridge) series processors with 64GB memory.
- KNL: node is a 64-core KNL processor (model 7210) running at 1.30GHz with 96GB of memory.
- GPU: node contains a dual socket 16-core Haswell E5-2698 running at 2.3 GHz with 256GB memory and 4 K40, 4 K80 or 4 P100 GPU.

Codes on all architectures are compiled with the Intel compiler (CPU v15, KNL & GPU v17).

The divide-and-conquer eigensolver routine DSYEVD is used throughout the test runs. The routine is linked from the following numerical libraries:

- CPU: Intel MKL Version 11.2.2
- KNL: Intel MKL Version 2017 Initial Release
- GPU: MAGMA Version 2.2

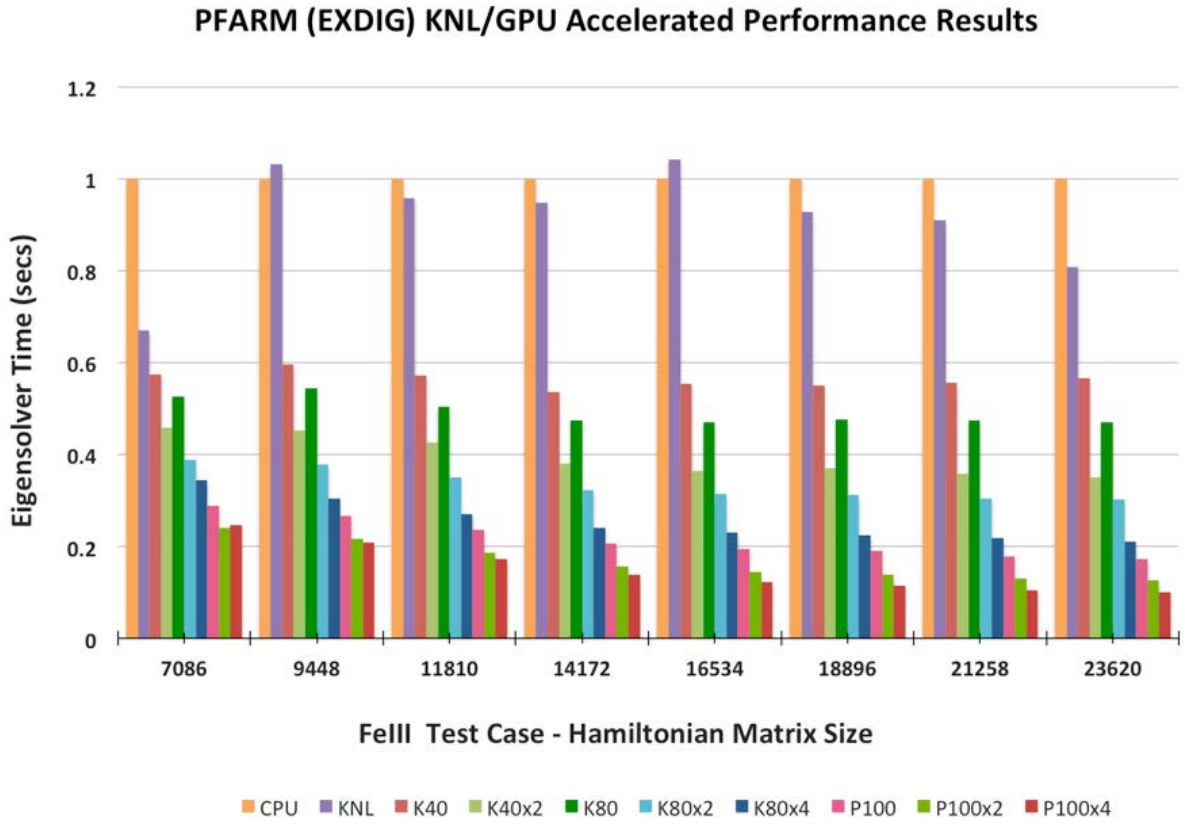

Figure 11 Eigensolver performance on KNL and GPU

EXDIG calculations are dominated by the eigensolver operations required to diagonalize each sector Hamiltonian matrix. Figure 11 summarizes eigensolver performance, using DSYEVD, over a range of problem sizes for the Xeon (CPU), Intel Knight's Landing (KNL) and a range of recent Nvidia GPU architectures. The results are normalised to the single node CPU performance using 24 OpenMP threads. The CPU runs use 24 OpenMP threads and the KNL

runs use 64 OpenMP threads. Dense linear algebra calculations tend to be bound by memory bandwidth, so using hyperthreading on the KNL or CPU is not beneficial. MAGMA is able to parallelise the calculation automatically across multiple GPU on a compute node and these results are denoted by the x2, x4 labels. Figure 11 demonstrates that MAGMA performance relative to CPU performance increases as problem size increases, due to the relative overhead cost of data transfer O(N^2) reducing compared to computational load O(N^3).

Test Case 1 – FeIII

Defining Computational Characteristics: 10 Fine Region Sector calculations involving Hamiltonian matrices of dimension 23620 and 10 Coarse Region Sector calculations involving Hamiltonian matrices of dimension 11810.

Test Case 2 – CH4

Defining Computational Characteristics: 10 'Spin 1' Coarse Sector calculations involving Hamiltonian matrices of dimension 5720 and 10 'Spin 2' Coarse Sector calculations involving Hamiltonian matrices of dimension 7890.

	CPU 24 threads	KNL 64 threads	K80	K80x2	K80x4	P100	P100x2	P100x4
Test Case 1 ; Atomic ; FeIII	4475	2610	1215	828	631	544	427	377
Test Case 2 ; Molecular ; CH4	466	346	180	150	134	119	107	111

Table 7 Overall EXDIG runtime performance on various accelerators (runtime, secs)

Table 7 records the overall run time on a range of architectures for both test cases described. For the complete runs (including I/O), both KNL-based and GPU-based computations significantly outperform the CPU-based calculations. For Test Case 1, utilising a node with single P100 GPU accelerator results in a runtime more than 8 times quicker than the CPU, correspondingly approximately 4 times quicker for Test Case 2. The smaller Hamiltonian matrices associated with Test Case 2 means that data transfer costs O(N2) are relatively high vs computation costs O(N3). Smaller matrices also result in poorer scaling as we increase the number of GPU per node for Test Case 2.

	1 MPI task per CPU node with 24 OpenMP threads per node		1 MPI task per KNL node with 64 OpenMP threads per node		1 MPI task per 2xM2070 GPUs	
Number of MPI tasks	Runtime (secs)	Speedup	Runtime (secs)	Speedup	Runtime (secs)	Speedup
1	5400	1	3141	1	1716	1
2	3042	1.77	1752	1.79	984	1.74
5	1094	4.94	637	4.93	377	4.55

Table 8 Overall EXDIG runtime parallel performance using MPI-GPU version

A relatively simple MPI harness can be used in EXDIG to farm out different sector Hamiltonian calculations to multiple CPU, KNL or GPU nodes. Table 8 shows that parallel scaling across nodes is very good for each test platform. This strategy is inherently scalable, however the replicated data approach requires significant amounts of memory per node. Test Case 1 is used as the dataset here, although the problem characteristics are slightly different to the setup used for Table 7, with 5 Fine Region sectors with Hamiltonian dimension of 23620 and 20 Coarse

Region sectors with Hamiltonian dimension of 11810. With these characteristics, runs using 2 MPI tasks experience inferior load-balancing in the Fine Region calculation compared to runs using 5 MPI tasks.

