SEVENTH FRAMEWORK PROGRAMME
Research Infrastructures

INFRA-2010-2.3.1 – First Implementation Phase of the European High Performance Computing (HPC) service PRACE

PRACE-1IP

PRACE First Implementation Project

Grant Agreement Number: RI-261557

D7.2.2
Final Report on Collaboration with Communities

Final

Version: 1.0
Author(s): Giovanni Erbacci, CINECA
John Donners, SARA
Joerg Hertzer, HLRS
Maria Francesca Iozzi, UiO
Paul Sherwood, STFC

Date: 25.06.2012

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: RI-261557**	
	Project Title: PRACE First Implementation Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: **D7.2.2**	
	Deliverable Nature: <DOC_TYPE: Report / Other>	
	Deliverable Level: PU *	**Contractual Date of Delivery:** 30 / June / 2012
		Actual Date of Delivery: 30 / June / 2012
	EC Project Officer: Thomas Reibe	

* - The dissemination level are indicated as follows: **PU** – Public, **PP** – Restricted to other participants (including the Commission Services), **RE** – Restricted to a group specified by the consortium (including the Commission Services). **CO** – Confidential, only for members of the consortium (including the Commission Services).

Document Control Sheet

Document	**Title: Final Report on Collaboration with Communities**	
	ID: D7.2.2	
	Version: 1.0	**Status:** Final
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D7.2.2.docx	
Authorship	**Written by:**	Giovanni Erbacci, CINECA, CINECA John Donners, SARA Joerg Hertzer, HLRS Maria Francesca Iozzi, UiO Paul Sherwood, STFC

	Contributors:	Berk Hess, KTH Erik Lindahl, KTH Michael Schliephake, KTH Rossen Apostolov, KTH Lilit Axner, KTH Mikael Rannar, SNIC Fabio Affinito, CINECA Carlo Cavazzoni, CINECA Massimiliano Culpo, CINECA Andrew Emerson, CINECA Ivan Spisso, CINECA Leandar Litov, NCSA Valentin Pavlov, NCSA Peicho Petkov, NCSA Ivan Girotto, ICHEC Michael Lysaght, ICHEC Alastair McKinstry, ICHEC Michael Moyles, ICHEC Peter Nash, ICHEC Filippo Spiga, ICHEC Nicola Varini, ICHEC Jussi Enkovaara, CSC Martti Louhivuori, CSC Vladimir Slavnic, IPB Iain Bethune, EPCC Adam Carter, EPCC Gavin Pringle, EPCC Kevin Stratford, EPCC Paschalis Korosoglou, AUTH Simen Reine, UoO Thomas Kjærgaard, UoO Trygve Helgaker, UoO, Olav Vahtras, KTH Zilvinas Rinkevicius, KTH Bogdan Frecus, KTH Thomas W. Keal, STFC Andrew Sunderland, STFC Andrew Porter, STFC C. Moulinec, STFC A.G. Sunderland, STFC Ilian Todorov, STFC Xavier Aguilar, PDC Ole Widar Saastad, UoO Judit Gimenez, BSC Mariusz Uchronski, PSNC-WCNS Agnieszka Kwiecien, PSNC-WCNS Marcin Gabarowski, PSNC-WCNS Chandan Basu, SNIC-LiU Marcin Zielinski, SARA Dalibor Lukas, VSB Petr Kovar, VSB Tereza Kovarova, VSB Jan Zapletal, VSB Orlando Rivera, LRZ Karl Fürlinger, LMU Murat Manguoglu, METU Bjørn Lindi, NTNU Petar Jovanovic (IPB) P. Kabelikova, VSB, A. Ronovsky, VSB, V. Vondrak, VSB, A. Turk, BU, C. Aykanat, BU C. Theodosiou, AU P. Kabelikova, VSB Ales Ronovsky, VSB Vit Vondrak, VSB

	Reviewed by:	Krassimir Georgiev, NCSA Florian Berberich, JSC
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	10/May/2012	Draft	Formatting issues clarified
0.2	25/May/2012	Draft	Initial skeleton draft
0.3	30/May/2012	Draft	First rough draft
0.4	01/June/2012	Draft	Draft chapter on Material Sciences
0.5	06/June/2012	Draft	Draft chapter on CFD
0.6	08/June/2012	Draft	Draft chapter on Chemistry
0.7	09/June/2012	Draft	Draft version of all chapters
0.8	11/June/2012	Draft	Version ready for internal review
1.0	25/June/2012	Final version	Version ready for EC submission

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure, Scientific communities, Application codes, Petascaling

Table of Contents

List of Figures

List of Tables

References and Applicable Documents

[1] http://www.prace-project.eu
[2] PRACE 1IP Deliverable D 7.2.1, "*Interim Report on Collaboration with Communities*", June 2011.
[3] PRACE 1IP Deliverable D 7.5, "*HPC Programming Techniques*", April 2012.
[4] PRACE 1IP Deliverable D 7.6, "*Efficient Handling of Petascale Data*", December 2011.
[5] https://is.enes.org
[6] www.mapper-project.eu
[7] www.scalalife.eu.
[8] www.verce.eu
[9] www.epos-eu.org
[10] www.prace-ri.eu/white-papers
[11] Improving the scalability of CP2K on multi-core systems: A dCSE Project, 2010, http://www.hector.ac.uk/cse/distributedcse/reports/cp2k02/cp2k02_final_report.pdf
[12] CP2K - Sparse Linear Algebra on 1000s of cores: A dCSE Project, 2012, http://www.hector.ac.uk/cse/distributedcse/reports/cp2k03/cp2k03.pdf
[13] *Molecular Dynamics Studies of Radiation Hard Materials*, Ian J. Bush and Ilian T. Todorov, CCLRC Daresbury Laboratory, United Kingdom. Report on the Jülich Blue Gene/L Scaling Workshop 2006, editors: Wolfgang Frings, Marc-André Hermanns, Bernd Mohr, Boris Orth, FZJ-ZAM-IB-2007-02, February 2007
[14] METIS: unstructured graph partitioning and sparse matrix ordering system. http://glaros.dtc.umn. edu/gkhome/views/metis
[15] ParMetis: parallel graph partitioning and fill-reducing matrix ordering. http://glaros.dtc.umn.edu/ gkhome/metis/parmetis/overview
[16] PT-SCOTCH, Software package and libraries for sequential and parallel graph partitioning. http:// www.labri.fr/perso/pelegrin/scotch
[17] ZOLTAN, Parallel Partitioning, Load Balancing and Data-Management Services. http://www.cs. sandia.gov/Zoltan/

List of Acronyms and Abbreviations

ADF	Amsterdam Density Functional software
AISBL	Association internationale sans but lucratif
BCO	Benchmark Code Owner
BG/P	BlueGene/P computer
BLAS	Basic Linear Algebra Subprograms
BSC	Barcelona Supercomputing Center (Spain)
BSCW	Basic Support for Cooperative Work (software package for collaboration over the Web)
CC	Coupled cluster. A numerical technique for describing many-body systems, used in quantum chemical post-Hartree–Fock ab initio quantum chemistry methods
ccNUMA	cache coherent NUMA
CEA	Commissariat à l'Energie Atomique (represented in PRACE by GENCI, France)
CERFACS	Centre Européen de Recherche et de Formation Avancée en Calcul, France
CERN	Organisation Européenne pour la Recherche Nucléaire
CFD	Computational Fluid Dynamics
CI	Configuration Interaction method. A post-Hartree–Fock linear variational method
CIEMAT	Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas
CINECA	Consorzio Interuniversitario, the largest Italian computing centre (Italy)
CMB	Cosmic Microwave Background
CNR	Italian National Research Council
CNR-IOM	CNR, Istituto di Officina Molecolare
CNR-STM	CNR, Istituto di Scienze e Tecnologie Molecolari
CPU	Central Processing Unit
CRPP	Centre de Recherches en Physique des Plasmas in Lausanne
CSC	Finnish IT Centre for Science (Finland)
CSCS	The Swiss National Supercomputing Centre (represented in PRACE by ETHZ, Switzerland)
CSR	Compressed Sparse Row (for a sparse matrix)
CUDA	Compute Unified Device Architecture (NVIDIA)
DARPA	Defense Advanced Research Projects Agency
DBCSR	Distributed Block Compressed Sparse Row library
DDPS	Domain-decomposing parallel sparse solver
DECI	DEISA Extreme Computing Initiative.
DEISA	Distributed European infrastructure for Supercomputing Applications. EU project by leading national HPC centres
DEMOCRITOS	DEmocritos MOdeling Centre for Research In aTOmistic Simulations, Trieste (Italy)
DFT	Density Functional Theory
DGEMM	Double precision General Matrix Multiply
DIC	Diagonal Incomplete-Cholosky
DILU	Diagonal Incomplete-LU
DoE	Department of Energy
DP	Double Precision, usually 64-bit floating point numbers
EC	European Community
ECMWF	European Centre for Medium-Range Weather Forecasts
EDF	Électricité de France

EESI	European Exascale Software Initiative
EFDA	European Fusion Development Agreement
ELPA	EigenvaLue solver for Petaflop Applications
EMBL	European Molecular Biology Laboratory
EPCC	Edinburg Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)
EPOS	European Plate Observing System. European research infrastructure in solid Earth Science
EPSRC	The Engineering and Physical Sciences Research Council (United Kingdom)
ESM	Earth System Model
ETHZ	Eidgenössische Technische Hochschule Züerich, ETH Zurich (Switzerland)
ESFRI	European Strategy Forum on Research Infrastructures; created roadmap for pan-European Research Infrastructure.
FDIC	Faster diagonal Incomplete-Cholesky
FFT	Fast Fourier Transform
FP	Floating-Point
FZJ	Forschungszentrum Jülich (Germany)
GAMG	Geometric-Algebraic Multigrid
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
GCS	Gauss Centre for Supercomputing (Germany)
GENCI	Grand Equipement National de Calcul Intensif (France)
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
GIPAW	Gauge-Including Projector-Augmented Wave Method. A first principles theory of solid state NMR
GNU	GNU's not Unix, a free OS
GPAW	Grid-based Projector-Augmented Wave method. GPAW is a DFT Python code based on the PAW method.
GPGPU	General Purpose GPU
GPL	GNU General Public License
GPU	Graphic Processing Unit
GPW	Gaussian and Plane Wave approach
GRIB	GRIdded Binary, a concise data format commonly used in meteorology
GROMACS	GROningen MAchine for Chemical Simulations. A molecular dynamics simulation package originally developed in the University of Groningen
GWU	George Washington University, Washington, D.C. (USA)
HECToR	High-End Computing Terascale Resource (UK's national supercomputing service)
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
HF	Hartree-Fock method
HiPER	European High Power laser Energy Research facility
HLRS	High Performance Computing Center Stuttgart, Germany

HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
HPL	High Performance LINPACK
IB	InfiniBand
IBM	International Business Machines Corporation
ICE	Internal Combustion Engine Group (at Politecnico di Milano, Italy)
ICHEC	Irish Centre for High-End Computing
IC3	Catalonian institute for climate sciences, Spain
IDRIS	Institut du Développement et des Ressources en Informatique Scientifique (represented in PRACE by GENCI, France)
IFS	Integrated Forecast System. An operational global meteorological forecasting model developed by ECMWF
I/O	Input/Output
IOIPSL	I/O institute Pierre Simon Laplace library. The IPSL I/O library
IPB	Institute of Physics Belgrade
IPCC	Intergovernmental Panel on Climate Change
IPM	Integrated Performance Monitoring (a portable profiling infrastructure for parallel codes)
IPP	Max-Planck-Institut für Plasmaphysik
IPSL	Institut Pierre Simon Laplace, France.
IS-ENES	InfraStructure for the European Network for the Earth System Modelling; an FP7-Project
ISV	Independent Software Vendor
ITER	International Thermonuclear Experimental Reactor
JSC	Jülich Supercomputing Centre (FZJ, Germany)
KNMI	Royal Dutch Meteorological Institute, The Netherlands.
KTH	Kungliga Tekniska Högskolan (represented in PRACE by SNIC, Sweden)
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator. A molecular dynamics code developed at Sandia National Laboratories
LAPACK	Linear Algebra PACKage
LES	Large eddy simulation
LINPACK	Software library for Linear Algebra
LiU	Linkoping University, Sweden
LOBPCG	Local Optimal Block Preconditioned Conjugate Gradient algorithm
LQCD	Lattice QCD
LR-TDDFT	Linear Response Time Dependent DFT formalism
LRZ	Leibniz Supercomputing Centre (Garching, Germany)
MAPPER	Multiscale APPlications on European e-infRastructures. FP7 EU Projetc
MCSCF	Multi-Configurational Self-Consistent Field method
MD	Molecular Dynamics
MFlop/s	Mega (= 10^6) Floating point operations (usually in 64-bit, i.e. DP) per second, also MF/s
MHz	Mega (= 10^6) Hertz, frequency =10^6 periods or clock cycles per second
MKL	Math Kernel Library (Intel)
MPI	Message Passing Interface
NAMD	Not (just) Another Molecular Dynamics program. A parallel molecular dynamics code for large biomolecular systems
NCF	Netherlands Computing Facilities (Netherlands)
NEMO	Nucleus for European Modelling of the Ocean: A state-of-the-art modelling framework for oceanographic research,

NMR	Nuclear Magnetic Resonance
NTNU	Norwegian University of Science and Technology, Norway
NUI	National University of Ireland
NUMA	Non-Uniform Memory Access or Architecture
OASIS	Ocean Atmosphere Sea Ice Soil. A tool for coupling the atmosphere and ocean components of a meteo-climatology model
OpenCL	Open Computing Language
OpenFOAM	Open Field Operation And Manipulation. Open source CFD software
OpenMP	Open Multi-Processing
OS	Operating System
PABS	PRACE Application Benchmark Suite
PAW	Projector Augmented Wave method
PBLAS	Parallel Basic Linear Algebra Subprograms
PISO	Pressure Implicit solution by Split Operator method
PME	Particle-Mesh Ewald method
PNNL	Pacific Northwest National Laboratory in Richland Washington
PP	Particle Particle force calculation
PRACE	Partnership for Advanced Computing in Europe; Project Acronym
PW	plane waves
PWR	Pressurised water reactor
PWscf	Plane-Wave Self-Consistent Field. A code for electronic structure calculations.
QCD	Quantum Chromodynamics
QDR	Quad Data Rate
QE	Quantum Espresso
QM	Quantum Methods
QM/MM	Quantum Mechanics/Molecular Mechanics
QR	QR method or algorithm: a factorisation of a matrix into a unitary and upper triangular matrix
RMDIIS	Residual Minimization with Direct Inversion In Subspace algorithm
RT-TDDFT	Real Time - Time Dependent DFT formalism
SARA	Stichting Academisch Rekencentrum Amsterdam (Netherlands)
ScaLAPACK	Scalable LAPACK library
SEM	Spectral-Element Method
SGEMM	Single precision General Matrix Multiply, subroutine in the BLAS
SGI	Silicon Graphics, Inc.
SIESTA	Spanish Initiative for Electronic Simulations with Thousands of Atoms. A code for performing electronic structure calculations and ab initio molecular dynamics simulations
SISSA	International School for Advanced Studies, Trieste (Italy)
SMHI	Swedish Meteorological and Hydrological Institute, Sweden
SNIC	Swedish National Infrastructure for Computing (Sweden)
SP	Single Precision, usually 32-bit floating point numbers
SPC	Simple Point Charge model. A common simulation model for water
STFC	Science and Technology Facilities Council (represented in PRACE by EPSRC, United Kingdom)
SVN	Subversion system. A software versioning and a revision control system
TD-DFT	Time-dependent density functional theory. A quantum mechanical theory used in physics and chemistry to investigate the properties and dynamics of many-body systems in the presence of time-dependent potentials

TFlop/s	Tera (= 10^{12}) 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
Tier-1	Denotes the national or topical HPC centres in the pyramid of HPC systems
UT	University of Technology
UKTC	UK Turbolence Consortium
VASP	Vienna Ab-Initio Simulation Package
VERCE	Virtual Earthquake and seismology Research Community in Europe e-science. FP7 EU project
VKI	Von Karman Institute for Fluid Dynamics
VMEC	Variational Moments Equilibrium Code
Wikki	Wikki Ltd (UK) / Wikki GmbH (Germany)
XC	Exchange-correlation functionals
2D	2-Dimensional
3D	3-Dimensional

Executive Summary

The present document describes the work carried out during the second year of activity of Task 7.2-1IP ”*Application Enabling with communities”*. Eleven key application codes of value to the main scientific communities at European level were selected for petascaling during the first year of Task 7.2. These applications, emblematic of the major computational disciplines and challenging for the main scientific communities are: GROMACS and DL-POLY for Molecular Dynamics; Quantum ESPRESSO and CP2K, for Material Science; GPAW for Nanoscience; DALTON for Computational Chemistry; EUTERPE for Plasma Physics; the EC-Earth 3 suite for Meteo-climatology, SPECFEM3D for Earth Sciences (earthquakes) and OpenFOAM and Code_Saturne for Engineering and CFD.

The enabling and optimization activity started after the selection of the codes and completed at the end of the second year of PRACE-1IP, allowing and reinforcing a long term cooperation with the communities and the owners of the codes. The results of this activity are significant: for some application codes the improvements were integrated in the official distribution releases of the software (i.e. Quantum ESPRESSO, GPAW and DALTON) or in the official version installed at the different PRACE Tier-0 Sites (CP2K, GROMACS, Quantum ESPRESSO). It is worth underlining that in some cases the scalability work allowed specific applications based on these restructured codes to be submitted to the regular access calls in PRACE (e.g. GPAW, Quantum ESPRESSO, GROMACS and CP2K).

For some applications the bottlenecks preventing petascaling were identified, analyzed and discussed with the owners of the codes (e.g. Open Foam and EC-Earth 3). In other cases more sustained efforts are required and has been scheduled to continue the work in PRACE-2IP (WP8 and WP9).

Task 7.2 issued an effective collaboration with Tasks 7.5 and 7.6 to afford different specific aspects, like hybrid parallelization or large dataset I/O optimization.

The cooperation with the structured communities continued and consolidated for a number of subject areas including IS-ENES, for meteo climate, MAPPER, for the communities involved in multi-scale methods, ScalaLife, for the Life Sciences community, and VERCE and EPOS for the seismological community.

The activity carried out by Task 7.2 for petascaling applications of interest for scientific communities is summarized in this document and has been documented in detail in 12 white papers from Task 7.2 and others 9 from Tasks 7.5 and 7.6 in collaboration with Task 7.2.

1 Introduction

Task 7.2 "*Application enabling with communities*" has the objective of developing relationships with important European applications communities and provide petascaling expertise to ensure that their key applications codes can effectively exploit Tier-0 systems [1]. Deliverable D7.2.1 issued in June 2011, documented the collaboration with scientific communities in the first year of activity of PRACE-1IP [2]. The document presented the methodology adopted to identify the scientific communities for cooperation and to select the application codes of interest for these communities.

During the first year of activity, Task 7.2 worked to fulfil its objectives by firstly contacting the scientific communities and selecting the application codes of interest, then starting to work with these communities in a long-term cooperation to petascale the codes.

Eleven application codes in the main domains of computational sciences were selected. For five of these applications the enabling work started in December 2010 and some very preliminary results have been already documented in Deliverable D7.2.1. For the remaining six applications the work started in May 2011, after the second selection process and the agreements with the scientific communities on the work plan to implement. The activity on the application codes, once concluded, allowed the related scientific communities to apply to the regular calls for access to Tier-0 resources (November 2011 and May 2012).

With respect to structured communities, cooperation has been established with the IS-ENES community, with the long-term objective to run the European climate models, of interest for this community, on PRACE Tier-0 systems. The work on EC-EARTH 3 codes was the occasion to consolidate the relationship between PRACE and the climate community represented by IS-ENES [5]. Further collaborations with other structured communities (i.e. MAPPER [6], ScalaLife [7], VERCE [8] and EPOS [9]) has been addressed during the current year, working on codes of interest for these communities (i.e. DALTON, GROMACS, SPECFEM3D).

The activity done in this last year of activity of PRACE 1IP has allowed to fully complete the petascaling process on the applications selected and to make the results available to enhance science by means of Tier-0 systems, demonstrating the utility of working closely with the scientific communities and the code developers. This approach has been clearly successful, also if it was not always easy to reach the expected results in terms of performance improvements with quite limited resources.

The work done in Task 7.2 has been documented in 12 different white papers, directly driven from task 7.2, plus other 9 prepared by Task 7.5 (cooperating for the aspects of new programming techniques and architectures for HPC) or Task 7.6 (efficient I/O and handling petascale data) in collaboration with Task 7.2.

This deliverable summarises these white papers. The technical details of interest for the scientific communities are reported in the white papers, which will be available on the PRACE website **http:// www.prace-ri.eu/white-papers**.

The deliverable is structured as follows: Section 2 covers the activity done with the material sciences, nano-sciences and Life sciences communities, on the codes GROMACS, Quantum ESPRESSO, GPAW and CP2K. Section 3 presents the work done on the codes DL-POLY and DALTON of interest for the computational chemistry community. The work on the code EUTERPE significant for the Plasma Physics community is presented in Section 4. Section 5 covers the cooperation with the IS-ENES community on the EC-EARTH 3 suite. The activity on the codes OpenFOAM and Code_Saturne representatives of the CFD and engineering communities are presented in Section 6. Section 7 presents an overview of the collaboration with the Communities. Finally some conclusions follow.

2 Material Science, Nanoscience and Life Science

In recent years, the use of computational modelling in Life Science and Material Science has enormously increased, becoming of crucial importance in many areas (nano-science, new-material engineering, protein simulations, docking etc.). This has been made possible by the development of theories into working equations which can be implemented into efficient codes. Many of these codes were originally designed for either scalar calculations or small parallel computing environments and need to be adapted in order to efficiently run on petascale systems. Furthermore the introduction of efficient levels of parallelization (MPI or hybrid MPI/OpenMP scheme) might in some cases require large modifications of the innermost structure of the original code, becoming therefore not an easy task to perform. In the context of PRACE-1P WP7.2, for the area of material sciences, nanosciences and life science, we have selected four widely used European codes to be optimized to petascale performances: GROMACS (life science), QuantumEspresso (material science), CP2K (life science and material science) and GPAW (material science). The activities were performed over a period ranging between 6 and 12 months, and were conducted in most cases in close collaborations with the developers' communities. Different optimization strategies were adopted for each code, involving modifications at different levels: from the optimization of the libraries and flags for the efficient installation on the PRACE machines up to the modification of the levels of parallelization inside the code. In some cases the optimization strategies were rather complex, requiring expertises and efforts from other tasks in WP7 (e.g. GROMACS, QuantumESPRESSO). For the four projects, the proposed goals were met and the finals results were reported in white papers.

2.1 GROMACS

GROMACS is a versatile package for performing classical molecular dynamics with hundreds to millions of particles. It is primarily designed for biochemical molecules, e.g. proteins, lipids, nucleic acids, but it is also widely used for simulating non-biological systems, e.g. polymers. The general goal of the project is to enable GROMACS to petascale simulations by identifying the main reasons for performance deterioration and designing dedicated actions to overcome them. The work on GROMACS in the scope of PRACE project has been performed in Task 7.2 by involving expertises from Task 7.5 (for the hybridization work) and Task 7.6 for data I/O optimization. In particular, the efforts from Task 7.2 and Task 7.5 resulted into a common white paper, namely "Performance Analysis and Petascaling Enabling of GROMACS" [10]. Results obtained from Task 7.6 effort has been reported in a separated white paper [10] and documented in [4]. In the following we report the main goals and achievement of the first white paper. The results presented in this paper have been combined with the achievements reached in collaboration with the ScalaLife project [7] working on the latest version of the GROMACS (version 4.6), released at the moment of the work on this white paper.

2.1.1 *Performance Analysis and Petascaling Enabling of Gromacs*

Supported by: Berk Hess (KTH), Erik Lindahl (KTH)

Collaborators: Fabio Affinito (CINECA), Andrew Emerson (CINECA), Leandar Litov (NCSA), Peicho Petkov (NCSA), Rossen Apostolov(KTH), Lilit Axner (KTH), Maria Francesca Iozzi (UiO).

The work consists of three independent tasks: (I) Optimization of GROMACS performance on BlueGene systems, (II) Parallel scaling of the OpenMP implementation, (III) Development of a new adaptive symplectic integration algorithm with variable step size.

Optimization of GROMACS performance on BlueGene systems

Most simulations require the calculation of long range interactions which consist of dispersion-type particle-particle (PP) interactions and the electrostatic forces obtained from the Particle Mesh Ewald (PME) method which makes extensive use of 3D FFT. On parallel computers these forces are obtained via a domain decomposition scheme which distributes the atomic interactions amongst the compute nodes assigned to the simulation. A further degree of flexibility is provided by the runtime environment of the BlueGene architecture which can be used to modify the default distribution of MPI ranks amongst the compute nodes. The aim of this work was to study these different scheme of the PP/PME partitioning on the BG/P system to obtain a set of optimal program or runtime parameters.

The different schemes have been tested on a membrane-like system containing nearly 2 million atoms, designed to show high parallel scalability. The results of the benchmark runs are given in the white paper and show that for a given number of cores the differences between the performances for the different partitioning schemes and BG/P mappings are small, perhaps at most around 10% for the highest core counts. Thus, for large projects it is worth investigating these settings but other factors maybe more important.

Parallel scaling of the OpenMP implementation

The parallel versions of GROMACS up to release v4.5 use pure MPI to handle the communications between the processor cores. As part of the ScalaLife project, the code developers have released a hybrid MPI-OpenMP version (v4.6) where multithreading has been applied to the PME calculations with the aim of increasing performance on common SMP nodes of parallel architectures. In cooperation with ScalaLife, Task 7.2 tested this hybrid version by performing benchmark runs on the PLX Intel cluster at the CINECA supercomputer centre in Italy (3288 cores Intel Westmere at 2.40 GHz) where nodes have 12 cores and are connected via an Infiniband network (about 5Gb/s speed). The tests, reported in the white paper, showed that there was in fact little difference in performance between the two versions of the code. This is in agreement with the documentation from ScalaLife which suggests that the hybrid version of the code is only likely to show improvements compared to the MPI-only version at high node counts or with slow interconnects: unfortunately on the PLX cluster only maximum of 40 nodes were available for tests. Additional tests were then run at the PDC Center for High Performance Computing on two different platforms, namely a Cray XE6 (Lindgren) and a AMD Opteron (Povel) sysytem. These clusters should provide a further test since more nodes are available and have different internal networks (2.7Gb/s Infiniband in the case of Povel and Cray Gemini for Lindgren). In addition, the performance of the new version 4.6 was compared to that of the previous 4.5.3 version.

Results are reported in Figure 1. We see that on Povel there is a significant improvement of the hybrid GROMACS with respect to the MPI-only code at high node counts and both versions are more performing than the earlier v4.5.3. For Lindgren, on the other hand, which has fairly fast internal network, there is no performance gain in using the hybrid code, even at high core counts. Thus, we see the hybrid MPI/OpenMP version 4.6 of GROMACS only really brings benefits with a large number of cores and when the internal network is fairly slow, as in the case of Povel.

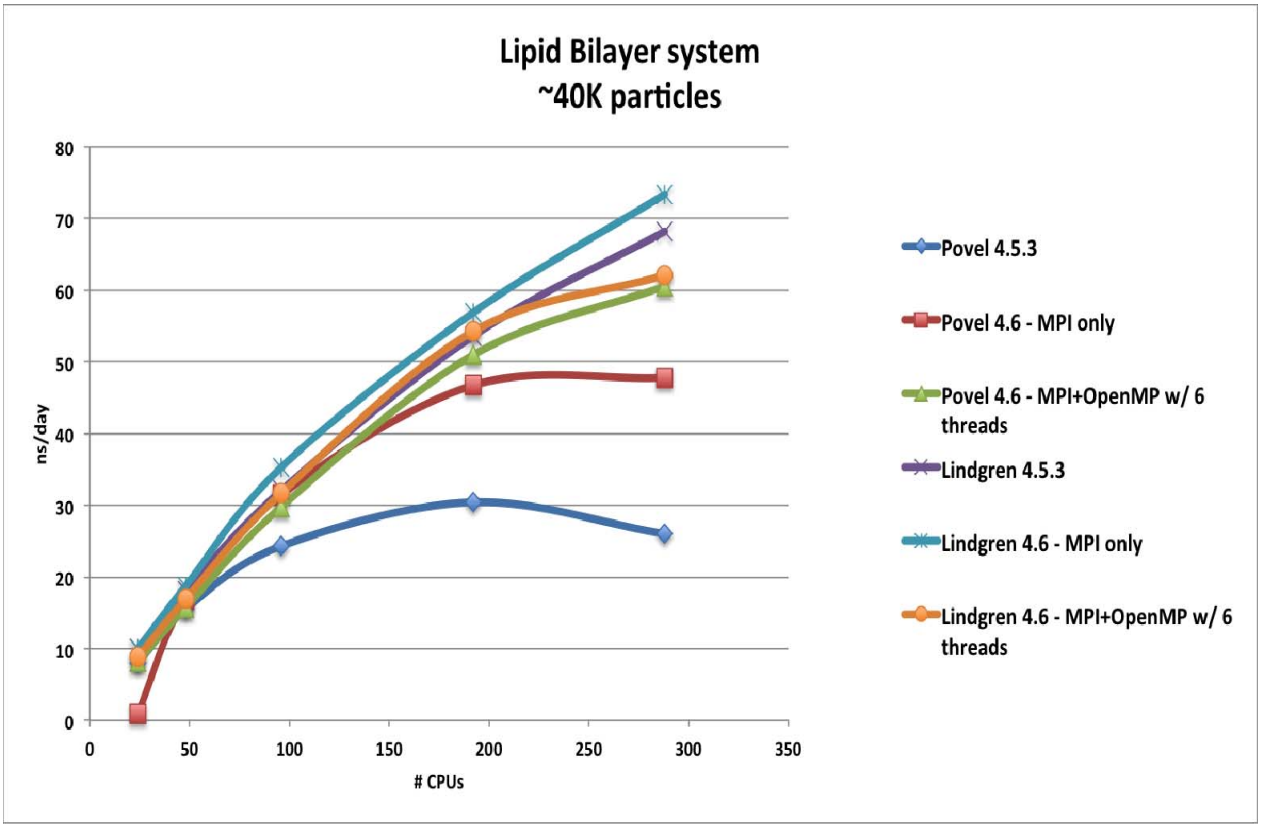

Figure 1: Scaling tests for the hybrid MPI/OpennMP version of GROMACS

Development of new adaptive symplectic integration algorithm with variable step size

For large atomic systems, one possibility for improving the performance of the MD simulations is to employ integration algorithms with variable step size for the calculation of full electrostatics. Limitations on the step size in this kind of integrators are severe, and are mostly due to stability rather than accuracy. The increasing size and complexity of the systems investigated stresses existing algorithms not only because of the exploding computation volumes but also because of the poor scalability with the number of the processors employed. We have studied the scalability and the work-load increase and distribution among the computing cores in the packages GROMACS (version 4.5.3), for the example of three test systems with increasing size (5x105, ~106 and ~2.2x106 atoms respectively), with the profiling tool SCALASCA and also by means of the GROMACS in-built tool g_tune_pme. We have also analyzed the stability and scalability of the existing integration algorithms with variable time-step implemented in GROMACS in order to define the sources of the instabilities (different kinds of resonances). Based on this analysis, a symplectic time reversible integration algorithm specially designed for Petascale biomolecular MD simulations is under development.

2.1.2 *Data I/O optimization in GROMACS*

The mechanism for data I/O provided by GROMACS does not scale well on MPP systems like BG/P. The data I/O bottleneck problems in GROMACS have been studied with the support of Task 7.6. A solution has been proposed using a Virtual Array structure. This activity is part of Task 7.6 and has been described in D 7.6 [4] and documented on the White Paper “Data I/O Optimization in GROMACS using the Global Arrays toolkit”, Supported by: Valentin Pavlov and Peicho Petkov (NCSA, Bulgaria), See [4].

2.2 Quantum Espresso

Quantum ESPRESSO (QE) is an integrated suite of computer codes for electronic-structure calculations and materials modelling based on density-functional theory (DFT), plane waves basis sets (PW) and pseudopotentials (PP). It is freely available and distributed as open-source software under the terms of the GNU General Public License (GPL). The present applicability of Quantum ESPRESSO ranges from simple electronic structure calculations to the most sophisticated theoretical spectroscopies such as Nuclear Magnetic Resonance (NMR), Electron Paramagnetic Resonance (EPR), Raman and Scanning Tunneling Microscopy, etc. The simulation tools implemented in Quantum ESPRESSO are used across a wide range of R&D applications. The relevance of this code has been highlighted by its adoption in a number of key research groups, as well as in industry.

The work aimed at improving the performance of important modules of the QE suite. In particular the goals of the submitted proposal were to enhance the scalability of the GIPAW module and the linear response code, needed for the NMR calculations and to improve the level of parallelization of the PWscf module. In addition, it was also decided to include the porting of the application to the GPGPU architecture among the goals. The work has been motivated by the existing strong partnership between PRACE partners and relevant members of the Quantum ESPRESSO users community and PRACE experts worked in close collaboration with developers to ensure that the Quantum ESPRESSO package can effectively exploit Tier-0 systems. Detailed description of the goals, optimization strategies and achievements are reported in the white paper: “Enabling of Quantum Espresso to petascale challenges” [10]. A brief summary is given below:

2.2.1 *Enabling of Quantum Espresso to petascale challenges*

Supported by: Ivan Girotto, Nicola Varini, Filippo Spiga (ICHEC, Ireland), Carlo Cavazzoni (CINECA, Italy)

Collaborators: Davide Ceresoli (CNR-STM, Italy), Layla Martin-Samos (CNR-IOM, Italy), Tommaso Gorni (University of Modena, Italy)

The work consists of three independent projects: (i) Scalability improvement of QE-GIPAW code and PWscf EXact eXchange (EXX) part; (ii) vdW interaction in Quantum ESPRESSO; (iii) GPU version of PWscf code. Performance analysis were performed on both PRACE Tier-0 systems and PRACE Tier-1 GPU systems.

I Scalability improvement of QE-GIPAW code and PWscf EXact eXchange

QE-GIPAW and the PWscf EXact eXchange part included in the Quantum ESPRESSO suite were showing poor scalability beyond a certain scale. To improve the scalability, additional levels of parallelism over electronic bands in both of these two codes have been implemented (namely two levels in QE-GIPAW and one in Pwscf Exact eXchange). The validation of the development activity was based on a real data set proposed by Prof. Stefano De Gironcoli for the study of cholesterol crystal equilibrium structures in human gallstones from first principles. Results from the modified QE-GIPAW code and Pwscf Exact eXchange are reported in Table 1 and Table 2, respectively.

In Table 1 the efficiency of the modified QE-GIPAW (number of bands > 1) are compared with the old version (number of bands = 1) for runs with different numbers of processors: while the band parallelization does not improve the efficiency of the new code when it runs on 64 processors, a significant improvement is observed already at 128 processors. Furthermore, the second level of parallelization allows the Green function to scale quite well up to over 3500 processors.

As shown in Table 2 the routine Vexx, that calculates the Exact-Exchange potential, scales perfectly by doubling the number of cores and the number of band groups. We measure an efficiency of around ~97% moving from 32768 to 65536 cores if we compare the time of execution of the single Vexx routine.

# Cores	# Threads	Number of bands	Elapsed Time (min)	#Green function (min)	Efficiency %
64	1	1	621.15	425.94	1.00
64	1	2	694.66	441.67	0.89
128	1	1	533.33	358.32	0.58
128	1	2	416.69	242.11	0.73
256	1	4	300.79	137.87	0.52
896	1	2	81.59	58.71	0.54
1792	2	2	51.82	37.60	0.42
3584	2	2	36.00	24.81	0.31

Table 1: QE-GIPAW benchmark data

# Cores	Number of bands	Vexx Time(s)	Elapsed Time(s)	Efficiency %
32768	64	417.18	523.45	1.00
65536	128	215.96	311.59	0.84

Table 2: EXX benchmark data

II vdW interaction in Quantum ESPRESSO

The rVV10 method is a non-local DFT approach that efficiently accounts for the local van der Waals interactions. Such interactions are in fact fundamental for a reliable simulation of large physical systems, of particular interest for soft matter and molecular biology. The method has been implemented in Quantum ESPRESSO. Tests show an improved accuracy of the rVV10 approach functional with respect to older implementation of vdW functionals. The new approach also keeps the same computational efficiency.

III GPU version of PWscf code

The PWscf code included into the Quantum ESPRESSO suite has been extended to fully exploit computing platform equipped with NVIDIA GPUs. Tests for the single node version were run on a medium size input benchmark for PWscf, (a gold surface of 112 atoms). Figure 2 compares wall-time and speed-up between single core, single CPU (6 cores) and also using one GPU. Adding one GPU to a single core PWscf achieves a 5.54x acceleration for gamma-point and 7.81x for k-point. The acceleration becomes 3.54x and 3.49x respectively if we consider the use of one GPU on top of a single CPU system. Results from tests of the distributed version are presented in Figure 3: The two graphs show wall-time and speed-up of parallel MPI+OpenMP versus MPI+OpenMP+CUDA. The data-set have been provided by Wei Zhang (Figure 3 top) and by Arrigo Calzolari (Figure 3 bottom). A detailed description of the test and comment of the results can be found in the white paper.