4.8 QCD

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

4.8.1 *First implementation*

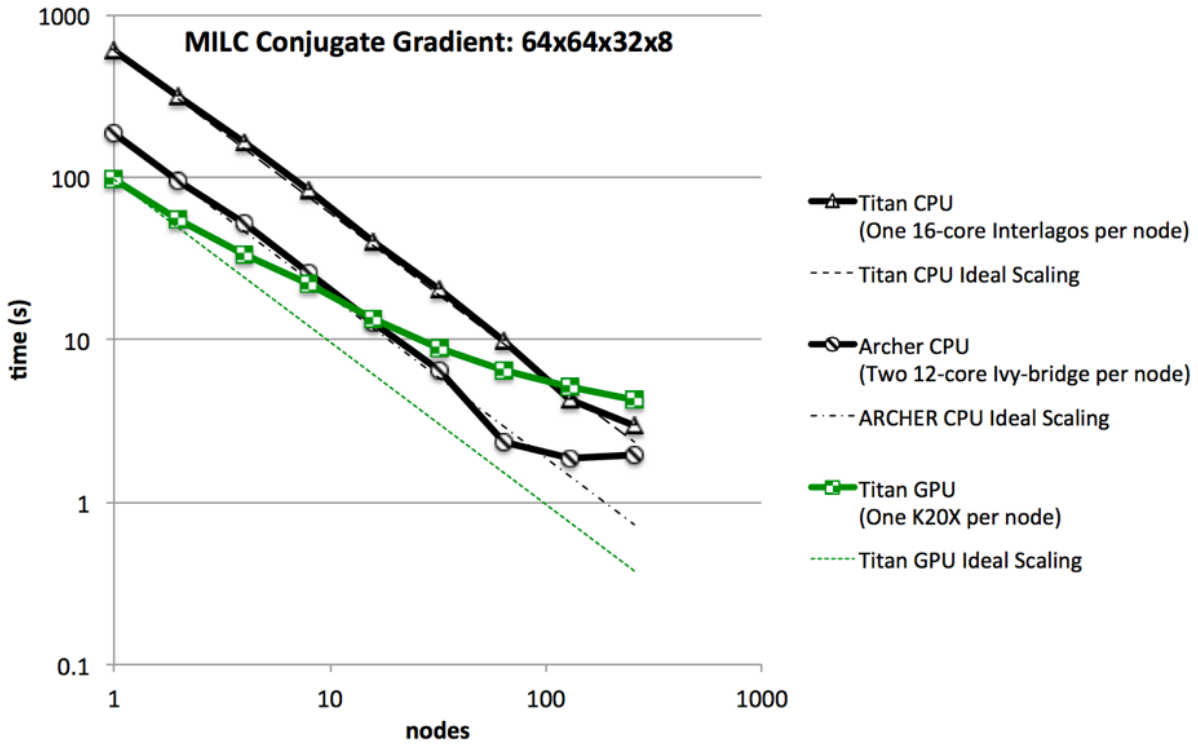

Figure 12 Small test case results for QCD, first implementation

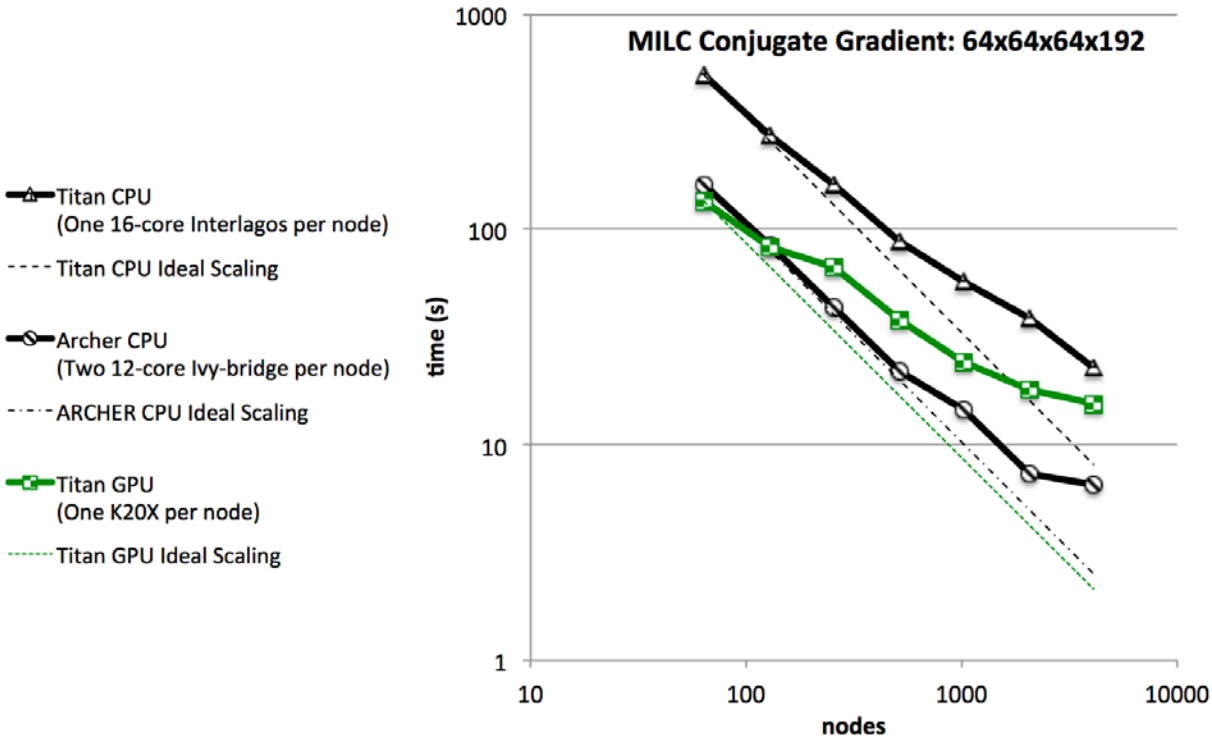

Figure 13 Large test case results for QCD, first implementation

The strong scaling, on Titan and ARCHER, for small (Figure 12) and large (Figure 13) problem sizes. For ARCHER, both CPU are used per node. For Titan, we include results with and without GPU utilization.