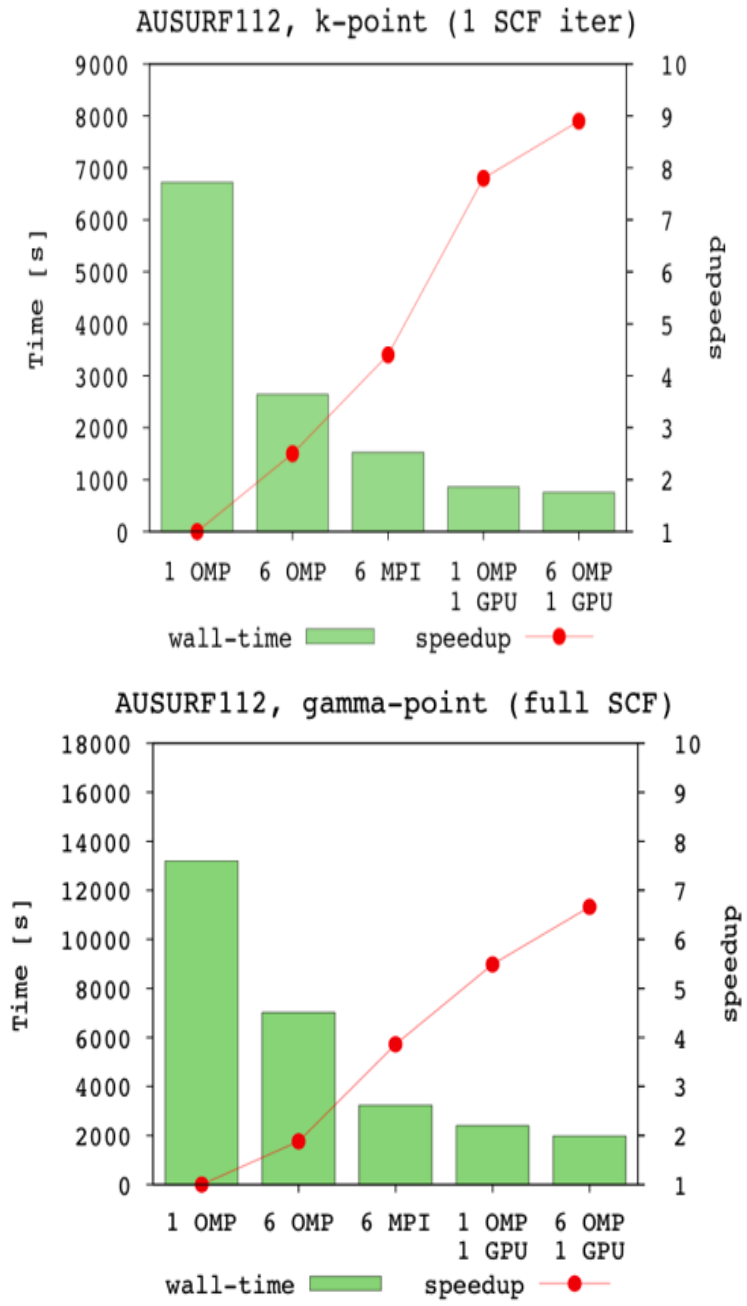

Figure 2: Best results running QE-PWscf on a six-core Intel Xeon X5650 and one Tesla C2050 single SCF iteration using K-points (top) and full SCF cycle using gamma-points (bottom)

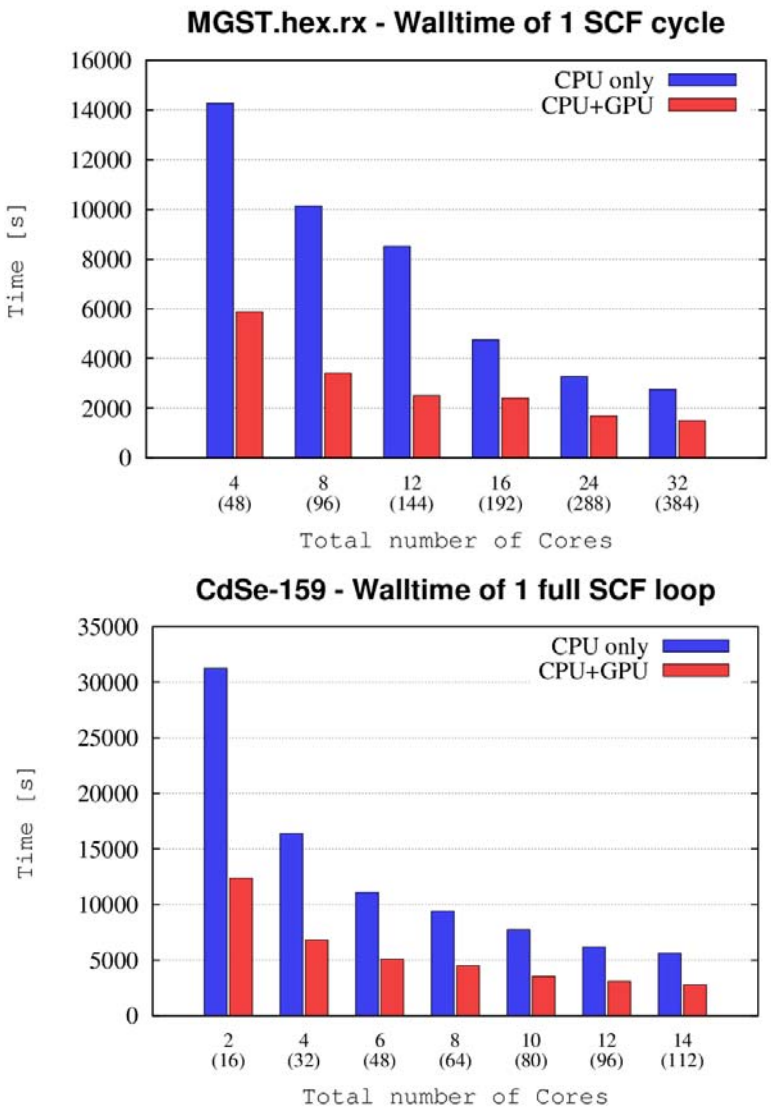

Figure 3: Wall-time and speed-up of parallel MPI+OMP versus MPI+OMP+CUDA data-set provided by Wei Zhang (top) and Arrigo Calzolari (bottom)

It is worth underlining that all the goals in the proposal have been met and the code has been significantly improved and is now upto petascale performances. Finally, we would like to point out that as an outcome of this work, De Gironcoli's research group (SISSA-Italy) was granted (millions of CPU-h) for accessing PRACE Tier-0 systems for a research project based on these software tools. More details are given in the white paper [10].

2.2.2 *Quantum ESPRESSO FFT Library Performance on PRACE Systems*

Supported by: Vladimir Slavnic, Milos Nikolic, Aleksandar Jovic, , Josip Jakic (IPB, Serbia)

Specific benchmarks have been performed on the FFT routines used in Quantum ESPRESSO. This activity was mainly done in the context of Task 7.5. Task 7.2 cooperated only in providing the Quantum ESPRESSO code and provide some consultancy on the use of FFT routines in Quantum ESPRESSO. The results of the activity are summarised in [3] and more extensively in the white paper by Pickles, Ivan Girotto, Peter Nash, Michael Lysaght, Milos Nikolic, Aleksandar Jovic, Josip Jakic, and Vladimir Slavnic. *"FFT Library Performance on PRACE Systems", 2012,* [10] produced as part of the activity in Task 7.5.

2.3 GPAW

GPAW is a versatile software package for first-principles simulations of nanostructures utilizing density-functional theory and time-dependent density-functional theory. The optimisation of GPAW for Tier-0 systems are reported in the white paper "Optimizing GPAW" [10] and summarised below.

2.3.1 *Optimising GPAW*

Supported by: Jussi Enkovaara, Martti Louhivuori (CSC, Finland), Vladimir Slavnic (IPB, Serbia), Petar Jovanovic (IPB), Mikael Rannar (SNIC)

The optimisation of GPAW for Tier-0 systems has been done in four distinct actions: I/O bottlenecks, optimisation of linear response TD-DFT implementation, hybrid OpenMP/MPI parallelisation schema, and optimisation of large dense matrix diagonalisations. The last activity was done in cooperation with Task 7.5 with the aim of incorporating ELPA (EigenvaLue solver for Petaflop Applications) in GPAW instead of ScalaPACK for Cholesky inversion routines. We now discuss each of these in turn:

The implementation based on the Python programming language introduces an I/O bottleneck during initialization which becomes serious when using thousands of CPU cores.

The reason is that when the Python interpreter starts-up, the Python code of all the used modules is read from the disk via import statements. The number of files read can be several hundreds, and all the processes read the same files. For each file to be read there are also several file open/close operations as well as directory operations when searching for the modules. With 65536 MPI tasks the start-up time can be over 48 minutes! We have modified the Python interpreter in such a way that only single process performs the I/O calls during imports, and MPI is used then for broadcasting the information to other process. In Blue Gene/P the initialization is improved by a factor of 100, *i.e.* start-up time is less than 30s with 65536 MPI tasks.

Recently, a memory bottleneck has been realized in linear response TD-DFT calculations. A large, so called Ω-matrix is replicated over all the MPI tasks, and in very large scale calculations there will not be enough memory to store the matrix on all processes. During this project, the implementation has been changed in such a way that each process stores only the part of the Ω -matrix it has calculated. Measurements of memory usage show that memory need decreases as expected, and that by increasing the number of CPUs for the electron-hole parallelization the memory requirement per CPU can now be kept constant.

Even though the current pure MPI implementation of GPAW performs well, it is expected that ever increasing number of cores within a CPU can make a hybrid MPI/shared memory implementation advantageous in near future. We have experimented with a hybrid implementation in Curie and in Blue Gene/P and Q systems. Initial results show no benefit from threading in Curie and Blue Gene/P, while in Blue Gene/Q small performance improvements can be obtained. An analysis of the communication patterns and performance suggests some further optimizations for hybrid implementation.

Large, dense matrix diagonalisations are expected to become a bottleneck in the future and the expectations are that when the number of electrons increases, the routines used from ScaLAPACK will not scale on par with the other parts of the code. We have investigated two alternatives to the current ScaLAPACK based implementation; Elemental and ELPA libraries.

Experiments indicate that the alternate libraries can improve the performance, and ELPA especially should be easily incorporated as it uses the same framework as ScaLAPACK.

Collaboration with scientific communities

GPAW is currently developed and used in several research groups in Europe and in the USA. During this project there has been active collaboration both with the user and developer communities.

The main way of collaboration has been via the mailing lists of GPAW users and developers. There has been also some developer teleconferences. Collaboration has been done also through visits to research groups in Finland, Denmark, and in the USA, and through participation to research workshops.

The work performed within this project has enabled three proposals to be made for PRACE production access, one in each of the PRACE 3rd, 4th, and 5th calls. The proposal in PRACE 3rd call was accepted and is currently running.

The modifications to the Python interpreter have been made publicly available in the Gitorious code repository http://gitorious.org/scalable-python and are available to the researchers who wish to use Python in massively parallel environments. All the modifications made to the GPAW code during the project have been made public via the GPAW's main source code repository https://trac.fysik.dtu.dk/projects/gpaw, and are thus available to all GPAW users.

2.4 CP2K

CP2K is a freely available (GPL) program, written in Fortran 95, for performing atomistic and molecular simulations of solid state, liquid, molecular and biological systems. CP2K is a popular and important code for materials science, life sciences and computational chemistry throughout Europe, and its supports the work of many research communities. One feature of CP2K which makes it a particularly important code with regards to use on Petascale systems in PRACE is its excellent scalability. Results showing scalability on tens of thousands of CPU cores are in the public domain and we demonstrated similar performance on the PRACE system Jugene in a previous PRACE project. This is achieved in part by a hybrid MPI/OpenMP parallelization approach, which allows the power of large numbers of CPU cores to be harnessed while reducing the impact of algorithms which scale less than linearly with the number of MPI processes used. In some cases such as Hartree-Fock Exchange (HFX) calculations, using OpenMP is required to allow each process to access the entire memory of a compute node in order to store tables of commonly re-used integrals, which provides excellent performance. In addition, the hybrid MPI/OpenMP version maps well to the fat-node architecture of modern multi-core node supercomputers such as Curie and Hermit, where MPI can be used between NUMA regions (and compute nodes), and OpenMP within a single NUMA region.

To best support the user communities of CP2K, this PRACE project ported and tested CP2K on the Curie PRACE system. In addition, we have improved and extended the implementation of OpenMP within the code, focusing on several areas where we believed performance or scalability was an issue. The activity done is documented on the white paper *CP2K – scalable atomistic simulation for the PRACE community* and a brief overview is reported below:

2.4.1 *CP2K - scalable atomistic simulation for the PRACE community*

Supported by: Iain Bethune, Adam Carter, Kevin Stratford (EPCC, UK), Paschalis Korosoglou (AUTH, Greece)

I Porting CP2K to Curie

Initially three different configurations (compilers and math libraries) have been considered for compiling CP2K on the Curie Tier-0 system. Upon compilation, from each one of these 3 configurations, 3 versions of the CP2K were obtained, namely **sopt** (for the serial version), **popt** (for the MPI only version) and **psmp** (for the hybrid MPI+OpenMP version), resulting in a total of 9 executables. Using the regression test suite provided with the CP2K source code we have been able to test these 9 versions of the application. By running the regression tests, several more problems with the GNU compiled versions linked with the custom build linear algebra libraries (BLAS, LAPACK etc) were identified thus further investigations using this configuration of CP2K were dropped. Furthermore, the tests using the **psmp** version of CP2K build with the Intel Compiler Suite have not been run to completion since for several test files the execution stalled. All the observed errors were fixed and the 5 remaining configurations were then tested: the results of the regression tests are shown in Figure 4.

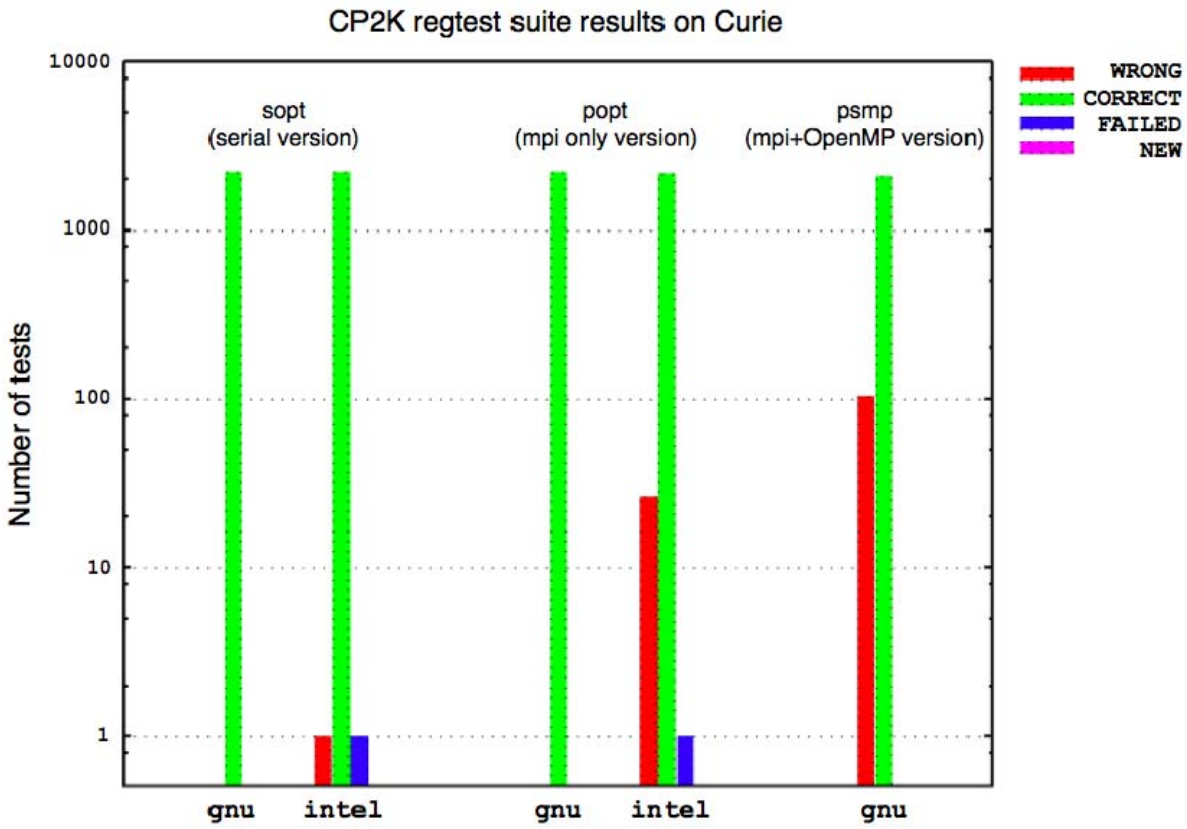

Figure 4. Final CP2K regression test suite results.

OpenMP parallelization of Exchange-Correlation Functionals

Prior work by Bethune [11] [12] has already introduced or improved OpenMP parallelisation in various areas of CP2K. However, it has also identified several other areas, which take up significant amounts of time for certain calculations which are not OpenMP parallelized, or which were in need of improvement. The exchange-correlation functionals (XC), which provide a relevant part of the energy in a density functional calculation, require such a parallelization. After surveying the current state of OpenMP for each XC functionals (there are 26 XCs in CP2K), we fully parallelized all the functionals using OpenMP, and validated the results of these changes by running the CP2K regression test suite. The OpenMP performance obtained was overall very good: on a Cray XE6 with 24 cores per node (two 12-core processors), 95% efficiency within a single NUMA region, and up to 92% efficiency across an entire node has been obtained.

Realspace grid operations

Another key element within CP2K is the use of real space grids, which are a key step in the transformation from the plane-wave basis (stored as complex values on plane wave grids), to the atom-centred Gaussian basis (stored as coefficients in a sparse matrix). In the routine calculate_rho_elec, Gaussian basis functions are received in matrix form and are written to the real space grids in parallel by the team of OpenMP threads. The existing algorithm had been observed to be costly as it does not scale well with increasing numbers of OpenMP threads. It was proposed to replace the reduction algorithm with a tree-based one which would minimize the total volume of data copied, and in particular reduce copies from one NUMA region to another. Three new tree-reduction algorithms were implemented (see the white paper for more details) and compared with the existing implementation. (see Table 3). Two of them were a significant improvement over the original implementation. Furthermore by carrying detailed profiling during the development of the new grid reduction algorithm, a few performance bottlenecks were found and resolved (see the white paper for more details).

Core Hamiltonian calculations

The calculation of the Core Hamiltonian matrix can take a significant amount of time for certain calculations, particularly those with large basis sets, or when using OpenMP since there was no existing OpenMP parallelization in this region of the code. We have implemented several OpenMP parallelization schemes but the obtained speed-up results were always disappointing. Alternative parallelisation strategies will be investigated in the future.

Number of Threads	1	2	4	6	8	12	16	24
Version 0	0.0000019	0.0028	0.0040	0.0059	0.0069	0.0082	0.011	0.013
Version 1	0.0000027	0.0026	0.0037	0.0052	0.0069	0.0097	0.011	0.013
Version 2	0.0000028	0.0029	0.0038	0.0058	0.0062	0.0072	0.0071	0.0083
Version 3	0.0000027	0.0026	0.0039	0.0061	0.0062	0.0073	0.0073	0.0084

Table 3: Comparison of time taken (seconds) for parallel grid reduction algorithms

2.4.2 *Other activities related to CP2K*

In Task 7.6 a specific activity was carried out for the fragment Orbital method (FMO) for Highly parallelised Quantum chemical calculations in CP2K. This activity was supported by Peicho Petkov, Petko Petkov, Georgi Vayssilov and Stoyan Markov (NCSA, Bulgaria) was mainly conducted in Task 7.6, with a light support from Task 7.2. The activity is reported in [4].

In Task 7.5 an activity to improve sparse matrix multiplication (SpMxM) operations in CP2K has been done. This activity supported by Cevdet Aykanat, Kadir Akbudak and Ata Türk (Bilkent) is reported in a specific white paper as part of the activity in Task 7.5 and documented in [3].

3 Computational Chemistry

Since there is considerable overlap between the chemistry, biology and materials sections, the chemistry effort focussed on types of computational chemistry software not covered in the other subtasks. This included molecular quantum chemistry (DALTON) and a classical simulation code with a focus on materials simulation (DL_POLY).

3.1 DL_POLY

DL_POLY is a general purpose molecular dynamics code, supporting a very extensive range of force-field options. It is owned and developed by the STFC, with work led from the Laboratory by Ilian Todorov. Most of the user community are in the materials science area, with a smaller number in the biomolecular simulation area. The central MD core of the domain-decomposed versions of DL_POLY are already highly scalable; for example DL_POLY_3 performance has shown good hard scaling for a 14 million Gd pyrochlore system (full Ewald electrostatics evaluation) on up to 16,000 cores of BG/L at Julich [13]. DL_POLY_4 (the current target code) has shown excellent weak scaling on up to 84,000 cores (500,000,000 Fe particles) of HECToR (http://www.hector.ac.uk/). This version also offers a scalable parallel I/O in both ASCII and netCDF.

Current performance bottlenecks are related to the weakness of the MPI latency and bandwidth across fatter nodes and the non-stopping trend of HPC vendors offering solutions of higher core density nodes and of inhomogeneous architecture (+GPU), all with less memory per core. As the Flops power (as well as RAM) per core decreases but the core count offering of the HPC architectures increases users start to use DL_POLY_4 in regimes (< 1000 particles per core) where the parallel performance is dominated by the pre-factors of algorithms and overheads in the communications. Thus a solution of hybridizing the MPI with OpenMP threads and further work-offloading to GPUs or alternatively redefining the CUDA implementation into OpenCL are the naturally ways to bring back work in well balanced regimes and push performance further to the levels expected for a petascale application.

Work within PRACE has primarily taken place as a collaboration between Task 7.2 and 7.5 (mainly concerning accelerators, but there has also been some work on the use of numerical libraries). The main activity in collaboration with Task 7.5 is described below and reported in the white paper *Benchmarking and analysis of DL-POLY 4 on GPU Clusters* and documented in [3].

3.1.1 *Benchmarking and analysis of DL_POLY_4 on GPU clusters*

Supported by: Mariusz Uchronski, Agnieszka Kwiecien, Marcin Gabarowski (PSNC-WCNS, Poland), Peter Nash, Michael Lysaght (ICHEC, Ireland), Ilian Todorov (STFC, UK).

The major achievement of this work is the synchronisation and subsequently the successful benchmarking on ICHEC's Stoney GPU cluster of the 'CUDA+OpenMP' port of DL_POLY with the latest changes of the vanilla MPI code. Figure 5 shows a comparison of the GPU code with the MPI, indicating the reduction of compute time obtained from the use of the accelerators.

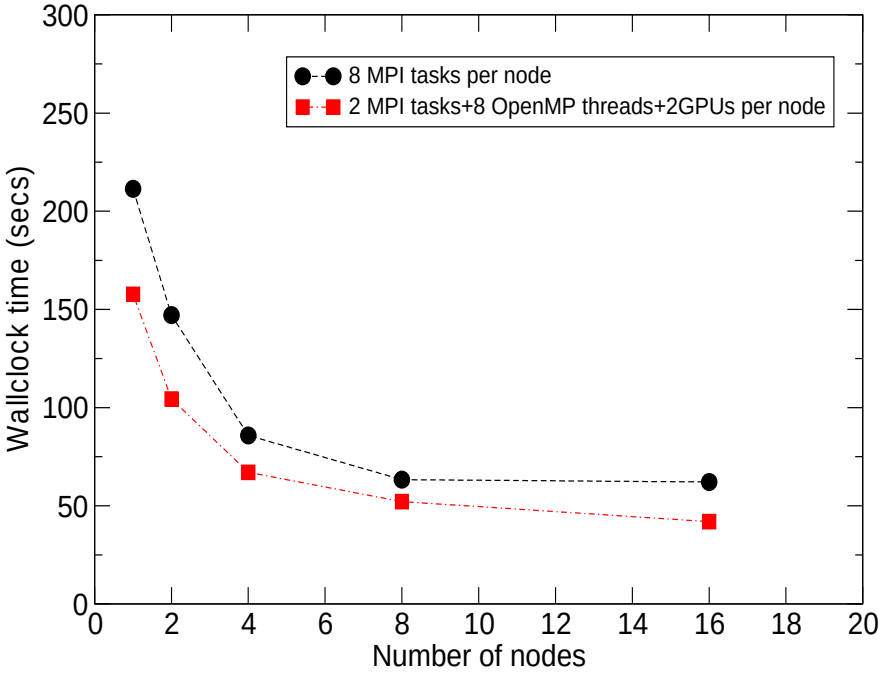

Figure 5: Performance comparison for pure MPI and OpenMP/GPU versions of DL_POLY.

The success of porting DL_POLY to hybrid architectures using the CUDA framework inspired the initiation of OpenCL porting of the CUDA DL_POLY version to extend access to more general accelerator-based architectures. The OpenCL work proved to be more laborious in comparison to the one involving the use of CUDA. However, it is worth emphasising that OpenCL is not tied to a particular vendor or even to a particular accelerator-based architecture.

The GPU-enabled port showed a marked performance advantage over the vanilla version for a small problem size in cases even where the pure MPI code has shown good scaling. The replacement of the DaFT (3DFFT) library of DL_POLY_4 with the GPU-enabled library, DiGPUFFT, did not offer any performance benefits for the grid sizes of interest to molecular modellers at the present. However, an impressive speedup was observed when using the single-CPU-core-single-GPU CUFFT library instead. This however could only be of benefit within an MPI 'gather-scatter' strategy on small-scale GPU clusters – an approach which could be investigated further.

The flexibility of the DL_POLY_OpenCL port was demonstrated by benchmarking it on different accelerator-based architectures including AMD's Radeon cards along with more familiar multicore CPUs. The performance results indicated that the DL_POLY_OpenCL code was slower than DL_POLY_CUDA (see Figure 6 for the comparison of a number of kernels of the SHAKE algorithm).

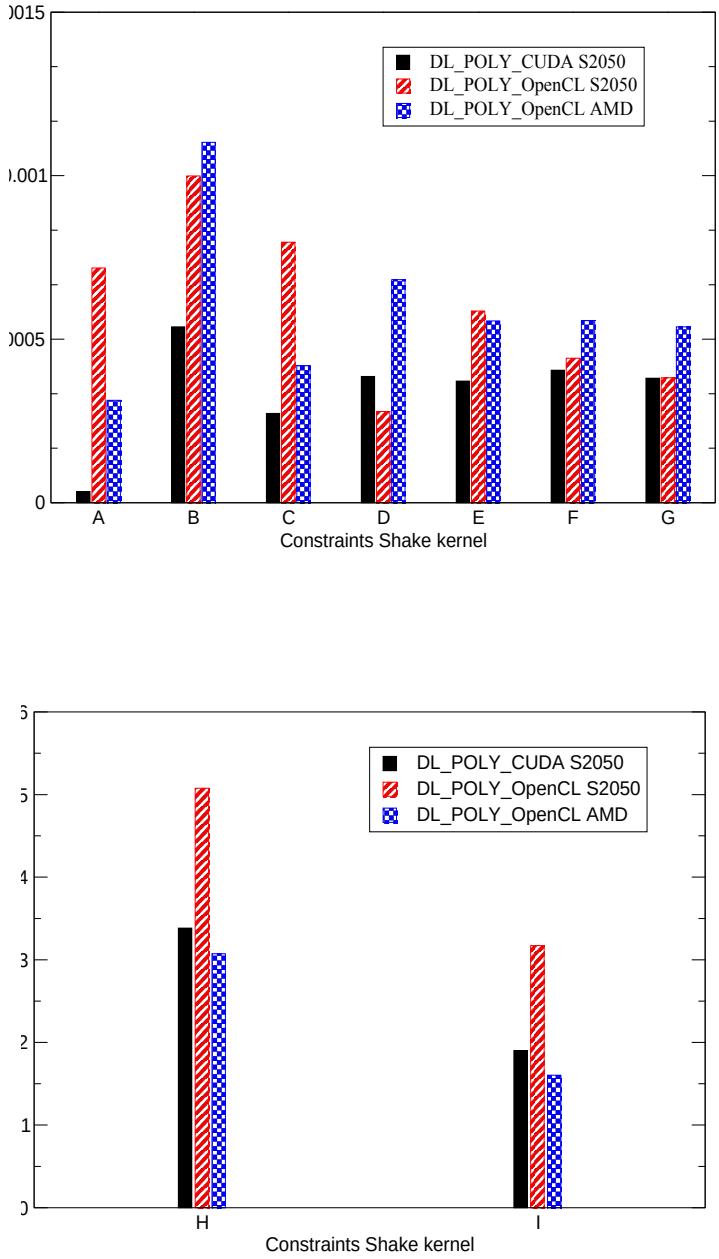

Figure 6: Performance comparison of DL_POLY_CUDA (top) and DL_POLY_OpenCL (bottom) The tests were runned on different systems. Key for Constraints Shake kernels: A=read, B=write, C=gather_dv_scatter_hs, D=gather_hs_scatter_dv, E=k1_th, F=k1_bh, G=correct_positions, H=initialize, I=insta.

It should be noted that there is room for further optimisation of the DL_POLY_OpenCL code to achieve better performance for particular accelerators. Further OpenCL porting of the CUDA+OpenMP features will be advantageous and the WCNS team will carry this out in PRACE-2IP WP12.2.

3.1.2 *Other activity analysis of DL_POLY_4 and collaboration with the Community*

In addition, the activity under Task 7.2 included two collaborations with organisations that were not funded under PRACE, but are contributing to the petascaling of the DL_POLY code. This work has not yet generated any major results that can be reported in a white paper but the collaborations have started work. The projects are:

1. A project (funded by the UK EPSRC) which will involve re-engineering all numerically intensive routines of the package in order to exploit the fine grain parallelism and shared memory structures of modern multi-core node architectures (openMP hybridisation). The partner in this is NAG Ltd., based in Oxford.
2. A collaboration with the "Molecular Systems" Simulation Lab at FZ-Julich to test and evaluate the FMM of their library ScaFaCoS (http://www2.fz-juelich.de/jsc/scafacos/) within the DL_POLY code as an alternative of the DL_POLY DaFT library used in the smoothed particle mesh Ewald evaluation.

Both projects use non-PRACE funding sources and the activity is done in strict cooperation with PRACE Task 7.2.

Work on the OpenMP hybridisation started in November 2011 and is planned to deliver a final product in November 2012. Preliminary data and benchmarking on HECToR (http://www.hector.ac.uk/) indicate that the critical regime of small number of particles per core in pure MPI parallelisation can be up to 8 fold improved with the hybridisation. The current prototype was also successfully compiled and run on a MIC card without any modification. A release of the first iteration of this work is planned for July 2012.

Ivo Kabadshow from Julich visited Daresbury Laboratory in February 2012 and spent a week with Ilain Todorov on learning DL_POLY and designing a call to FMM in ScaFaCoS that complements the DL_POLY data structures. The preliminary results show that the most expensive operation is adapting the data from DL_POLY to ScaFaCoS format which renders this expensive for conventional system sizes over large processor counts. However, the investigation continues.

It was not possible, since the organisations are working involving partners outside PRACE, to guarantee delivery within the timescale of PRACE-1IP but it is still expected that these activities will deliver enhanced petascaling of DL_POLY on PRACE systems over the next 12 months.

3.2 DALTON

The DALTON project brought together a number of institutions with plans to work on two distinct versions of the DALTON package. The work included both petascaling optimisation performance analysis and support for QM/MM calculations in the ChemShell package.

3.2.1 *Petascaling and Performance Analysis of Dalton on Different Platforms*

Supported by: Simen Reine,Thomas Kjærgaard, Trygve Helgaker (UoO, Norway), Olav Vahtras, Zilvinas Rinkevicius, Bogdan Frecus (KTH, Sweden), Thomas W. Keal, Andrew Sunderland, Paul Sherwood (STFC, UK), Michael Schliephake, Xavier Aguilar, Lilit Axner (PDC, Sweden), Maria Francesca Iozzi, Ole Widar Saastad (UoO, Norway), Judit Gimenez (BSC, Spain)

The activities have been organized into four tasks within the PRACE activity, and we will provide a short summary of each:

1) *Analysis of the current status of the Dalton code and identification of bottlenecks, implementation of several performance improvement of Dalton QMM and first attempt of hybrid parallelization;*

The optimizations in the application code focused on the part for the calculation of the molecular properties, that is, the calculation of the response function. These function values

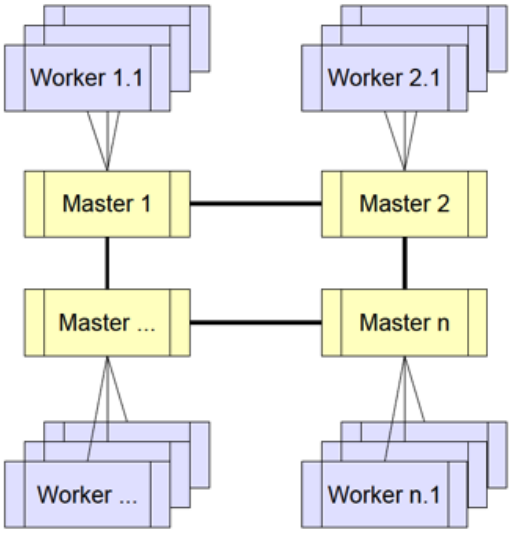

Figure 7: "Team of masters" communication model in Dalton

consist of several contributions from different modelling approaches like quantum mechanics or density functional theory.

The bottleneck in the communication between master and worker processes has been removed through the introduction of a "team of masters" (see Figure 7). Before the optimization, all workers took part in the computation of every contribution to the response function although all these different contributions can be calculated independently and in parallel.

Now after the refactoring, the available set of workers is partitioned into groups that need to compute only one contribution. Every group has a size adapted to the computational work needed for the calculation and its own master that collects the results. A substantial ease of the master's workload can be reached in that way. Smaller worker groups dealing with only one task lead to shorter waiting times for point to point communications as well as less complex and shorter collective communications. Furthermore, multiple workers can communicate at the same time realizing better use of interconnect and a higher degree of parallelism. The collected contributions are finally exchanged between the masters, which also compute the response function and initiate a new iteration until the convergence is reached.

The original program version scales until 512 cores (see Table 4) and provides a parallel efficiency around 50% (see Figure 8 and

Figure **9**). Almost no speedup can be gained beyond this number of nodes. The optimized version, on the other hand, now runs up to 1024 cores giving a parallel efficiency around 50% or more. We found that our optimized version has increased its average computation time from the initial value of 70% for workers to 86%, reducing its average MPI time from around 30% to 14%.

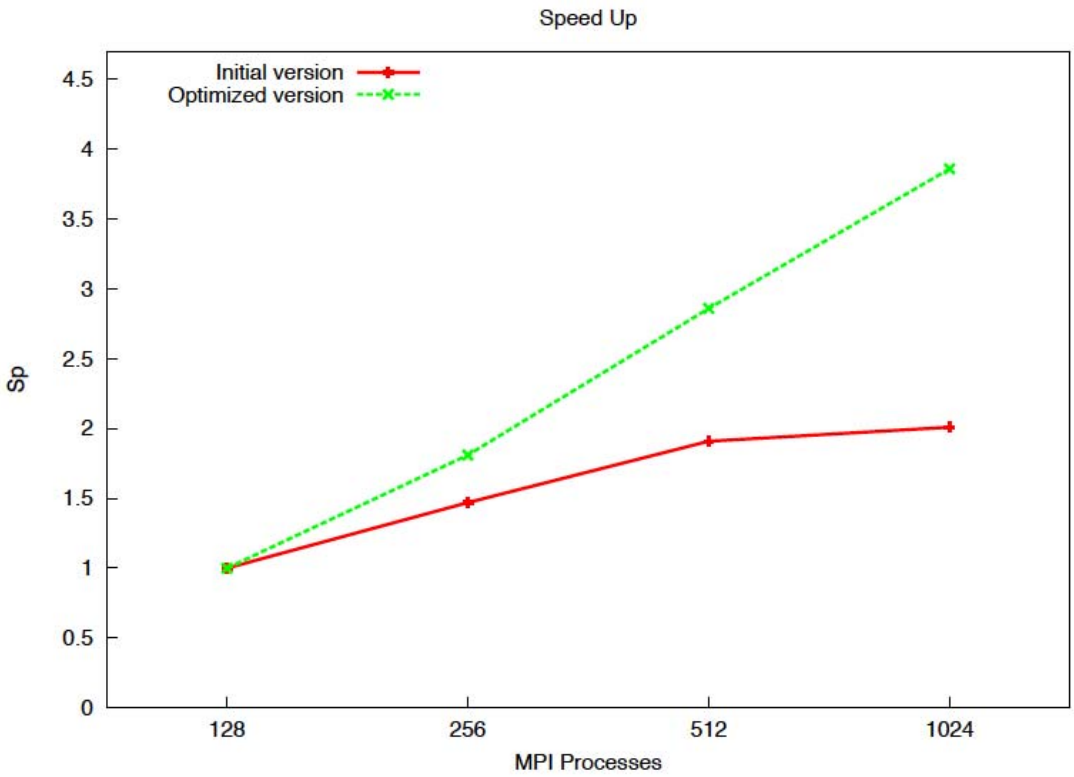

Figure 8: Parallel speedup of Dalton

2) *Implementation of MPI integral components into LSDALTON, improvements of optimization and scalability, interface of matrix operations to PBLAS and ScaLAPACK numerical library routines;*

The MPI parallelization approach adopted in the LSDALTON code can be categorized into two main categories, which naturally follows the underlying quantum-chemical code structure that basically boils down to the evaluation of various molecular integral components and a number of matrix operations. Efficient integral evaluation requires highly specialized routines and as a result specially tailored parallelization strategies, whereas the matrix operations are obvious candidates to parallelize using externally developed libraries.

For the analytical coulomb integral evaluation, we have developed our own MPI strategy where we divide the computations into *tasks* that rely on a priori predictions of the integral-evaluation cost. Clearly our approach relies heavily on this partitioning scheme. The challenges with our scheme are twofold, namely to 1) give good time-estimates and 2) make a reasonable partitioning based on the time estimates. As such a significant amount of our time in this project has been devoted to ensure that both these two challenges have been addressed.

The numerical quadrature used in quantum chemistry is based on a superposition of atomic grid points, and for efficient evaluation batches of grid points are evaluated simultaneously. The parallelization strategy adopted here is for each MPI node to evaluate a predetermined subset of these batches and the subsequent looping over these subsets is again parallelized using OpenMP.

In addition to the integral-evaluation components, DFT relies heavily on linear algebra: for example for the energy optimization, molecular responses to external perturbations, geometry optimization routines and more. Although typically the matrix operations are not the time-limiting factor of the LSDALTON code (for serial calculations) the computational time are in many cases still significant, especially for large molecular systems (comprising tens of

thousands of basis functions). In this project we have interface our matrix operations to the PBLAS and ScaLAPACK numerical library routines, with assistance from STFC (the Science and Technology Facilities Council) in the UK.

We have also re-designed the integral code to enable a suitable memory-distributed format also for the analytical integral evaluation. This scheme relies on transformation between our memory distributed format and the ScaLAPACK format. These transformation steps have been implemented during the final phases of this project, but are not yet fully tested.

To determine the performance and scalability of the LSDALTON program we have chosen to look at the valinomycine (168 atoms) and insulin molecule (787 atoms, 7604 basis functions). The examples given below are for insulin. The two molecules represent typical biochemical systems that we would like to investigate using the LSDALTON program, and for both molecules we have investigated the scalability of the different components of the code by performing typical Kohn-Sham energy optimization iterations using the pure DFT functional BLYP. All calculations have been conducted on the so-called 'fat' nodes on the Curie supercomputer, where each node consists of 4 eight-core x86-64 CPUs

# cores	*regJ*	*xc*	*Dens*	*Total*	*Speed-up*	*Ideal*	*Efficiency*
256	3175	32	93	3300	1.0	1	100
512	1682	20	68	1770	1.9	2	93
1024	1004	15	49	1068	3.1	4	77
2048	599	12	131	742	4.4	8	56

Table 4: Hybrid MPI/OpenMP for insulin using BLYP/cc-pVDZ with no density-fitting (Timings in seconds).