On each node, Titan has one 16-core Interlagos CPU and one K20X GPU, whereas ARCHER has two 12-core Ivy-bridge CPU. In this section, we evaluate on a node-by-node basis. For Titan, a single MPI task per node, operating on the CPU, is used to drive the GPU on that node. We also include, for Titan, results just using the CPU on each node without any involvement from the GPU, for comparison. This means that, on a single node, our Titan results will be the same as those K20X and Interlagos results presented in the previous section (for the same test case). On ARCHER, however, we fully utilize both the processors per node: to do this we use two MPI tasks per node, each with 12 OpenMP threads (via targetDP). So the single node results for ARCHER are twice as fast as those Ivy-bridge single-processor results presented in the previous section.

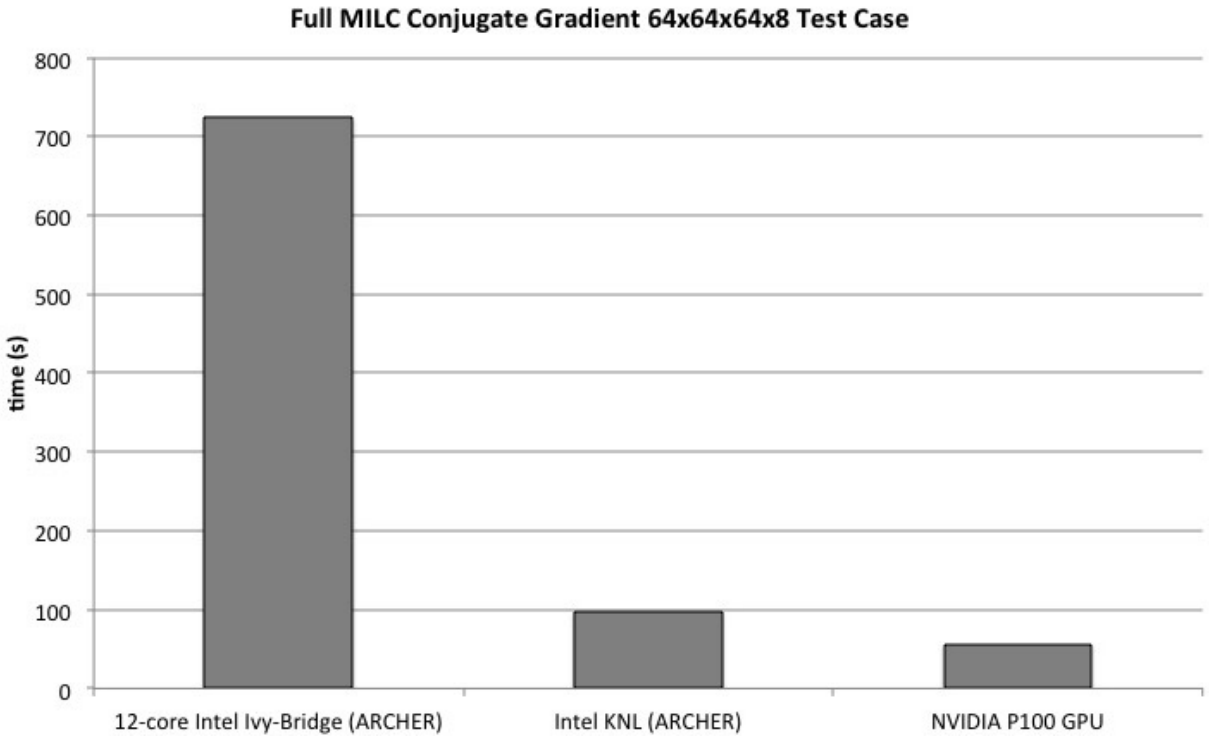

Figure 14 shows the time taken by the full MILC 64x64x64x8 test cases on traditional CPU, Intel Knights Landing Xeon Phi and NVIDIA P100 (Pascal) GPU architectures.

In Figure 14 we present preliminary results for on the latest generation Intel Knights Landing (KNL) and NVIDIA Pascal architectures, which offer very high bandwidth stacked memory, together with the same traditional Intel-Ivy-bridge CPU used in previous sections. Note that these results are not directly comparable with those presented earlier, since they are for a different test case size (larger since we are no longer limited by the small memory size of the Knights Corner), and they are for a slightly updated verion of the benchmark. The KNL is the 64-core 7210 model, available from within a test and development platform provided as part of the ARCHER service. The Pascal is a NVIDIA P100 GPU provided as part of the “Ouessant” IBM service at IDRIS, where the host CPU is an IBM Power8+.

It can be seen that the KNL is 7.5X faster than the Ivy-bridge; the Pascal is 13X faster than the Ivy-bridge; and the Pascal is 1.7X faster than the KNL.

4.8.2 *Second implementation*

GPU results

The GPU benchmark results of the second implementation are done on PizDaint located in Switzerland at CSCS and the GPU-partition of Cartesius at Surfsara based in Netherland, Amsterdam. The runs are performed by using the provided bash-scripts. PizDaint is equipped with one P100 Pascal-GPU per node. Two different test-cases are depicted, the "strong-scaling" mode with a random lattice configuration of size 32x32x32x96 and 64x64x64x128. The GPU nodes of Cartesius have two Kepler-GPU K40m per node and the "strong-scaling" test is shown for one card per node and for two cards per node. The benchmark kernel is using the conjugated gradient solver which solve a linear equation system given by D * x = b, for the unknown solution "x" based on the clover improved Wilson Dirac operator "D" and a known right hand side "b".