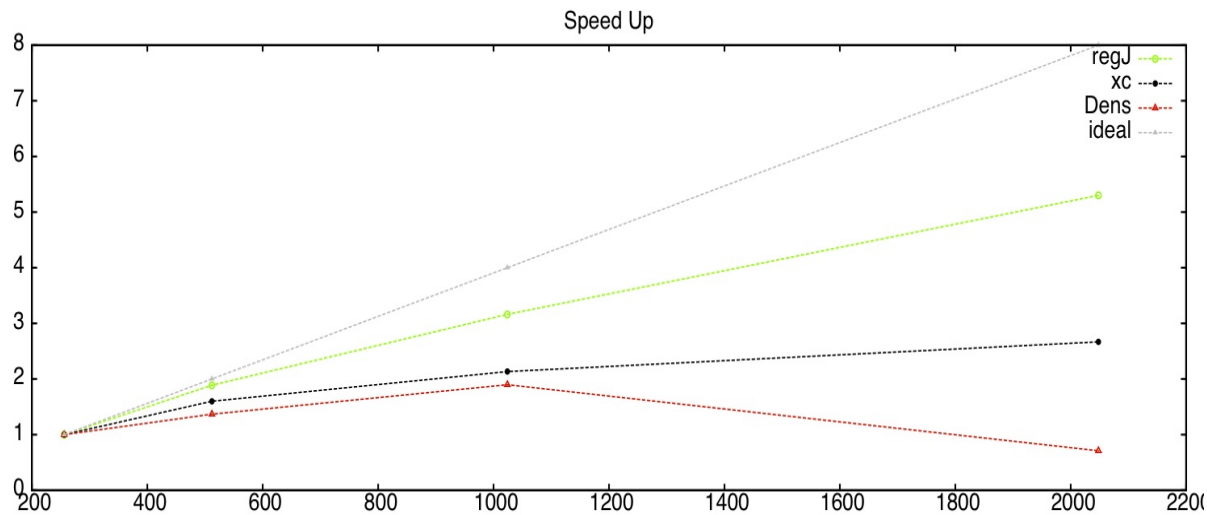

Figure 9: Speedup of Dalton for different components as a function of the number of cores

3) *Interfacing the DALTON and LSDALTON QM codes to the ChemShell QM/MM package and benchmarking of QM/MM calculations using this approach;*

A QM/MM scheme was successfully implemented for both. The benchmark system was a QM/MM model of the enzyme *p*-hydroxybenzoate hydroxylase (PHBH). The model consists of the protein, the oxidized form of the co-factor flavin adenine dinucleotide, substrate *p*-hydroxybenzoate and a surrounding shell of solvent molecules, giving a total of 22,748 atoms. Three QM regions were considered in order to test how the MPI scaling changes with system size. Region A consisted of the substrate, part of the co-factor and a neighboring amino acid

residue (no. 293). Region B consisted of region A plus the rest of the co-factor. Region C consisted of region B plus several more amino acid residues (nos. 294-298).

As the exact exchange routines had not yet been optimized in the latest LSDALTON build, the benchmark calculations were performed with the BLYP GGA functional. In all cases the TZVP basis set ("Turbomole-TZVP") and the df-def2 auxiliary basis for density fitting was used.

It was not possible to run baseline single MPI process benchmarks for all three regions due to the excessive wall time required, and therefore we compare parallel efficiency indirectly using "speedup compared to running with half the number of MPI processes" which we denote "S(N/2)" in the table. If ChemShell and LSDALTON were ideally parallelized S (N/2) would be 2 for every doubling of the number of MPI processes.

The S (N/2) measure is quite volatile, but does give some indication of the relative parallel scaling behavior for each QM region. Because the LSDALTON calculation is the dominant factor in the scaling results, the S (N/2) results for LSDALTON are almost identical to S (N/2) for ChemShell as a whole. In Figure 10 we plot the LSDALTON S (N/2) scaling for all three QM regions.

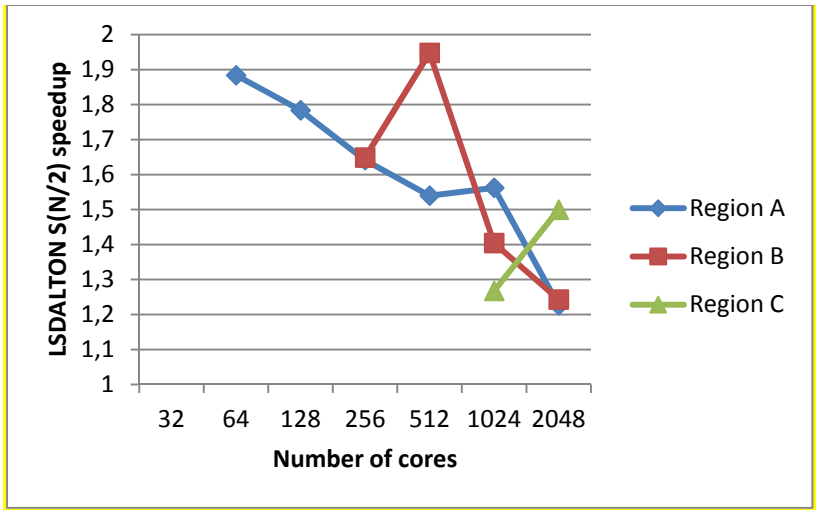

Figure 10: LSDALTON S (N/2) scaling for the 3 QM regions

4) *Analysis of the impact of Dalton QMM system components with Dimemas. Part of the results reported here have been achieved through the collaboration with ScalaLife project.*

The goal of the parametric study has been to evaluate the sensitivity to different architectural parameters. We have focused on 3 parameters that we considered may be the more relevant. The two first parameters, network latency and bandwidth will allow us to evaluate the network requirements of the code, and estimate the penalty if we move to a machine with a low latency network and to measure the potential benefit from moving to a better network. The third parameter we call the CPU ratio and corresponds to the speed-up when using more processors in parallel.

The results are shown in Figure 11 – 13 and correspond to the variations of two parameters while keeping the third one constant. Figure 11 corresponds to the scenario with a fixed latency of 4 microseconds that is quite similar to the current latency on the machine used to obtain the trace file. We can see how both the CPU ration and the bandwidth have an impact on the achieved speed-up. With networks slower than 512MB/s there would be no benefits on the code even if the processor is 64 times faster than the current machine. On the other hand, to see significant benefits on the bandwidth at least an 8 times faster CPU is required. This

last observation indicates the importance of optimizing or parallelizing the current sequential computations to benefit from future faster networks.

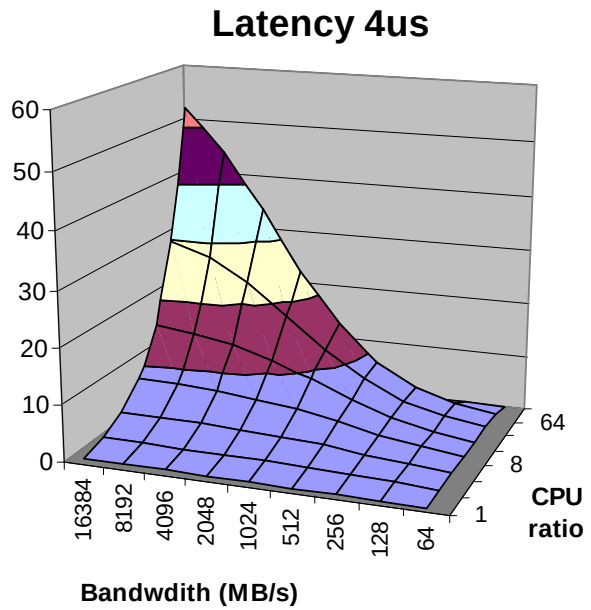

Figure 11: Fixed latency at the variations of 2 parameters keeping the third one constant

Figure 12 corresponds to the scenario with an improved processor 8 times faster than the one used for obtaining the trace file; this scenario focuses on the impact of improving the computing time with no changes on the network requirements. The first observation is that while with the previous scenario we were able to obtain a speed-up up to 50 times faster than the current execution, here we obtain only a factor of 8. Again this is indicating a problem with the computing regions. The fact that all the combinations have a speed-up over the reference execution is because of the 8 times faster processor. We can see that in this case there is almost no impact from variations of the latency. Only when the bandwidth is greater than 4GB/s can we start to see a very small benefit from reducing the latency. We can see the benefits of increasing the network bandwidth in this scenario but also we see that with 16GB/s it is starting to reach the plateau.

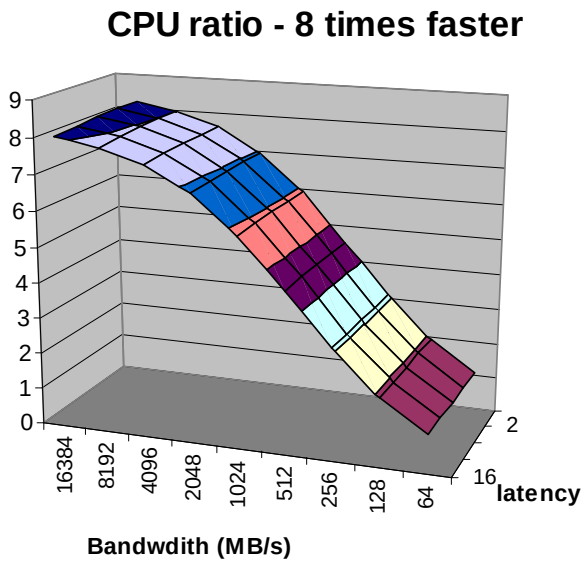

Figure 12: Impact of improving the computing time with no changes on the network requirements

Figure 13 corresponds to the scenario with a fixed bandwidth of 1GB/s that is quite similar to the current bandwidth on the machine used to obtain the trace file. Again we can see almost a plain behavior when changing the latency with the fixed bandwidth except for the cases of 32 and 64 times faster CPU where a latency of 16 microseconds reduces the performance. In this scenario, the improvement of the processor speed goes up to 16 indicating that may be important to consider reducing the network requirements if the current serial codes are optimized for instance porting the code to accelerators.

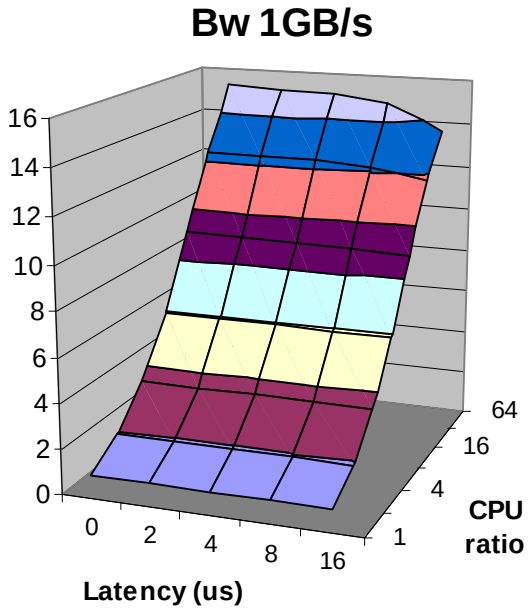

Figure 13: Changing the latency with respect to a fixed bandwidth.

The scenario with a fixed bandwidth of 1GB/s is quite similar to the current bandwidth on the machine used to obtain the trace file.

3.2.2 *Conclusions*

The work to reengineer the software architecture of DALTON allowed the use of high-level parallelism that is part of the numerical model in the program. The implementation used so far is for the parallel computations to be employed for the calculation of each contribution to the response function. However, the scaling of these single contributions is limited at the moment. The modification of the master-worker design pattern to a team of masters allows several contributions to be calculated at the same time and increases the scalability of the application accordingly.

For the LSDALTON code we have demonstrated good scalability up to 2048 cores for pure DFT calculations, through the development of MPI integral components, utilizing a newly developed task partitioning scheme, and through the interface to the PBLAS and ScaLAPACK numerical library routines. We have thus established the main code framework and strategies to exploit parallelism at all levels of the code, and thanks to the PRACE community we have learned useful tools for future developments. The results presented clearly show that we are well on our way toward peta-scaling capabilities.

The LSDALTON and DALTON QM codes have also been successfully interfaced to the ChemShell QM/MM package with both binary and directly-linked interfaces. Through ChemShell, the Dalton codes have access to a range of MM approaches and supporting functionality to help enable flexible QM/MM modeling on PRACE systems. The LSDALTON interface also supports shared MPI communicators for advanced task-farming

parallelization techniques. Benchmark QM/MM calculations using ChemShell/LSDALTON achieve good scaling of up to at least 1024 cores.

4 Astrophysics, Cosmology, High Energy Physics and Plasma Physics

The communities of Astrophysics, Cosmology, High Energy Physics and Plasma Physics widely use computational methods for their research activities, and, as a consequence, a consistent part of the resources available in PRACE Tier-0 systems are consumed from these groups. These communities in general develop by themselves new algorithms and techniques to improve the performance and scalability of their codes, and exploit the new Tier-0 architectures. As reported in D7.2.1 [2], the requests which arrived from these communities to petascale codes of interest for them was quite limited. The process issued in Task 7.2 for the selection of codes to petascale, identified EUTERPE as the representative of these communities.

4.1 EUTERPE

EUTERPE is a gyrokinetic particle-in-cell (PIC) code for the global linear and non-linear simulations of fusion plasma instabilities in three-dimensional geometries, in particular in tokamaks and stellarators. The code provides good results both in linear and non-linear simulations of ion temperature gradient (ITG) instabilities carried out in screw-pinch geometry. It is at the forefront of plasma simulations and requires huge amounts of computational resources.

The petascaling activity involving EUTERPE was mainly focused in the following directions: introduction of the hybrid parallelisation schema (MPI+OpenMP) to improve scalability, porting to heterogeneous architectures (GPU) and I/O performance.

The activity done has been reported in two different white papers: "*Evaluating application I/O optimisation by forwarding layers*" (by Jan Christian Meyer, Jorn Amundsen, Xavier Saez) and "*Euterpe*" (by Xavier Sáez, Taner Akgün, Edilberto Sánchez). The first one analyses the I/O aspects and, being part of the activity in cooperation between Task 7.6 and Task 7.2, is described in [4]. The second describes the remaining activities and is summarised below.

4.1.1 *EUTERPE Optimization*

Supported by: Xavier Sáez (BSC, Spain), Taner Akgün (BSC, Spain)

Collaborators: Edilberto Sánchez (Laboratorio National de Fusion, Spain)

The original parallelization of EUTERPE was based on MPI (using only tasks). This has been extended by introducing OpenMP in the most time-consuming routines and developing a hybrid solver (mixing MPI and OpenMP) to solve the quasi-neutrality equation. We have focused our investigation on improving the solver and developed a more complex version using the Block Jacobi Preconditioned Conjugate Gradient method. The results suggest that the speedup is good, as shown in Figure 14.

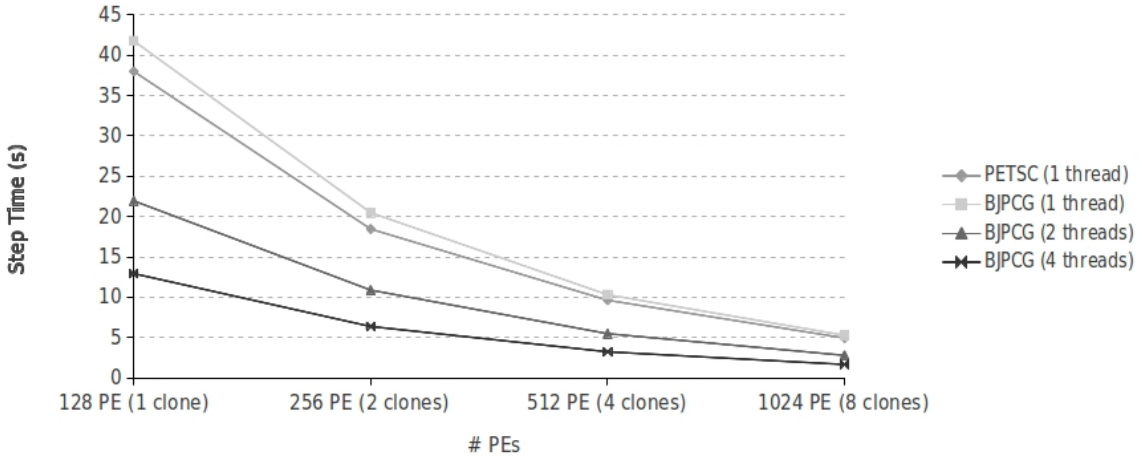

Figure 14: Benchmark tests of the hybrid Euterpe code

The next task was to port the code to the GPU architecture. We decided to start with the solver for the quasi-neutrality equation which allowed us to understand and test the CUDA framework. A more detailed analysis suggested that 'particle push' and 'current deposition' were the best candidates to port to accelerate the execution of the code. In particular, the particle push was the first choice, because there were no memory access conflicts as in the current deposition. It is possible that some unnecessary memory transfers are being done between the GPU and host, so this part requires further tuning.

The performance of EUTERPE is also affected by the choice of parameters. An analysis of the following parameters has been done: the step size, the number of markers, and the grid size. The influence of them on the signal to noise ratio and energy conservation has been studied.

EUTERPE also demands I/O processing, which may consume considerable time and can therefore potentially reduce speed-up at petascale, especially when checkpoints are activated. A study has been done to evaluate the feasibility of introducing I/O forwarding in EUTERPE to improve the performance. The study concluded that while a file creation bottleneck is observed, which can be reduced by user-space I/O forwarding, using the IOFSL forwarding software layer comes at prohibitive costs (see. "*Evaluating application I/O optimisation by forwarding layers*" by Jan Christian Meyer, Jorn Amundsen and Xavier Saez, [4]).

5 Weather, Climatology and Earth Sciences

The EC-EARTH 3 climate model was selected during the initial phase of the project, as it was proposed unanimously by the IS-ENES community, which represents the climate community across Europe. The NEMO model, one component of EC-EARTH, has been in the PRACE benchmark suite since the beginning of PRACE. Different IS-ENES community members have applied successfully for computing time to run EC-EARTH on Tier-1 systems. Furthermore, some other projects applied successfully for computing time on Tier-0 systems with climate models that use components of the EC-EARTH model.

The SPECFEM3D seismic wave propagation model was selected due to its broad user base, with the added benefits that the developers are very helpful and that PRACE has had a good experience of SPECFEM3D.

5.1 EC-Earth 3

The EC-EARTH model is a global, coupled climate model that consists of the separate components IFS, for the atmosphere, and NEMO, for the ocean, that are coupled using the OASIS coupler. During a meeting between PRACE and IS-ENES in Paris in December 2010, different priorities were defined. The following activities were started by PRACE: a performance analysis, efficient mapping tasks and threads, investigate the use of CUDA, analyze I/O patterns,. Several activities were started, in this cooperation, by IS-ENES: validate and benchmark the OASIS4 coupler for EC-Earth and the design of an application for ensemble simulations. Furthermore, the STFC partner worked on the implementation of dynamical memory allocation and load-balanced domain decomposition in NEMO.

The activity carried out by Task 7.2 in cooperation with the IS-ENES community is reported in the white paper *Performance analysis of EC-EARTH 3.1* and summarized below. The white paper integrates and complements the results of white paper High *Resolution EC Earth Porting, Benchmarking on Curie* by Chandan Basu, documenting the activity done in the context of Task 7.5. Furthermore, it references the white papers called "*Improving MPI communication latency on Euroben kernels*" and "*Analysis of 3DFFT on multi-GPU systems*", submitted as part of the Task 7.5 activity, since these papers are also relevant for EC-EARTH, but broader in scope.

5.1.1 *Performance analysis of EC-EARTH 3.1*

Supported by: John Donners (SARA), Chandan Basu (SNIC-LiU), Alastair McKinstry (ICHEC), Andrew Porter (STFC)

Collaborators: Eric Maisonnave (CERFACS), Sophie Valcke (CERFACS), Muhammad Asif (IC^3) , Uwe Fladrich (SMHI)

EC_EARTH was initially ported and run on the Curie System. Different configurations, using resolutions from T159 (approx. 128 km) to T799 (approx 25 km), were available for benchmarking. Porting to the Curie system was relatively straightforward, probably helped by the fact that the software and hardware used on Curie resembles the system used for the main development of the code. On Curie, the MKL library can be used to provide better performing BLAS routines. The activity was done in strict collaboration with partners IS-ENES, in particular SMHI (the Swedish Meteorological Institute) was leading the development and was very helpful in getting the model ported and running on different architectures. The collaboration between different developers throughout Europe was organised efficiently

through a website that was used to host the software, share plans and have discussions about errors, etc. In addition bug reports were always answered, and solved, quickly.
The IFS model consists of different modules, of which some are only run at certain intervals. The most important types of timesteps for our simulations are: computational, radiation, coupling, I/O, initialization and finalization. Although the I/O timesteps are the slowest and the least scalable, the main bottleneck for scaling of the highest-resolution configuration is the time needed for computational timesteps (see Figure 15).

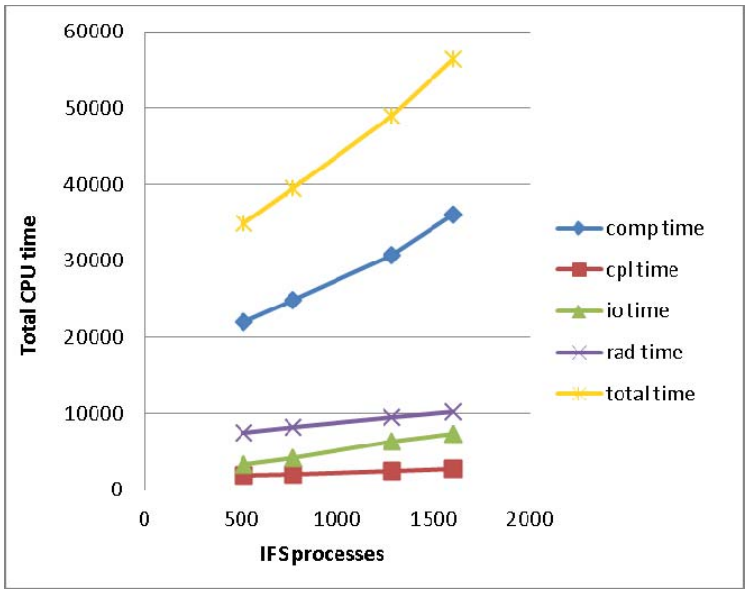

Figure 15: Parallel scaling for the IFS model.

Scalasca was used to analyze the performance of the model in detail. The IFS model uses two MPI_Alltoallv calls per timestep and these dominate the loss of scaling at 1024 cores. An example program is provided to test the scalability of the MPI_Alltoallv call. This shows that the use of the OpenMP functionality in IFS could increase scalability considerably, but the computational performance using OpenMP does not yet improve on Curie. Another PRACE white paper shows that the MPI_Alltoallv performance on Curie is suboptimal. Work is ongoing to make MPI_Alltoallv more efficient on Curie. It is expected that I/O and/or coupling does become a bottleneck when IFS can be scaled further than 2000 cores. Therefore, the OASIS team increased the scalability of OASIS dramatically with the implementation of a radically different approach (OASIS3-MCT), showing less than 1% overhead at 2000 cores. The scalability of NEMO was improved dramatically during an earlier PRACE project with the implementation of a changed communication pattern and the round-robin distribution of MPI tasks across nodes.

The I/O subsystem in IFS is described and is probably not easily accelerated unless it is rewritten and uses a different file format. The bottleneck for the I/O in IFS is not the disk activity itself, but the communication for redistribution and gathering of output fields. The Darshan tool was used to profile the I/O. Although the summary gives a nice first impression of the I/O activity, a more detailed analysis of the raw data is required to see the intensity and find specific bottlenecks. The CDO tool was found to be a bottleneck for post-processing large amounts of files. It was found that compiler optimizations, most notably SSE options and multi-thread support, will go a long way on the very regular operations in CDO. At an EC-EARTH meeting in Copenhagen in 2011 it was confirmed that CDO was not any longer the bottleneck in post-processing the data.

The IFS model uses a built-in routine FFT992 to calculate FFTs, which was replaced with an implementation to use the standard FFTW3 interface for 1-D FFT calls. IFS can now use standard external FFT libraries. This enables the use of the Intel MKL implementation of FFTW3 and the use of GPGPU hardware through the CUDAFFT-library. Testing is underway to find the performance gain for IFS using GPGPU hardware.

There was no discernible difference in performance between using a block and a cyclic distribution of MPI processes across the Curie nodes. Since the OpenMP-version of EC-EARTH on Curie was not faster than the MPI-only version, how to distribute a mix of MPI-only and hybrid applications efficiently across nodes on different systems was not investigated.

In the context of the collaboration with IS-ENES, the IC^3 partner developed Autosubmit, a tool to manage and monitor climate forecasting experiments by using supercomputers remotely. It is designed with the following goals: efficient handling of dependent jobs; optimal utilization of resources; start, stop and monitor experiments; auto restart in case of failure; database for experiment history. The current version of Autosubmit has an object oriented design and has been developed in Python.

5.1.2 *Further work on EC-EARTH 3.1*

The three components of EC-EARTH (IFS, NEMO and OASIS) have been engineered to run as one MPMD program. Porting the EC-EARTH package to new architectures is quite complicated as the different components were developed separately by different development teams. This has been significantly simplified by the new compilation system in EC-EARTH 3.1, which unifies the different build mechanisms into one XML file. The EC-EARTH model is ported to different architectures, including the Intel-based Curie system and the IBM Power6-based Huygens system (a Tier-1 cluster hosted by SARA).

The porting effort led to several modifications in the build system. Different configurations with a resolution up to T799 for IFS and 0.25 degree for NEMO (about 25 km for both components) were tested on the Curie platform. Many parts of the IFS model are only activated at specific intervals and the time per timestep can therefore vary considerably. This is a complicating factor when looking for a balanced combination of IFS and NEMO tasks. The SCALASCA tool was used to separate the performance of each component. It is best to choose sufficient MPI tasks for the NEMO model, so it is fast enough to keep up with the shortest interval between two coupling timesteps in IFS. An analysis of the different timesteps in IFS shows that the computational timesteps (without radiation, coupling or I/O) are the bottlenecks for further scalability on Curie. The performance of the IFS model depends critically on the MPI_Alltoallv primitive, which is in general quite slow on Curie and does not scale beyond 1000 MPI tasks.

A simple benchmark shows that the use of OpenMP in IFS could lower the MPI communication overhead. Although IFS supports OpenMP, we were not able to use it effectively on Curie. Although the OASIS coupler is not the major bottleneck, it is expected to be when the scaling increases. The OASIS team changed the coupled code from a stand-alone program to a library linked with the IFS and NEMO components, which diminishes the coupling overhead (OASIS3-MCT). A built-in FFT routine in IFS was replaced with a generic call to the FFTW3-interface, which enables the use of platform-optimized FFTW-routines, e.g. MKL or CUDA.

The distribution of MPI processes across the nodes does not change the performance. A description of the I/O subsystem in IFS is included as a first step to improve its I/O performance. The Darshan tool was used to get a good idea of the total I/O in EC-EARTH,

both of the genuine output files and logs. Different verbosity options were described to decrease the amount of metadata operations due to log updates. The development of a framework for ensemble simulations and a more flexible domain decomposition are also described.

The collaboration with the IS-ENES community is continued in PRACE2IP WP8, addressing somewhat different subjects: OASIS3-MCT is further analyzed, an IO-server approach, new dynamical cores and advanced features for the NEMO ocean model (GPU port and fault tolerance).

5.2 SPECFEM3D

The code SPECFEM3D simulates seismic wave propagation based upon the spectral-element method (SEM). The SEM is a continuous Galerkin technique, which can easily be made discontinuous. SPECFEM3D has two modes of operation; it can perform either global (Earth-scale) or local (continental-scale) simulations. The petascaling activity in this project focused mainly on the global earth version SPECFEM3D_GLOBE.

The activity on SPECFEM3D_GLOBE has been done mainly in two directions: the design of a hybrid implementation (mixed MPI + OpenMP) and the study of a 3D domain decomposition to efficiently deal with complex meshing issues. The activity done has been reported in three white papers, summarised below. The first one describes the activity done in Task 7.2 for the 3D partitioning of the internal mesher; the second presents the hybridization activity, done in collaboration between Task 7.2 and 7.5, and the last paper describes a parallel fast BEM for the Helmholtz Equation as an extension of SPECFEM3D. The latter work is a collaboration between Tasks 7.2, 7.5 and 7.6.

5.2.1 *3D partitioning in SPECFEM3D internal mesher.*

Supported by: Valentin Pavlov (NCSA)

A new parallel acoustic simulation package has been created, using the boundary element method (BEM). The acoustical simulation relies on a Fourier transform of the seismic elastodynamic data, resulting from SPECFEM3D_GLOBE, which are then postprocessed by a sequence of solutions to Helmholtz equations, in the exterior of the globe.

For the acoustic simulations BEM has been employed, which reduces computation to the sphere; however, the naive implementation suffers from quadratic time and memory complexity, with respect to the number of unknowns. To overcome the latter, the method was accelerated by using hierarchical matrices and adaptive cross approximation techniques, which is referred to as fast BEM.

First, a hierarchical clustering of the globe surface triangulation is performed. The arising cluster pairs decompose the fully populated BEM matrices into a hierarchy of blocks, which are classified as far-field or near-field. While the near-field blocks are kept as full matrices, the far-field blocks are approximated by low-rank matrices. This reduces the quadratic complexity of the serial code to almost linear complexity, i.e. O(n*log(n)), where n denotes the number of triangles. Furthermore, a parallel implementation was done, so that the blocks are assigned to concurrent MPI processes with an optimal load balance.

The novelty of this approach is based on a nontrivial and theoretically supported memory distribution of the hierarchical matrices and right-hand side vectors so that the overall memory consumption leads to O(n*log(n)/sqrt(N)), which is the theoretical limit at the same time. Figure 16 shows the scalability of SPECFEM3D with the original and modified internal meshers.

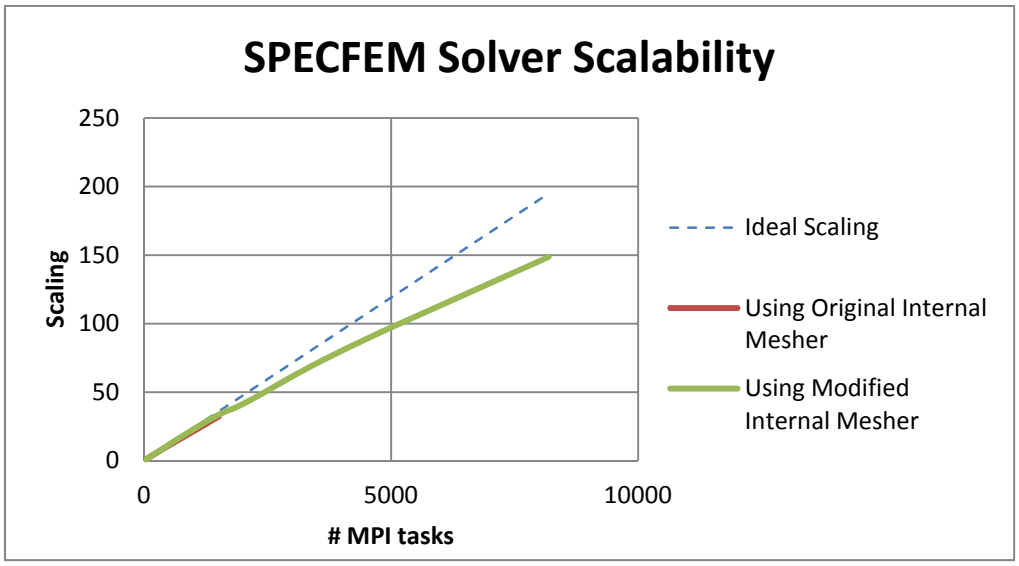

Figure 16: Scalability of SPECFEM3D with original and modified internal meshers. Original mesher does not scale further than 1500 cores due to time-step constraints.

5.2.2 *Hybridization of the SPECFEM3D_GLOBE*

Supported by: Marcin Zielinski (SARA), John Donners (SARA)

The hybrid approach (MPI with OpenMP threads) has been implemented for the solver application. The code has been parallelized very efficiently with MPI, however, due to its nature the application uses entirely static arrays with fixed dimensions that renders the whole application inflexible in use. The application has been divided in two parts called the 'mesher' (meshfem3D), which creates the three-dimensional mesh of Earth, and the 'solver' (specfem3D), which calculates synthetic seismograms in that three-dimensional earth model.

The split of the application between the mesher and the solver requires rerunning the preprocessing (mesher) when the number of MPI tasks changes, which also entails recompilation of the entire application. The application divides the three-dimensional mesh using five main parameters which are set explicitly by the user in the input file. The restrictions put on these parameters cause the inflexibility in choosing number of the MPI tasks, which can be also changed by the user himself. The hybrid version of the solver application adds more freedom to the number of parallel tasks that can run. Although not fully threaded with the OpenMP-approach, it is efficient for low numbers of OpenMP-threads. The implementation of the hybrid approach was a great occasion to discuss and better clarify the structure of the code with the main developers.

5.2.3 *A Parallel Fast BEM on Distributed Memory Systems for the Helmholtz Equation as an Extension of SPECFEM3D*

Supported by: Dalibor Lukas (VSB), Petr Kovar (VSB), Tereza Kovarova (VSB), Jan Zapletal (VSB)

The SPECFEM3D package uses a parallel implementation of a variation of the Galerkin procedure called Spectral Element Method (SEM). The advantage of this method is that it produces a diagonal mass matrix which allows for quick, explicit solving of the seismic wave equations. The drawback is that using an explicit method the solver is not stable unless the Courant-Friedrichs-Lewy (CFL) condition for convergence holds, imposing a certain inequality relation between the time step and spatial size of the spectral elements. The internal

mesher provided with the package uses a partitioning approach that directly links the number of parallel solver tasks with the spatial size of the elements. For any given mesh, increasing the number of parallel processes leads to decreasing the element size, and via the CFL condition to decreasing the time step. This directly compromises the scalability of the solver by requiring more time steps for the same amount of work with no gain in performance. The goal of this project was to improve the scalability of the package by reconsidering the partitioning approach of the internal mesher. We have analyzed the existing approach and we have proposed a new one, based on using the SCOTCH partitioning library, already integrated in the workflow for solving the problem on externally generated mesh. Our results indicate that the modified partitioning scheme leads to substantial performance benefits for regional modeling with the internal mesher in petascale environments. We validate the results by running the examples included in the package and observe that the solver produces identical synthetic seismograms when run with meshes generated by the original and modified internal mesher.

5.2.4 *Conclusions*

A new parallel acoustic simulation package has been created, using the boundary element method (BEM). Several algorithmic optimizations reduce the quadratic complexity of the serial code to almost linear complexity, i.e. O(n*log(n)), where n denotes the number of triangles. Furthermore, a parallel implementation was done, so that the blocks are assigned to concurrent MPI processes with an optimal load balance.

SPECFEM3D has been parallelized very efficiently with MPI, however, due to its nature, the application uses static arrays with fixed dimensions that render the whole application inflexible. The hybrid version of the solver application adds more freedom to the number of parallel tasks that can be run and it is efficient (at least for a low numbers of OpenMP-threads).

The SCOTCH partitioning library is used to create externally a modified partitioning scheme. This leads to substantial performance benefits for regional modelling in comparison to the internal mesher of SPECFEM3D in petascale environments.

6 Engineering and CFD

In engineering and CFD (Computational Fluid Dynamics) different users, depending on their individual problems, use quite different software. For this reason the application community of engineering and CFD is not organised as a single community.

In industry, codes from ISVs (Independent Software Vendors) dominate and usually the software vendors do not disclose the source code.

Such ISV codes are often used in research groups, while software developed especially for individual needs, is also widely used in research groups. Nevertheless openly available codes are used both in industry and research. Typically, from this wide area, CFD programs have the highest requirements for computing power and consequently the highest demand for HPC systems.

Only codes not bound by the disclosure limitations of ISV, were selected in Task 7.2. During our discussions we found that OpenFOAM and Code_Saturne, both from the CFD area, are the most widely used codes being openly available and requiring top level HPC systems for current problems in research and industry [2].

6.1 OpenFoam

The activity in OpenFoam planned in [2] was carried out by the following contributors:

CINECA	Ivan Spisso, Massimiliano Culpo
EPCC	Gavin Pringle
HLRS	Joerg Hertzer
LRZ	Orlando Rivera
METU Turkey	Murat Manguoglu
NTNU	Bjørn Lindi
ICHEC	Ivan Girotto, Michael Moyles, Peter Nash

The OpenFOAM® (Open Field Operation and Manipulation) CFD Toolbox is a free, open source CFD software package originally produced by a commercial company, OpenCFD Ltd. While this project was running OpenCFD Ltd was acquired by SGI Corp. OpenFOAM has a large user base from most areas of engineering and science, from both commercial and academic organisations.

The user and developers community is wide and active throughout Europe, the US and India. The project partners are in touch with different research groups working on relevant scientific and engineering cases which will benefit from the enabling of OpenFOAM on a Tier-0 system to make advances on their research activity.

The core technology of OpenFOAM is a flexible set of efficient C++ modules. These are used to build a wealth of solvers, to simulate specific problems in engineering mechanics, utilities to perform pre- and post-processing tasks ranging from simple data manipulations to visualisation and mesh processing, and libraries to create toolboxes that are accessible to the solvers/utilities, such as libraries of physical models.

Following deliverable D7.2.1 [2] priority of the work was given to bottleneck analysis. Both IPM (for analysis of computing performance) and darshan (for I/O analysis) proved to be useful for this. Unexpectedly, the disclosure of the model used at the ICE Group at the Energy Department of Politecnico di Milano was not allowed. Therefore the work at CINECA concentrated on benchmark cases known to have similar performance characteristics to this case. In addition to the bottleneck analysis, some studies for performance and scalability improvements were done.

The parallelization of OpenFOAM is performed by means of MPI (Message Passing Interface). Applications and solvers in OpenFOAM are the same for serial or parallel execution. In parallel the master-slave configuration is used, where a small set of MPI functions are deployed; non-blocking and blocking send/receive functions and reduction functions are at the core of each solver. The whole communication system is encapsulated into a single library. Parallel optimization and analysis can be done within this library, and no extensive modification is required in other sections of the code.

OpenFOAM's parallel behaviour is not well understood when run on massively parallel systems. Scalability and efficiency of OpenFOAM is still an open debate, depending of the solvers and the input data.

Because of the complexity of OpenFOAM, and because of different priorities of different project partners and community members, work on OpenFOAM was done in several groups. Therefore the results were reported in several white papers:

- Orlando Rivera, Karl Fürlinger: *Parallel Aspect of OpenFOAM in the solution of Turbulent Flows*;
- Murat Manguoglu: *Parallel solution of sparse linear systems in OpenFOAM*;
- Massimiliano Culpo: *Current bottlenecks in the scalability of OpenFOAM on massively parallel clusters*;
- Michael Moyles, Peter Nash, Ivan Girotto: *Performance Analysis of Fluid-Structure Interactions using OpenFOAM*.