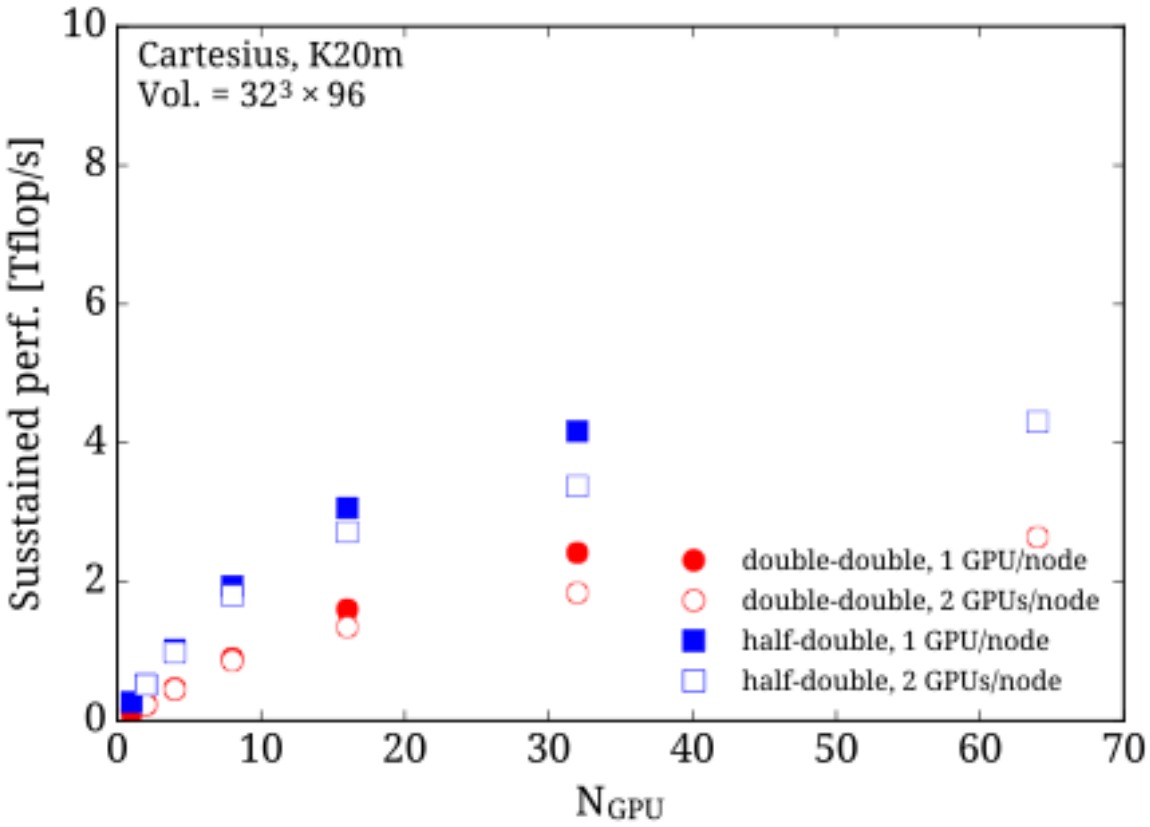

Figure 15 Result of second implementation of QCD on K40m GPU

Figure 15 shows strong scaling of the conjugate gradient solver on K40m GPU on Cartesius. The lattice size is given by 32x32x32x96, which corresponds to a moderate lattice size nowadays. The test is performed with a 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).

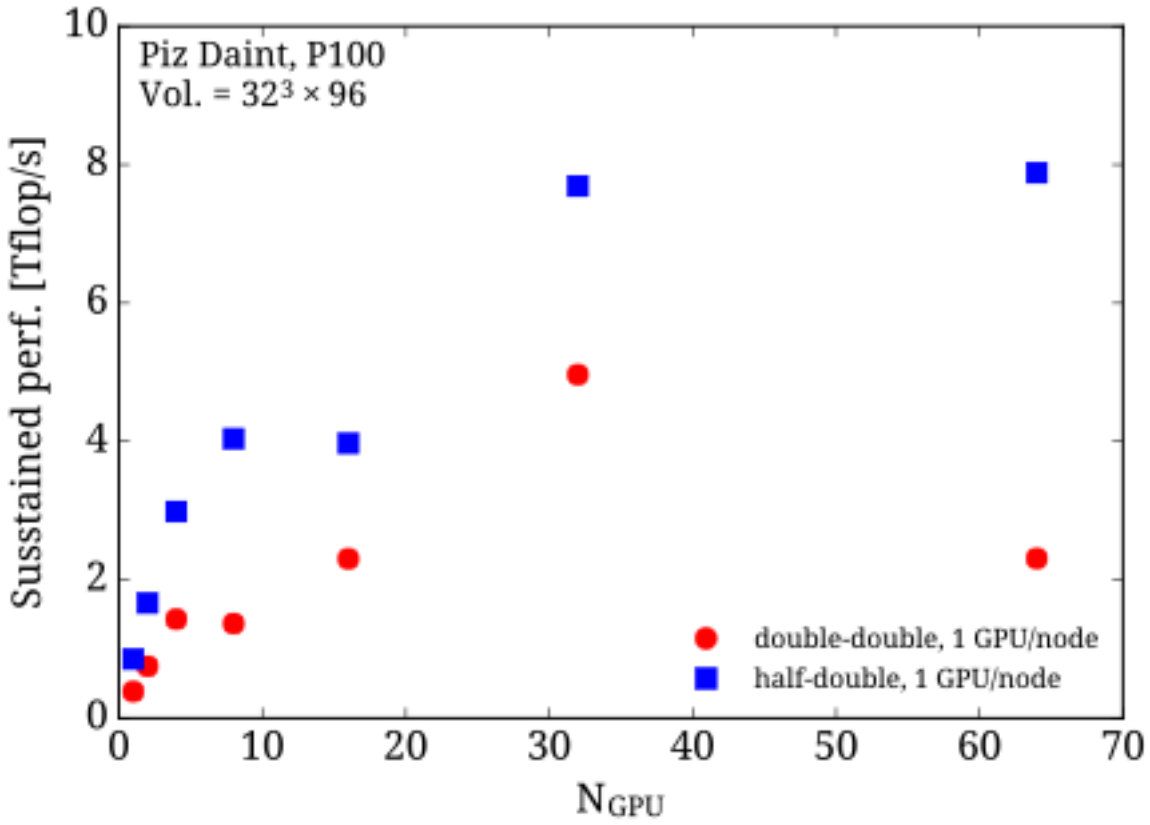

Figure 16 Result of second implementation of QCD on P100 GPU

Figure 16 shows strong scaling of the conjugate gradient solver on P100 GPU 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).