The activity reported in these white papers is the result of a close collaboration of Task 7.2 with Task 7.5 (for improvements of algorithms) or 7.6 (for improvements of I/O). The work of Murat Manguoglu was mainly part of WP7.5 but is nevertheless reported here.

Some other white papers, directly documented as part of the specific activity in Task 7.5 or 7.6 are related to OpenFOAM. The white paper *I/O-profiling with Darshan* by Bjørn Lindi presented in [4] should be mentioned in this sense [10].

Some more work not mentioned in the white papers was done by Gavin Pringle (EPCC), mostly preparation of installations and test runs. HLRS continued with the coordination for the work on OpenFOAM.
All test cases used are either established within the community or real applications selected from current usage.

The results from the white papers are described in more details in the following sections.

6.1.1 *Parallel Aspect of OpenFOAM in the solution of Turbulent Flows*

Supported by: Orlando Rivera (LRZ, Germany).

Collaborators: Karl Fürlinger (LMU, Germany)

The test case used was a Large Eddy Simulation (LES). LES is based on the concept that small scale turbulence is isotropic and can be modelled, while larger turbulent and energetic eddies can be simulated. The backward-facing test case with a *Re* of 4800 with respect to the step height is used. The solver is called pisoFoam for LES, OpenFOAM version 1.7.x, and uses the Pressure Implicit solution by Split Operator method (PISO). At the sides of the domain, periodic boundary conditions are used, while top and bottom were set as non-slip walls.

Pressure equations were solved using the geometric-algebraic multigrid (GAMG) solver with 3 pressure corrections. Other fields were solved with the Biconjugate gradient method (BiCG) solver. Tests were run over 100 time steps with output at every 50 time steps.

Three meshes with different resolutions were used. The first mesh has 250, 96 and 64 cells in the x, y and z direction respectively. It contains 2.15 millions of hexahedrical cells while the second and third meshes contain 4 and 8 times more cells than the first mesh, respectively. 256 MPI tasks were used on the 8 million-cell mesh, and 512 on the finer mesh. These meshes were used in weak scaling studies.

The IPM (Integrated Performance Monitoring) application is a portable profiling and workload characterization tool for MPI applications and was used to gather profiling information. IPM minimizes drastically the overhead caused by application instrumentation and displays very detailed information at the same time. It makes important parameters visible beyond mere studies of scalability. IPM identifies bottlenecks, detects hot spots and collects statistics that help to optimize representative sections of the code. IPM was able to instrument OpenFOAM despite its complexity, i.e., the hidden MPI implementation, use of dynamic libraries and very complex C++ constructions; other similar tools were unsuccessful or provided unusable data.

All tests were conducted in a massively parallel general purpose computer, an SGI Altix 4700, with 9728 Intel-Itanium2 cores, a peak performance of 62.3 Tflops and 19 partitions connected by a high performance NUMA link interconnection. All runs, when possible, were restrained to a single shared-memory partition; in case a partition was not large enough, two partitions with the same number of cores were specified. The system was operated by the Leibniz Supercomputing Centre (LRZ).

For the scalability test 16, 32, 64, 128 and 256 MPI tasks were used on the 2.15 million-cell mesh. All runs have the same setup and the domain decomposition was done with the Metis partitioner in order to have approximately the same number of cells per sub domain and to minimize the maximum connectivity of the sub-domains.

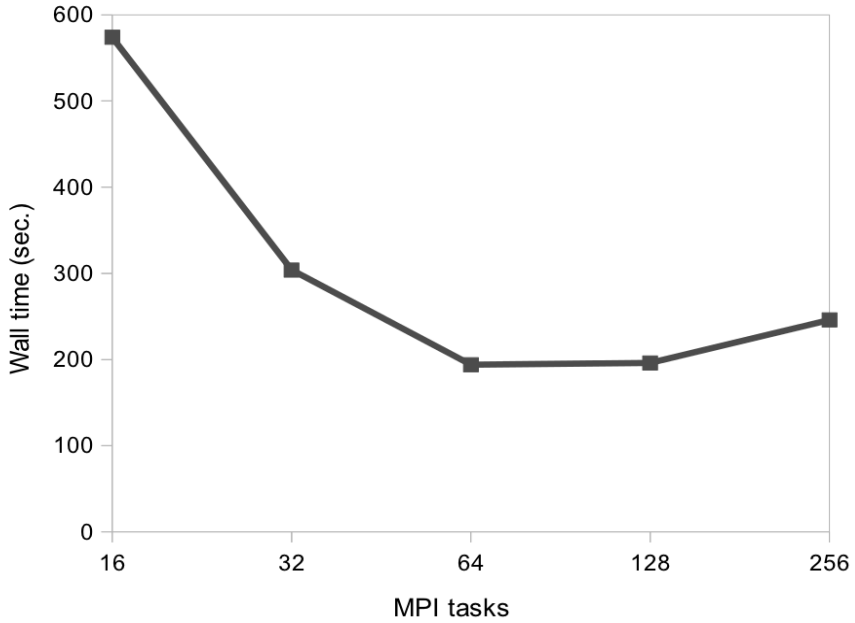

Figure 17: Scalability for 16, 32, 64, 128 and 256 MPI Tasks.

From Figure 17 we see that the pisoFoam solver has an acceptable scalability up to 64 MPI

tasks. However, with 128 and 256 MPI tasks the performance deteriorates.

As described in more detail in the white paper some patterns and specific characteristics of the bottlenecks have been discovered. Some configurations are more suitable for scaling. One of these parameters is to specify the GAMG solver.

This solver converges rapidly to a solution, faster than the BiCG, but its MPI footprint increases and become more evident with a larger number of MPI tasks. The scalability of the GAMG solver is limited up to a certain point (for these cases 64 MPI tasks). For finer meshes you can still deploy 512 MPI tasks using the GAMG, and despite its overhead, it is the sensible choice for BiCG. For further improvement an even more detailed investigation of the GAMG and its scalability issues is needed.

A good partitioning and domain decomposition are vital in order to reduce the MPI time and number of visits of MPI functions. Metis is one of the most used graph-based partitioning implementations in many areas of science and engineering. However, a deeper knowledge of these implementations and their impact on the algorithm is required. Improved or new partitioning methods have to be developed to produce domains which are as balanced as possible.

Simple theories of Volume/Area were insufficient here. These partitioning strategies have to take in account not only the number of elements, or communication area, but also locality, near neighbours, topologies, nodal and element weights, etc. The list of influential parameters is quite long but a decent trade off has to be found.

All these findings would not have been possible without profiling and tracing tools. IPM was flexible enough to produce the required information (where other tools have failed) and at the same time it was concise and neat.

6.1.2 *Parallel solution of sparse linear systems in OpenFOAM)*

Supported by: Murat Manguoglu

The most time consuming operation in CFD codes including OpenFOAM is the solution of sparse linear systems. This is more pronounced when the mesh is fine (i.e. the coefficient matrix is large). With the introduction of multicore processors and emergence of exa-scale clusters in which a single prcoessor contains many cores, it is essential to come up with new algorithms that are tolerant to the memory and cache hierarchies of these platforms and communication is mostly limited to be between the neighbouring nodes. Classical preconditioned iterative solvers, although scalable, are known to be not robust while direct solvers are robust yet they only provide limited scalability.

We have recently developed a general sparse solver based DS factorization as opposed to LU factorization. The solver can be either used as a direct solver or as a solver for a preconditioned linear system where the preconditioner is obtained by ignoring small elements in absolute value. An inner and outer iterative scheme exists. The local solves are done in parallel using either a direct solver or incomplete LU factorization but it is possible to use sparse approximate inverse or other types of approximate solvers as well.

The algorithm used is based on sparse DS factorization and called domain-decomposing parallel sparse solver (DDPS).

OpenFOAM comes with the variety of preconditioners: Diagonal Incomplete-Cholosky (DIC), faster diagonal Incomplete-Cholesky (FDIC), diagonal Incomplete-LU (DILU), diagonal, and geometric-algebraic multigrid (GAMG). We note that the DIC, FDIC, and DILU are block jacobi type preconditioners (diagonal preconditioner is simple jacobi

preconditioner) and they do not consider the elements outside the off-diagonal blocks. Gemortric multigrid preconditioners are known to be fast. However, since they require some information about the geometry they are not applicable to all problems. Hence, algebraic mutligrid (AMG) preconditioners are developed to alleviate this weakness of geometric multigrid preconditioners. Algebraic multigrid preconditioners, however, lack strong scalability. The DDPS solver has several advantages compared to traditional direct and iterative methods. Perhaps the most significant one is that it is a hybrid solver that combines direct and iterative solvers. Hence, it is as scalable as most iterative solvers and as robust as a direct solver.

The Lid Driven Cavity Flow was used as a test case. Due to the simplicity of the cavity geometry, applying a numerical method on this flow problem in terms of coding is quite easy and straight forward. Despite its simple geometry, the driven cavity flow retains a rich fluid flow physics manifested by multiple counter rotating recirculating regions on the corners of the cavity depending on the Reynolds number. We have extracted a linear system from the problem with Reynolds's number of 5000 and using a mesh with 4 million unknowns using OpenFOAM v2.0.

For the following numerical test Curie cluster located at CEA, France was used. Curie consists of 360 "fat" nodes where each node has 4 eight core Intel EX X7560 processors running at 2.26 GHz and 128 GB of memory. The nodes are connected with an InfiniBand QDR Full Fat Tree network.

We have used the BiCGStab as the outer and inner iterative solver in DDPS. In Tables Table 5 and Table 6, we present the parallel scalability of the DDPS solver using only MPI (Table 5) and MPI combined with OpenMP (Table 6). We note that the algorithm is implemented such that it mostly relies on an efficient BLAS kernel. For both cases we use the Intel Fortran compiler and for BLAS we use single and multithreaded Math Kernel Library (both the compiler and MKL are part of Intel Composer XE ver. 2011.3.174), Bull MPI (ver. 1.1.10.1) with "-O2" compiler optimization level and "-openmp" to enable OpenMP threading. The hybrid runs are obtained by just enabling threads during runtime and using multithreaded version of MKL while the pure MPI runs use sequential MKL and one thread per MPI process. In both cases we use no more than 16 cores per node due to memory limitations. While the total solve time decrease as we increase the number of MPI processes up to 64 for the given problem size, it starts to increase beyond 32 processes for the MPI-OpenMP runs.

The best time is obtained to be 1.55 seconds. If one, on the other hand, uses only MPI the scalability is not affected by the increase in the number of MPI processes and the best time is obtained to be 1.00 seconds. The most likely reason for the pure MPI implementation scaling better is the fact that the MPI processes run across a smaller number of nodes and hence the MPI communication is mostly handled via the shared memory. The hybrid implementation, however, places one MPI process per node and therefore the communication time starts to dominate the total time especially for large number of processes. If one considers the overall efficiency of the algorithm, we observe that the efficiency is higher if one uses only MPI even though it makes sense to use some number of threads when the amount of work per MPI processes is large (up to 32 processes for this test problem). We note that the factorization stage is not affected by increasing the number of threads per node which is not surprising as we use an unthreaded routine for it. Nevertheless, the factorization time is a very small fraction of time and it scales as one increase the number of partitions (i.e. number of MPI processes).

nodes	processes	cores	factor	solve	total
1	8	8	0,41	22,66	23,08
1	16	16	0,20	19,38	19,58
2	32	32	0,09	1,82	1,91
4	64	64	0,03	0,97	1,00

Table 5: Scalability of DDPS using one MPI process per core on Curie (given as wall clock time in seconds for factor, solve and total)

nodes	processes	cores	factor	solve	total
8	8	128	0,37	7,06	7,44
16	16	256	0,18	6,57	6,75
32	32	512	0,09	1,46	1,55
64	64	1024	0,03	3,71	3,74

Table 6: Scalability of DDPS using 1 MPI process per node and 16 threads per process on Curie (given as wall clock time in seconds for factor, solve and total)

part	red. sys. size	outer	avg. inner	rel.res
8	3200	24,5	1,16	3,47E-11
16	9600	24,5	1,19	2,57E-11
32	32	26,5	1,83	1,27E-11
64	96	28	1,72	2,65E-11

Table 7: DDPS solver parameters using different number of partitions
Size of the reduced system, number of outer iterations, average number of inner iterations and the final relative residual for DDPS solver using different number of partitions

In Table 7, the reduced system size, number of outer and average inner iterations, and the final relative residual is presented. Both inner and outer numbers of iterations are very weakly dependent on the number of partitions.

We have presented scalability results of DDPS algorithm for a sparse linear system extracted from OpenFOAM 3D lid-driven cavity test problem on Curie cluster. Pure MPI implementation performs and scales better than the MPI-OpenMP hybrid implementation.

6.1.3 *Current bottlenecks in the scalability of OpenFOAM on massively parallel clusters*

Supported by: Massimiliano Culpo (CINECA)

Collaborators: Ivan Spisso (CINECA)

To understand the scaling behaviour of OpenFOAM on the latest PRACE Tier-0 and PRACE Tier-1 computers we analyzed the scalability of two OpenFOAM solvers on problems that are considered to be good representatives for realistic production runs on Tier-0 systems. The tests have been executed on two computers: CINECA PLX (PRACE Tier-1 system) and TGCC Curie (PRACE Tier-0 system). OpenFOAM versions 1.6-ext, 1.7.1 and 2.0.0 have been installed both with GNU and Intel C++ compilers. The code has been instrumented with the Integrated Performance Monitoring library.

Much effort has been devoted in establishing and maintaining contacts with the Italian OpenFOAM developer communities and with the Wiki head developer (Prof. Hrvoje Jasak).

Two test cases were used: The first one was the lid-driven cavity flow benchmark involving the solution (using the icoFoam solver) of a laminar, isothermal, incompressible flow in a three-dimensional cubic domain. All the boundaries are modelled as walls and are considered static, with the exception of the top one that moves in the x direction at a speed of 1 m/s. Notice that, despite its simplicity, this benchmark is of particular interest as it is widely employed at different sites for benchmarking purposes, thus permitting a direct comparison of different application set-ups. The second test case was the droplet splash benchmark involving the solution of a two-phase incompressible system that models the impact of a droplet against a wall. The solver being used is in this case interFoam.

Checks proved that the observed performance does not significantly depend on the compiler used; also the overhead of the IPM instrumentation was found to have no significant performance degradation. The results clearly show that the size of the problems that can be handled on a HPC cluster lies far beyond the limitations imposed by smaller in-house clusters. Still, the scalability of OpenFOAM is such to permit a proper usage of Tier-1 rather than Tier-0 architectures. Furthermore the computational performance on the single node is limited by the bandwidth to memory available on the node itself. As already pointed out in literature, the scalability and performance issues are both related to the sparse linear algebra core libraries. To motivate the last statement the Preconditioned Conjugate Gradient (PCG) method, was briefly analyzed as a representative of the class of Krylov subspace iterative solvers. The most relevant operations performed during each PCG iterative cycle are scalar products, preconditioning steps and matrix-vector multiplications. It is worth stressing that these operations are common to all the methods based on Krylov subspaces, and therefore their optimization will have a positive impact on the whole set of linear solvers.

The conducted analysis suggests two orthogonal ways to improve the performance of OpenFOAM solvers. The first is the implementation of cache blocking techniques to reduce the number of cache misses in the core operations due to the random access patterns. This may require considerable effort as the basic matrix class must be revised to allow for the storage of small, contiguous blocks of scalar type as ”unit” entries of the format. The second approach is the modification of the basic linear algebra routines in a way that makes them multi-threaded. This will indeed mitigate the increase in the time spent inside MPI routines as well designed multi-threaded tasks can ideally exploit the resources provided by the largest shared memory portion of the machine.
Preliminary results on PLX with the lid-driven cavity flow test case are shown in Figure 18. As expected the best performance (29 sec.) obtained by the hybrid code is 20% faster than the best pure-MPI run (36 sec.) and occurs at double the number of cores. Notice that though these timings stems from an ad-hoc test case, they suggest that a more careful hybridization of the linear algebra core libraries may improve both the scalability and the usability of OpenFOAM on many HPC architectures. What could be expected in the best case for a production run is a substantial reduction of the intra node MPI communication and, as a consequence, a better scalability of the application.

More extensive studies on this approach and on the effect of cache-aware storage formats will be conducted in the future.

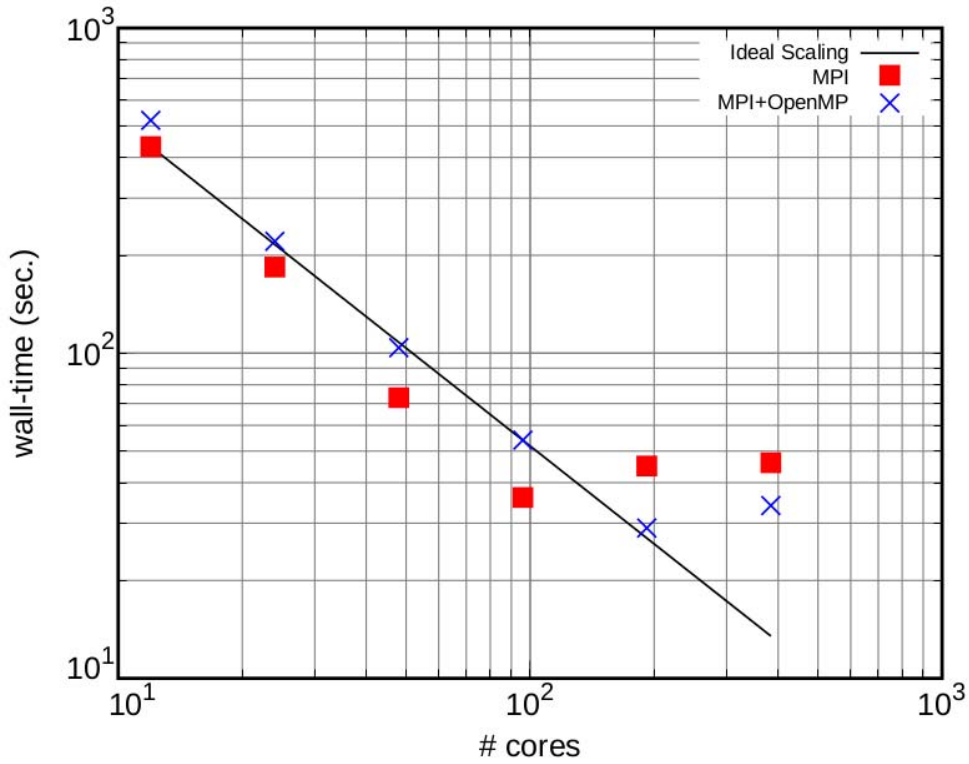

Figure 18: Scaling results obtained on multiple nodes for the 100×100×100 cells case
A diagonal preconditioner has been used instead of an Incomplete Cholesky Factorization for the PCG algorithm.

6.1.4 *Performance Analysis of a large scale Fluid-Structure Interaction*

Supported by: Michael Moyles, (ICHEC, Ireland)

Collaborators: Peter Nash, Ivan Girotto - (ICHEC, Ireland)

This study focused on a simulation provided by a CFD group within the National University of Ireland, Galway (NUIG). Their research focuses on understanding the turbulent scale of fluid interactions with a large, active research vessel – the "Celtic Explorer". The model physics and domain remained entirely unaltered during preliminary scaling tests. The simulation was composed of mesh creation and decomposition, snapping the mesh to the *CelticExplorer* surface and then executing the *interFoam* solver over each subdomain. The solver and *snappyHexMesh* utility were found to be the most time consuming operations and so these were analysed individually. The *interFoam* tests were performed on the Tier-0 machine, Curie, the results of which showed a significant performance increase up to 256 cores but extremely poor scaling thereafter, see Figure 19. In a similar fashion to *interFoam* it demonstrated good performance increases but only up to 36 cores (best results found using 12 cores). After 48 cores, the performance decreased dramatically.