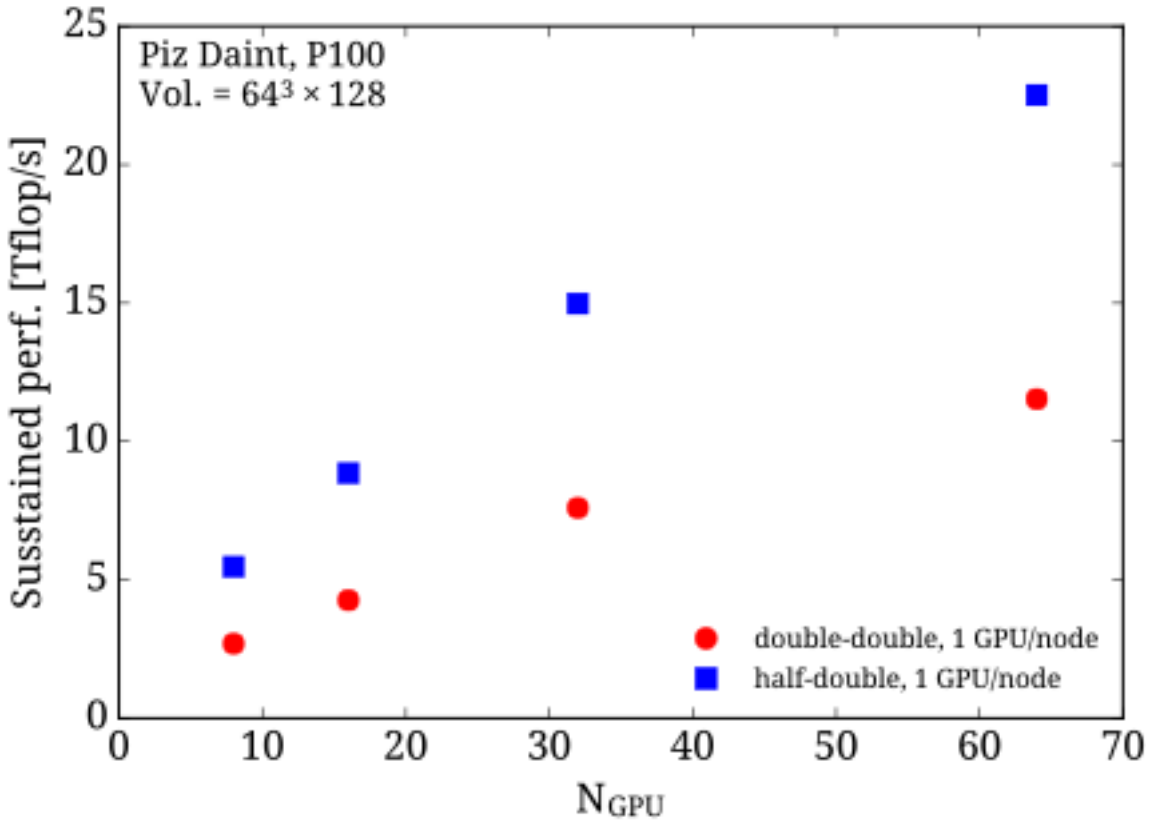

Figure 17 Result of second implementation of QCD on P100 GPU on larger test case

Figure 17 shows strong scaling of the conjugate gradient solver on P100 GPU on PizDaint. The lattice size is increase to 64x64x64x128, which is a large lattice nowadays. By increasing the lattice the scaling test shows that the conjugate gradient solver has a very good strong scaling up to 64 GPU.

Xeon Phi results

The benchmark results for the XeonPhi benchmark suite are performed on Frioul 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 KNC per node. The data on Frioul are generated by using the bash-scripts provided by the second implementation of QCD and are done for the two test cases "strong-scaling" with a lattice size of 32x32x32x96 and 64x64x64x128. In case of the data generated at MareNostrum, data for the "strong-scaling" mode on a 32x32x32x96 lattice are shown. The benchmark kernel uses a random gauge configuration and the conjugated gradient solver to solve a linear equation involving the clover Wilson Dirac operator.

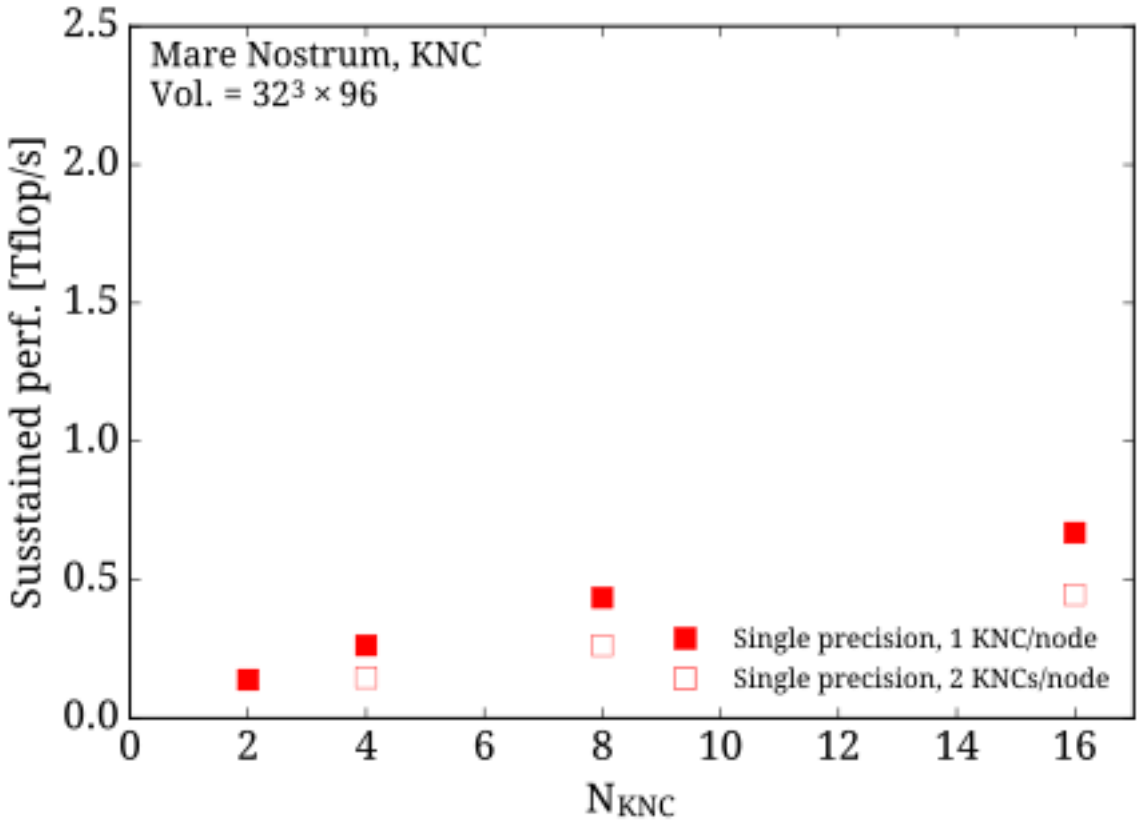

Figure 18 Result of second implementation of QCD on KNC

Figure 18 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 process. The run is done on one KNC per node (filled) and two KNC node per node (non-filled).

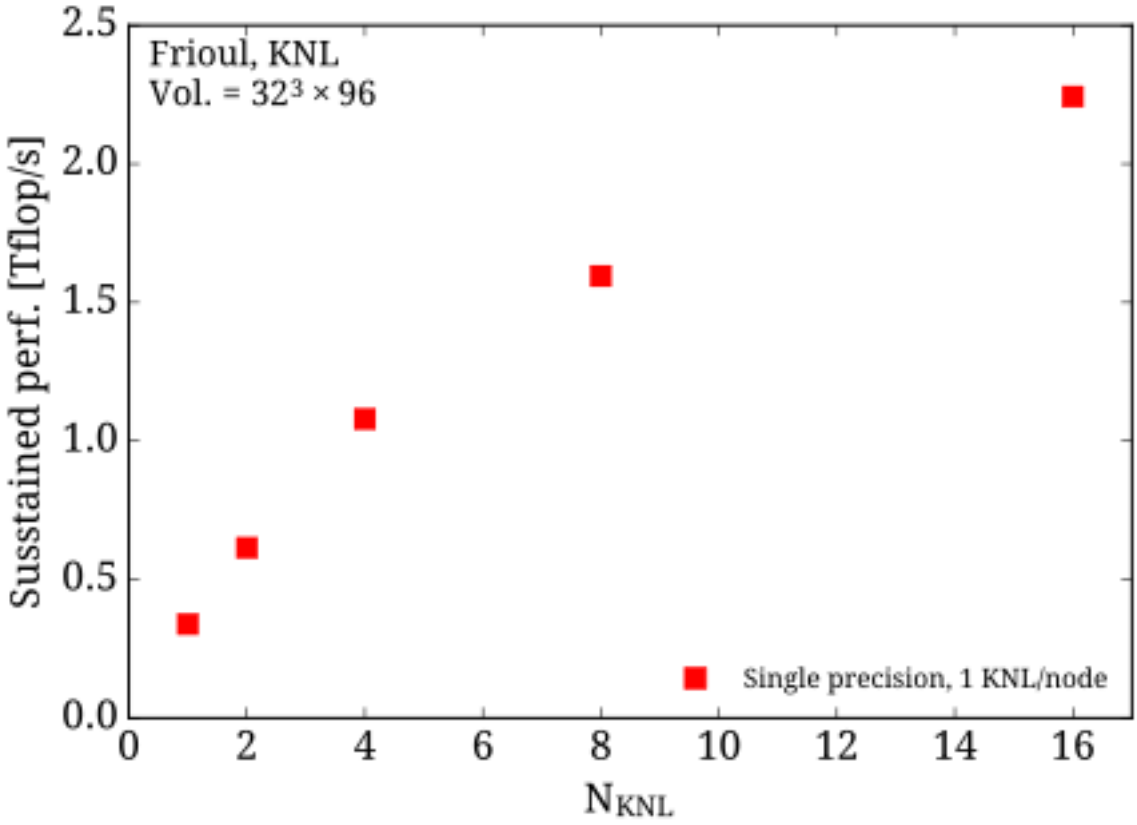

Figure 19 Result of second implementation of QCD on KNL

Figure 19 shows strong scaling results of the conjugate gradient solver on KNL's on Frioul. The lattice size is given by 32x32x32x96 which is similar to the strong scaling run on the KNC on MareNostrum III. The run is performed in quadrantic cache mode with 68 OpenMP processes per KNL. The test is performed with a conjugate gradient solver in single precision.

4.9 Quantum Espresso

Here are sample results for Quantum Espresso. This code has run on Cartesius (see section 2.2.1) and Marconi (1 node is 1 standalone KNL Xeon Phi 7250, 68 core 1.40 GHz, 16BG MCDRAM, 96BG DDR4 RAM, interconnect is Intel OmniPath).

Runs on GPU

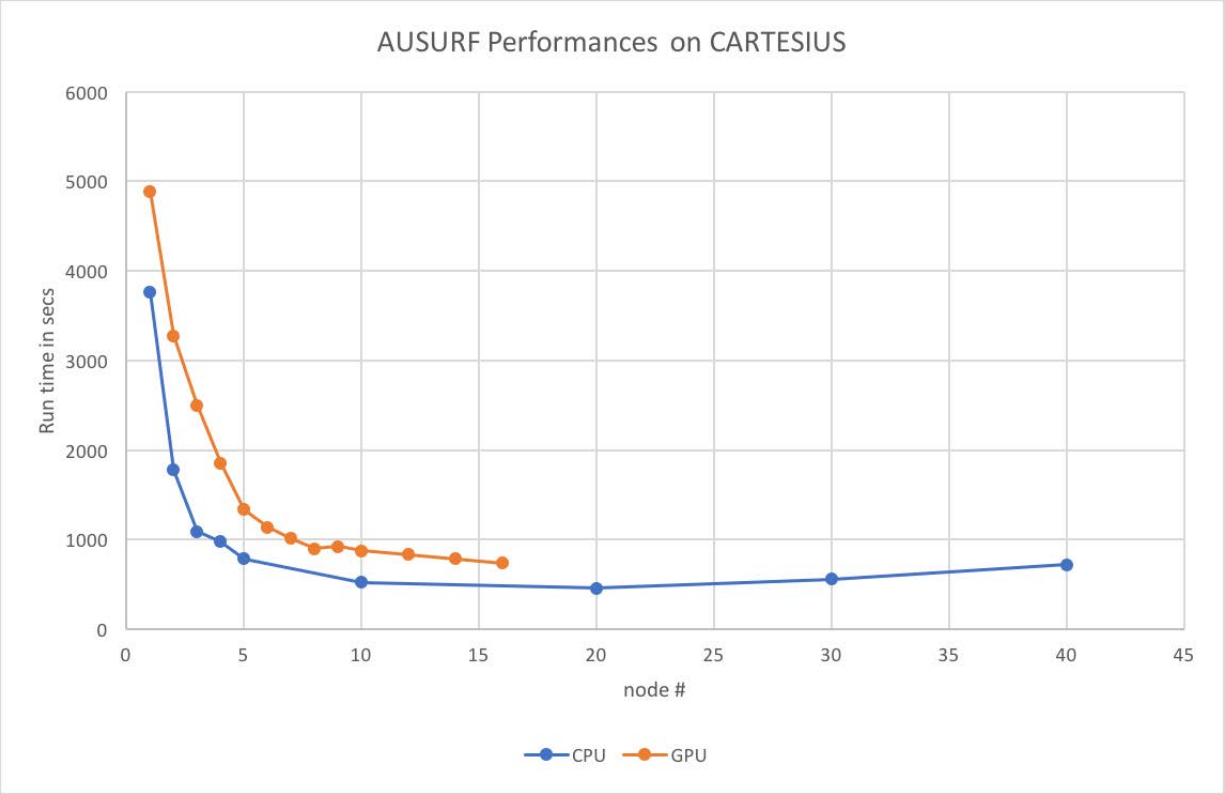

Figure 20 Scalability of Quantum Espresso on GPU for test case 1

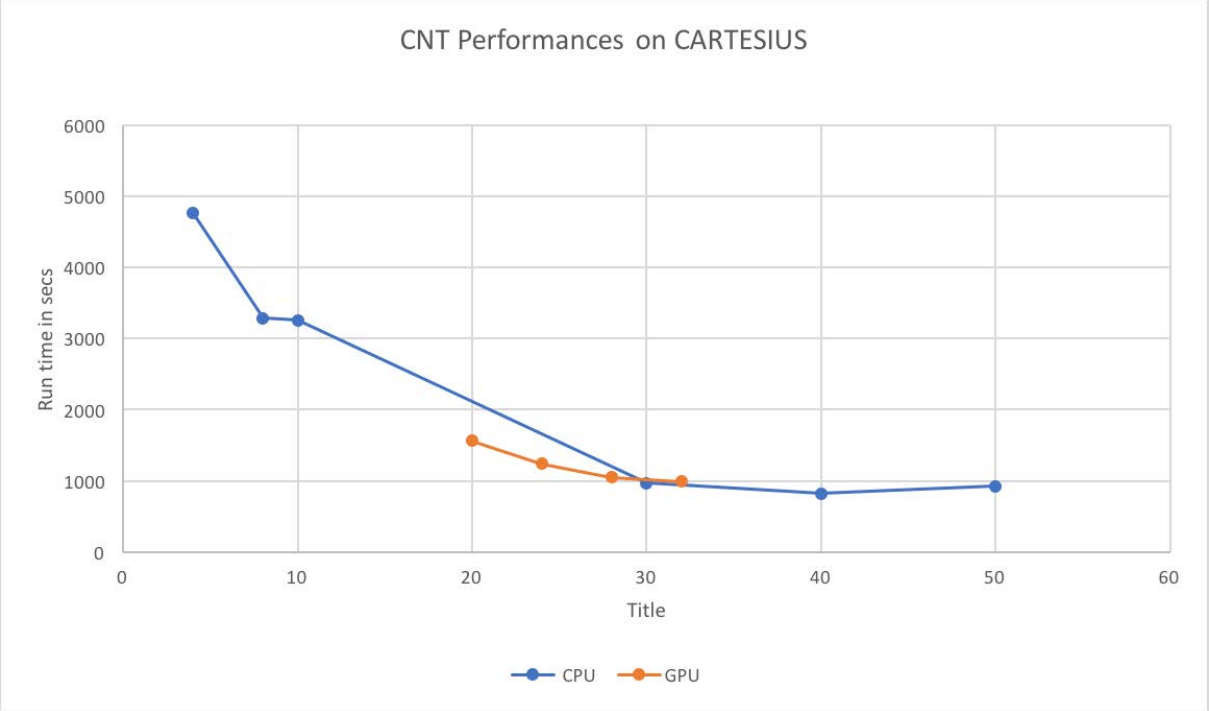

Figure 21 Scalability of Quantum Espresso on GPU for test case 2

Test cases (Figure 20 and Figure 21) show no appreciable speed-up with GPU. Inputs are probably too small, they should evolve in the future of this benchmark suite.

Runs on KNL

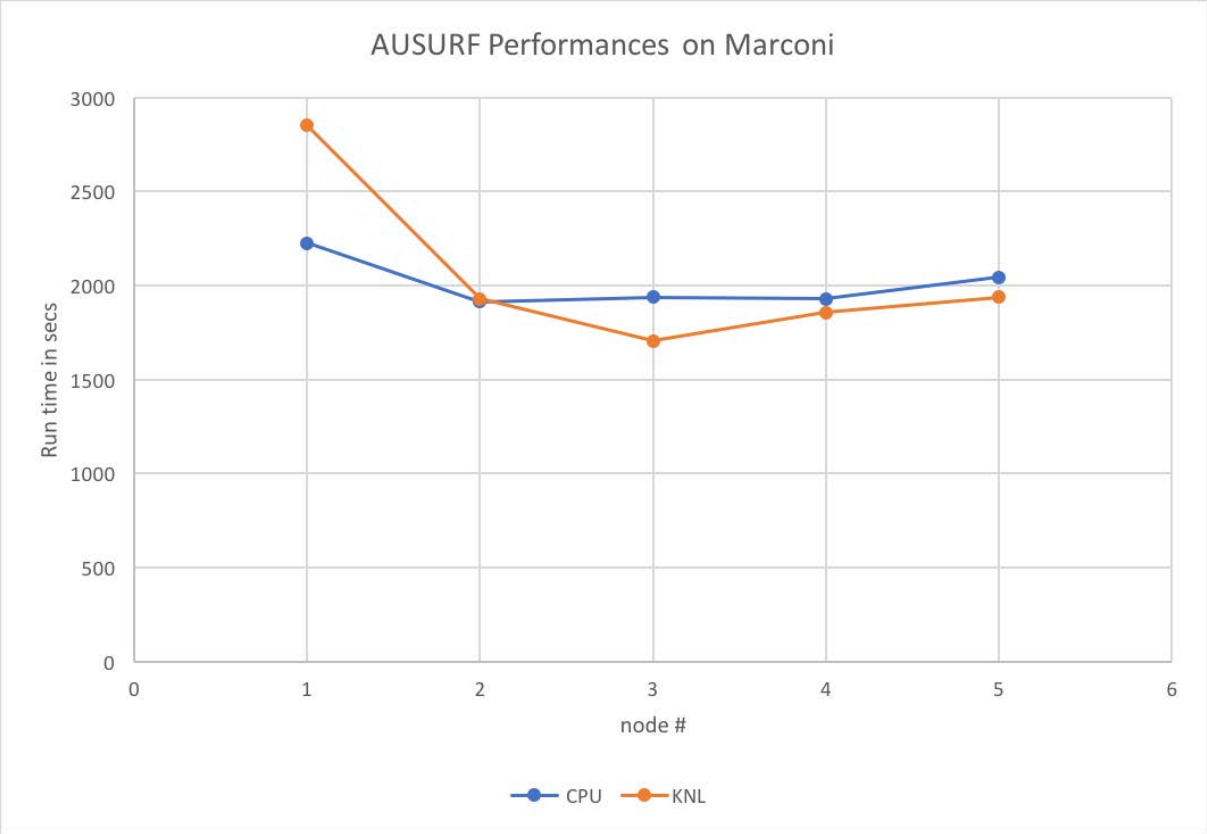

Figure 22 Scalability of Quantum Espresso on KNL for test case 1

Figure 22 shows the usual pw.x with the small test case A (AUSURF), comparing Marconi Broadwell (36 cores/node) with KNL (68 cores/node) - this test case is probably small for testing on KNL.

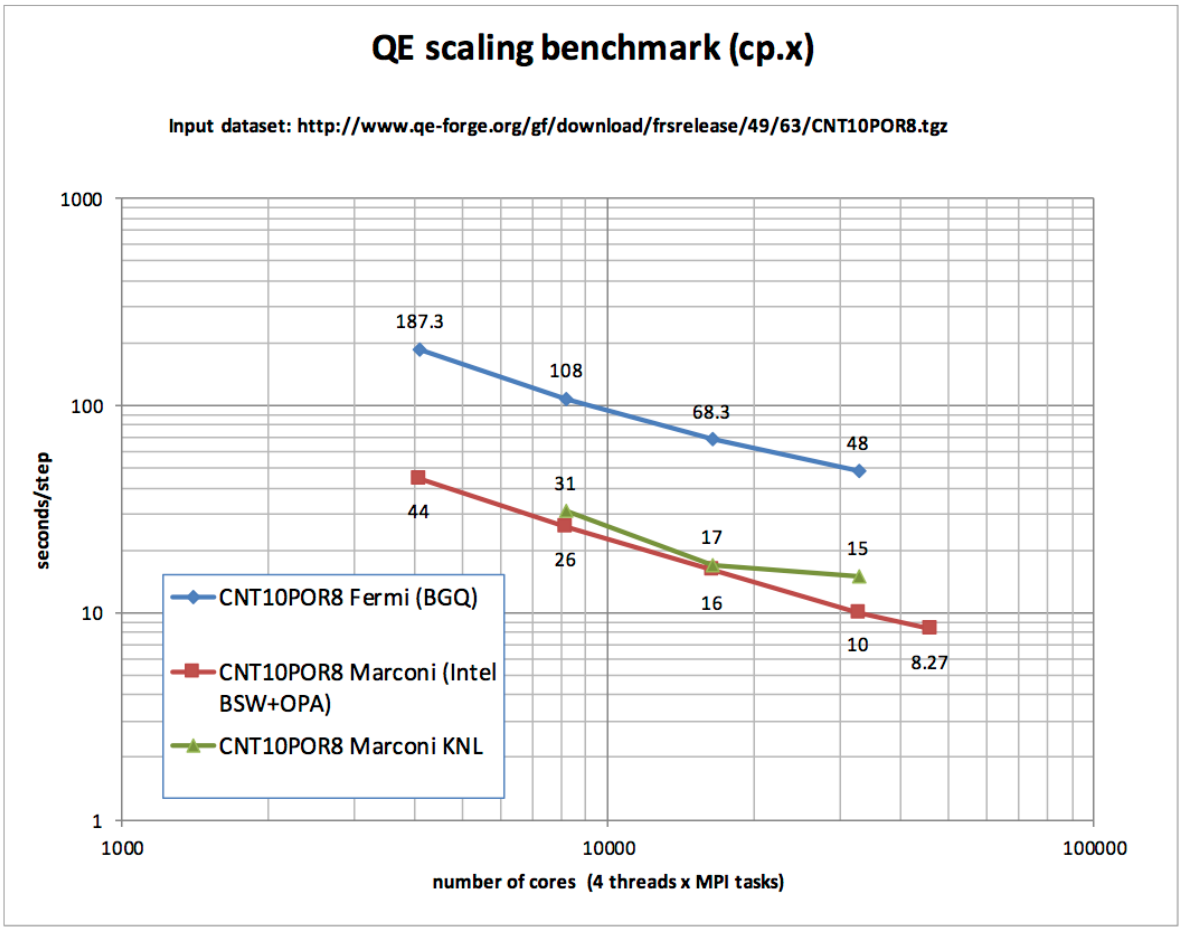

Figure 23 Quantum Espresso - KNL vs BDW vs BGQ (at scale)

Figure 23 presents CNT10POR8 which is the large test case, even though it is using the cp.x executable (i.e. Car-parinello) rather than the usual pw.x (PW SCF calculation).

4.10 Synthetic benchmarks (SHOC)

The SHOC benchmark has been run on Cartesius, Ouessant and MareNostrum. Table 9 presents the results:

	NVIDIA GPU			Intel Xeon Phi		
	K40 CUDA	K40 OpenCL	Power 8 + P100 CUDA	KNC Offload	KNC OpenCL	Haswell OpenCL
BusSpeedDownload	10.5 GB/s	10.56 GB/s	32.23 GB/s	6.6 GB/s	6.8 GB/s	12.4 GB/s
BusSpeedReadback	10.5 GB/s	10.56 GB/s	34.00 GB/s	6.7 GB/s	6.8 GB/s	12.5 GB/s
maxspflops	3716 GFLOPS	3658 GFLOPS	10424 GFLOPS	21581	2314 GFLOPS	1647 GFLOPS

maxdpflops	1412 GFLOPS	1411 GFLOPS	5315 GFLOPS	16017	2318 GFLOPS	884 GFLOPS
gmem_readbw	177 GB/s	179 GB/s	575.16 GB/s	170 GB/s	49.7 GB/s	20.2 GB/s
gmem_readbw_strided	18 GB/s	20 GB/s	99.15 GB/s	N/A	35 GB/s	156 GB/s
gmem_writebw	175 GB/s	188 GB/s	436 GB/s	72 GB/s	41 GB/s	13.6 GB/s
gmem_writebw_strided	7 GB/s	7 GB/s	26.3 GB/s	N/A	25 GB/s	163 GB/s
lmem_readbw	1168 GB/s	1156 GB/s	4239 GB/s	N/A	442 GB/s	238 GB/s
lmem_writebw	1194 GB/s	1162 GB/s	5488 GB/s	N/A	477 GB/s	295 GB/s
BFS	49,236,500 Edges/s	42,088,000 Edges/s	91,935,100 Edges/s	N/A	1,635,330 Edges/s	14,225,600 Edges/s
FFT_sp	523 GFLOPS	377 GFLOPS	1472 GFLOPS	135 GFLOPS	71 GFLOPS	80 GFLOPS
FFT_dp	262 GFLOPS	61 GFLOPS	733 GFLOPS	69.5 GFLOPS	31 GFLOPS	55 GFLOPS
SGEMM	2900-2990 GFLOPS	694/761 GFLOPS	8604-8720 GFLOPS	640/645 GFLOPS	179/217 GFLOPS	419-554 GFLOPS
DGEMM	1025-1083 GFLOPS	411/433 GFLOPS	3635-3785 GFLOPS	179/190 GFLOPS	76/100 GFLOPS	189-196 GFLOPS
MD (SP)	185 GFLOPS	91 GFLOPS	483 GFLOPS	28 GFLOPS	33 GFLOPS	114 GFLOPS
MD5Hash	3.38 GH/s	3.36 GH/s	15.77 GH/s	N/A	1.7 GH/s	1.29 GH/s
Reduction	137 GB/s	150 GB/s	271 GB/s	99 GB/s	10 GB/s	91 GB/s
Scan	47 GB/s	39 GB/s	99.2 GB/s	11 GB/s	4.5 GB/s	15 GB/s
Sort	3.08 GB/s	0.54 GB/s	12.54 GB/s	N/A	0.11 GB/s	0.35 GB/s
Spmv	4-23 GFLOPS	3-17 GFLOPS	23-65 GFLOPS	1-17944 GFLOPS	N/A	1-10 GFLOPS
Stencil2D	123 GFLOPS	135 GFLOPS	465 GFLOPS	89 GFLOPS	8.95 GFLOPS	34 GFLOPS
Stencil2D_dp	57 GFLOPS	67 GFLOPS	258 GFLOPS	16 GFLOPS	7.92 GFLOPS	30 GFLOPS
Triad	13.5 GB/s	9.9 GB/s	43 GB/s	5.76 GB/s	5.57 GB/s	8 GB/s
S3D (level2)	94 GFLOPS	91 GFLOPS	294 GFLOPS	109 GFLOPS	18 GFLOPS	27 GFLOPS

Table 9 Synthetic benchmarks results on GPU and Xeon Phi

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

- KNC MaxFlops (both SP and DP): In this case the compiler optimizes away some of the computation (although it shouldn't) [19].
- KNC SpMV: For these benchmarks it is a known bug currently being addressed [20].
- Haswell gmem_readbw_strided and gmem_writebw_strided: strided read/write benchmarks doesn't make too much sense in the case of the CPU, as the data will be cache in the large L3 caches. It is reason why we see high number only in the Haswell case.

4.11 SPECFEM3D

Tests have been carried out on Ouessant and Firoul.

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 10 SPECFEM 3D GLOBE results (run time in second)

5 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.