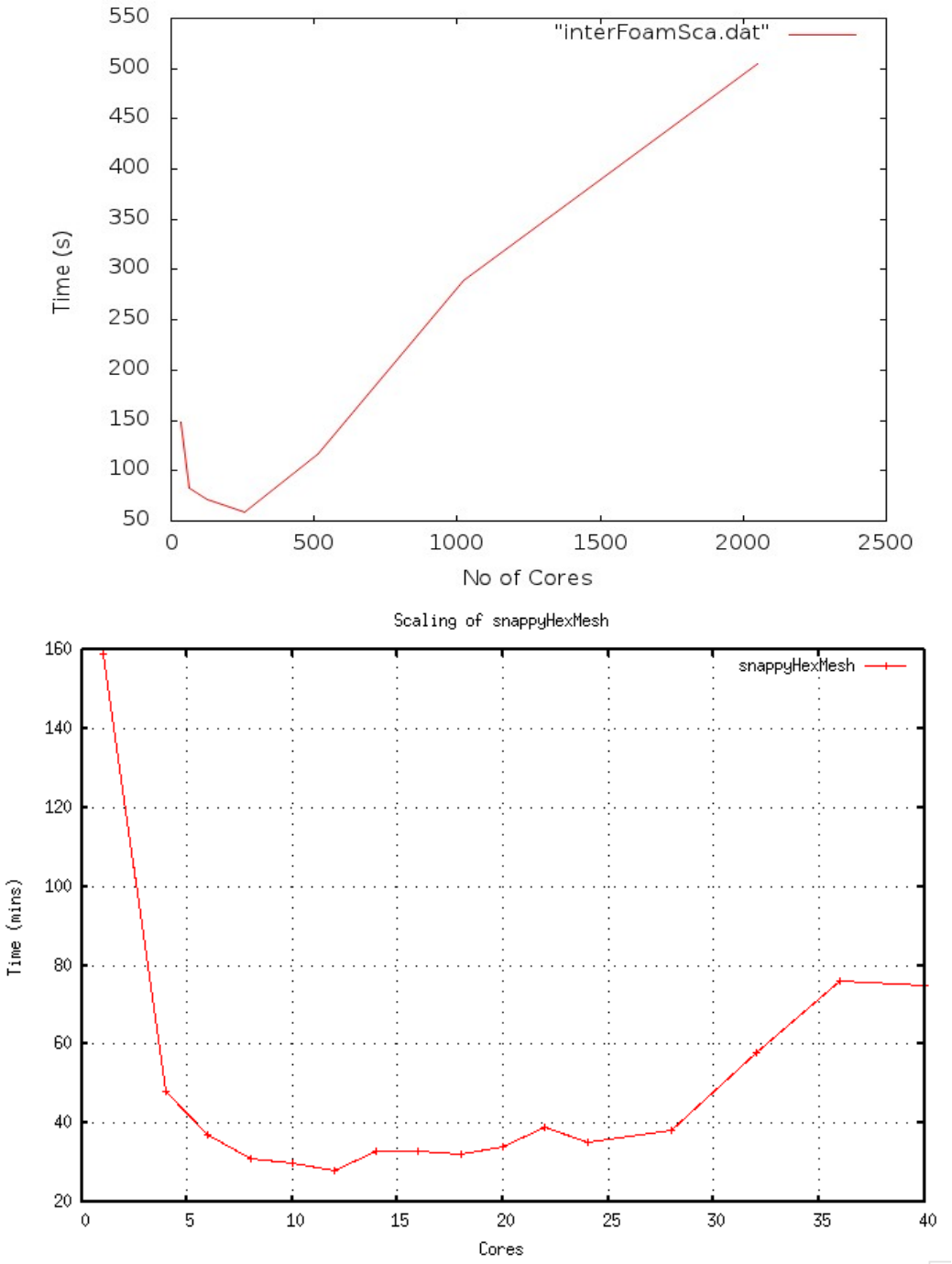

Figure 19: Scaling of interFoam on Curie (top) and snappyHexMesh on Stokes (bottom)

The two binaries were analysed individually. *snappyHexMesh* was found to be highly sensitive to domain characteristics, most notably the aspect ratio of the block mesh cells. The best performance was found for cells with an aspect ratio as close to 1:1:1 as possible. Remarkably, ~600,000 cells of even dimensions were found to converge about 15% faster than ~400,000 cells with a 7:2:2 aspect ratio.

Aspect Ratio	*# of Cells*	*# of Domains*	*Convergence (min)*
7:2:2	421,875	144	348
1:1:1	655,360	144	295
1:1:1	81,920	144	48

Table 8: Affect of cell aspect ratio on snappyHexMesh scaling

Using the I/O profiler, Darshan, the behaviour of the *interFoam* solver was profiled, the output shown in Table 9 and Figure 20. The most striking result from this exercise was the significant increase in the number of files read by the solver as the number of cores increases.

The number of files written increased linearly with an increasing number of cores, as expected, and this behaviour was found to be controlled by the *writeInterval* in *controlDict.* Importantly however, lowering the interval with which the solver wrote to disk did not produce significant performance gains.

# Cores	*Walltime*	*Cumulative MetaData*	*% of I/O in simulation*	*# of File Created*	*# of Files Read*	*Average File Size*	*# of Stat() calls*
64	686	64	9.3	512	1089	597K	500000
128	801	202	25	1024	2177	317K	10000000
256	890	274	31	2048	4353	163K	20000000
512	1161	389	34	4096	9729	84K	4400000
1024	2248	892	39	8192	17409	47K	8500000

Table 9: I/O Profile of interFoam using Darshan

Table 9: I/O Profile of interFoam using Darshan

shows that as the percentage of I/O in simulation increases the number of files read has a more dramatic increase than the number of files created. An analysis of the model set up found this to be due to an OpenFOAM function that controls modification during the execution of the solver – *runTimeModifiable* located in *controlDict*. In the existing set up (as used by NUIG) the enablement of this function told the solver to reread all dictionary input files at the start of each timestep – of which there are 4E03. As the model does not require run time modifications this feature was disabled and the scaling was rerun, the output contained in Figure 21.

Overall, this study has shown that each OpenFOAM case must be considered independently when scaling is concerned. Although there have been cases within this paper that show petascaling of OpenFOAM to be a highly successful operation, we cannot apply a general set of rules to improve the performance of any one particular model. We have proven that there are measure that can be taken to ensure optimal performance is achieved within a model, but model geometry and functions used along with user output requirements can impede strong scaling of OpenFOAM. Nevertheless, if the user can enforce such measure without altering their studies then OpenFOAM will scale very well – certainly up to 264 cores as has been shown here.

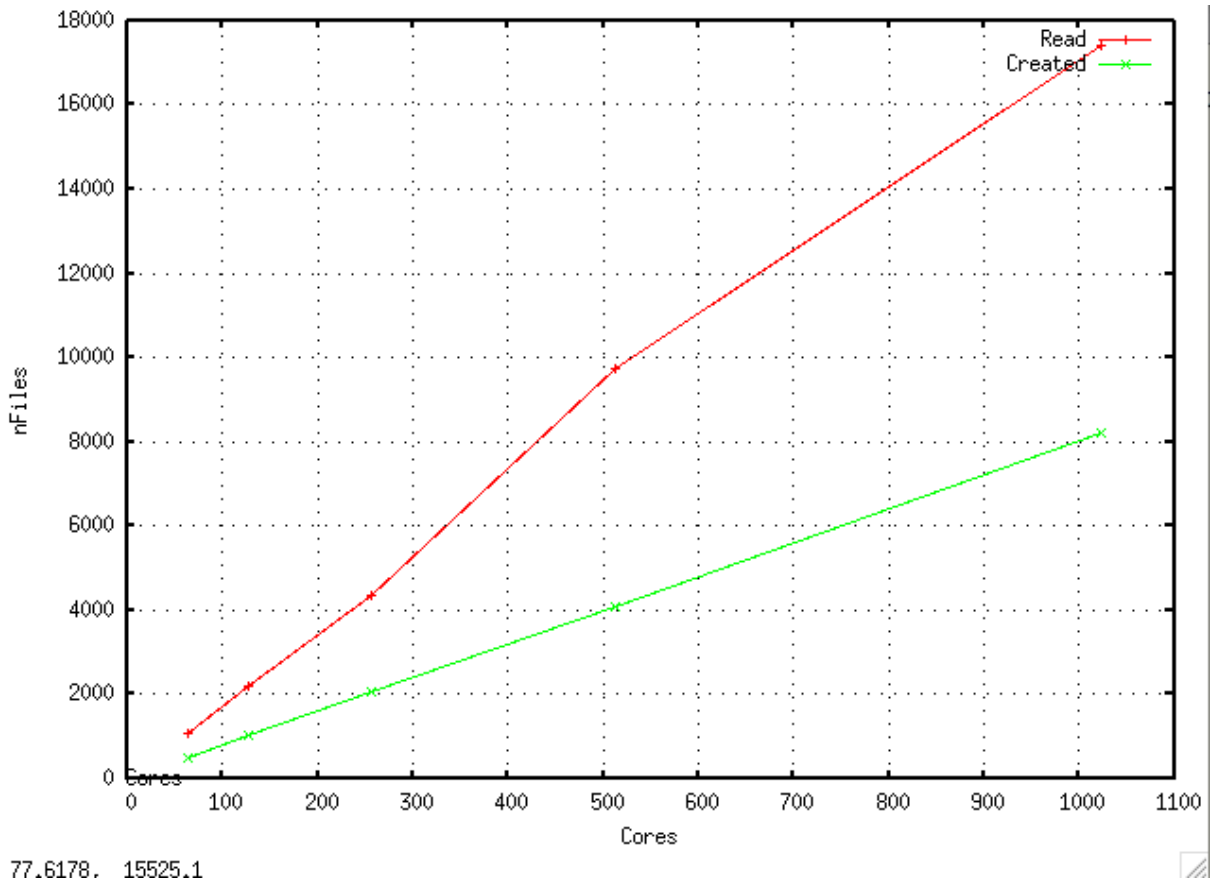

Figure 20: Files read and written by interfoam for increasing cores

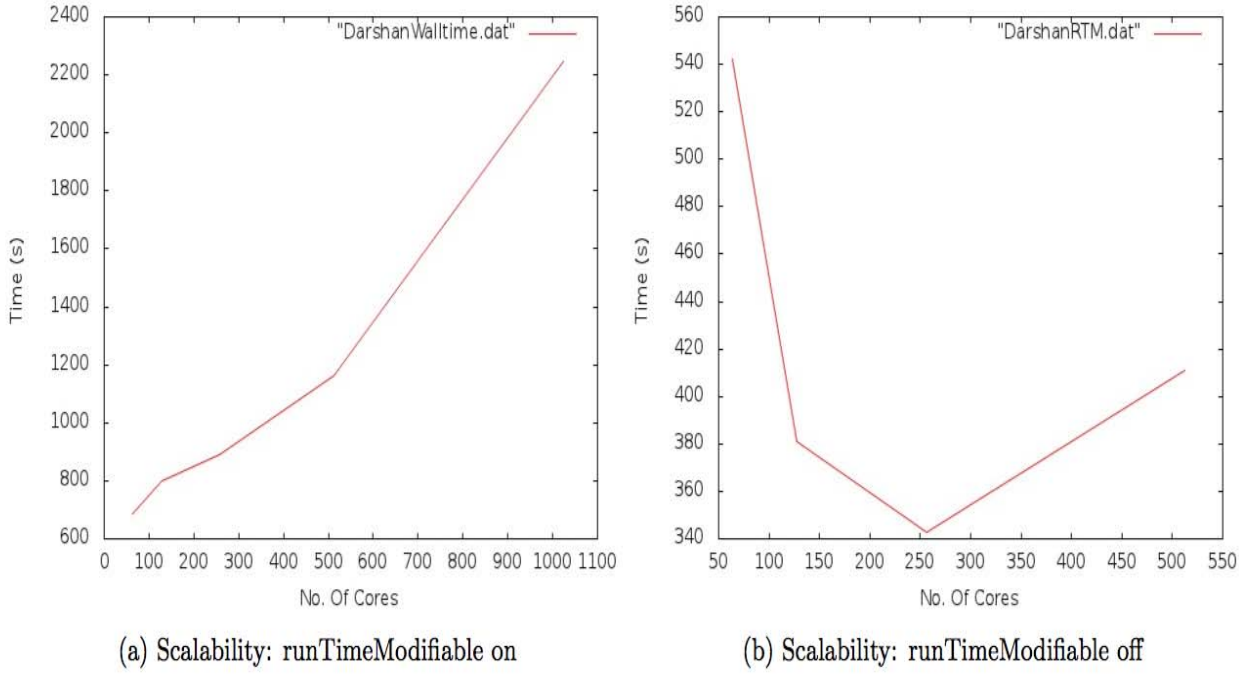

Figure 21: Affect of runTimeModifiable on walltime of interFoam

For this model, optimal performance was achieved by executing *snappyHexMesh* over a subset of the cores used to execute the *interFoam* solver on. To do this, there must be two sets of decomposing instructions (*decomposeParDict*) and care must be taken to ensure the relevant binaries read the correct set of instructions. By executing *snappyHexMesh* over 12 cores and *interFoam* over 264 we obtain the following speed up shown in .Table 10: Speedup of snappyHexMesh and interFoam

Model	snappyHexMesh walltime (min)	interFoam walltime (min)	Total walltime (min)
Sequential	316	951	1267
Optimised	28	341	369

Table 10: Speedup of snappyHexMesh and interFoam

6.1.5 *Conclusions*

The work done by the partners involved in OpenFOAM performance was quite extensive and focused on different aspects. The activity took a lot of human resources, given the complexity of the software involved. Several performance bottlenecks of OpenFOAM have been shown and studied, like MPI communication, memory access (cache misses) and I/O. The activity was quite complex and the first steps for improvements were done and tested. Given the importance of the application code, both for academic and industrial activity at European level, the continuation of the scaling and optimisation activity in the next future is recommended.

6.2 Code_Saturne

It is estimated that a mesh of about 15 billion cells would be required to simulate by Computational Fluid Dynamics (CFD), and more specifically by Large-Eddy Simulation (LES), the flow in a full reactor of a power plant. Code_Saturne is a multi-purpose CFD software developed by EDF-R&D since 1997. It has been released as open source in 2007 and is now distributed under a GPL license. The code was originally designed for both industrial applications and for research activities in several fields related to energy production, and has been selected as part of the PRACE benchmark suite for the engineering community.

Code_Saturne is based upon a co-located finite volume approach that can handle three-dimensional meshes built with any type of cell (tetrahedral, hexahedral, prismatic, pyramidal, polyhedral) and with any type of grid structure (unstructured, block structured, hybrid). The code is able to simulate either incompressible or weakly compressible flows, with or without heat transfer, and has a variety of turbulence models. The main bottleneck in CFD applied to complex geometries, is the ability to generate large meshes and the time required for it.

Alternative solutions exist, as for instance joining parts of a large mesh or splitting all the cells of an original mesh (also called mesh multiplication). Just to give an example, Figure 22. presents the contour map showing velocity variations in the liquid cooling fuel rods in a nuclear reactor, obtained with a Code_Saturne simulation.

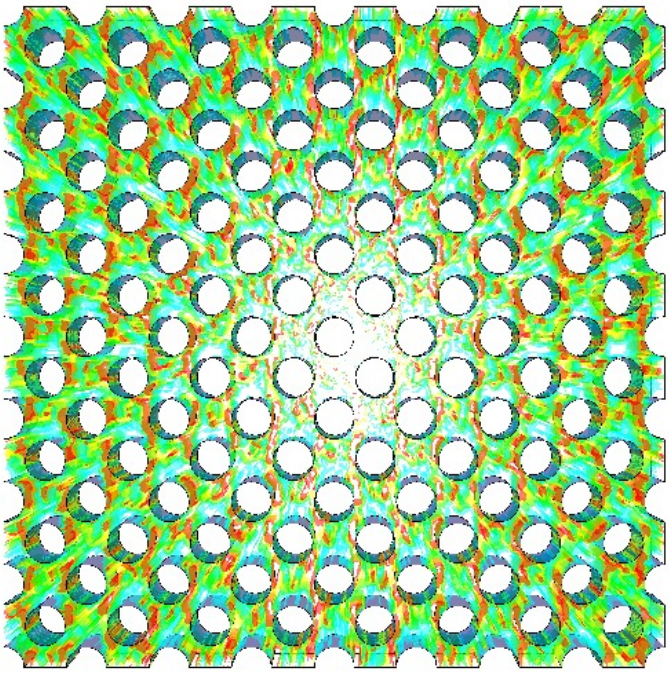

Figure 22: Simulation of the velocities in the cooling liquid fuel rods in a nuclear reactor

It is premature to aim at this stage for 15 billion cell meshes, but a 2 billion cell mesh has already been generated by a Code_Saturne joining procedure in 2011, the solver not being run.

The activity related to Code_Saturne carried out in Task 7.2 focused mainly on three technical objectives:

- generate meshes of more than 2 billion cells,
- test the Navier-Stokes solver for LES,
- run Code_Saturne on 60,000+ cores.

The work performed on Code_Saturne by Task 7.2 in collaboration with Task 7.6 is reported in three white papers, one from Task 7.2 (C. Moulinec, A.G. Sunderland: *Optimisation of Code_Saturne for Petascale Simulations*) [10] and two from Task 7.6 (Turka, C. Moulinec, A.G. Sunderland, C. Aykanata: *Code_Saturne Optimizations in Preprocessing*, Pavla Kabelikova, Ales Ronovsky, Vit Vondraka: *Parallel Mesh Multiplication for Code_Saturne)* [4], [10].

Partitioning for hundreds of millions or even billions of cells needs to be undertaken in parallel as neither front-end machines nor cluster nodes have sufficient memory to perform this task serially. The parallel partitioner in Code_Saturne relies on a Space Filling Curve (SFC) algorithm based on the Morton approach, but other libraries might be used within Code_Saturne pre-processing stage, e.g. METIS 5.0, SCOTCH 5.1.12, ParMETIS 4.0.2 and PT-SCOTCH 5.1.12. Former tests conducted on Jugene with a 107M mesh [2] concluded that PT-SCOTCH produces better partitionings for Code_Saturne than ParMETIS and SFC (Morton). In order to prepare for the new generation of high-end machines comprising of

many nodes with many cores per node, Bilkent University has developed a hierarchical mesh partitioner based on graph partitioning between nodes (based on ParMETIS) and a fast partitioning locally between cores of the machine. This work was also undertaken within the PRACE-1IP WP7.6 project and is summarised in Section 6.2.3.

6.2.1 *Optimisation of Code_Saturne for Petascale Simulations*

Supported by: C. Moulinec (STFC, UK), A.G. Sunderland (STFC, UK)

Collaborators: P. Kabelikova (VSB, Czech Republic), A. Ronovsky (VSB, Czech Republic), V. Vondrak (VSB, Czech Republic), A. Turk (BU, Turkey), C. Aykanat (BU, Turkey) , C. Theodosiou (AU, Greece)

This white paper included an overview of the complete project and used the outputs from the WP7.6 Code_Saturne projects described in the sections below to undertake a performance analysis of very large scale datasets on the PRACE Tier-0 machines.

The main test cases consist of Large-Eddy Simulations in a bundle of staggered-distributed tubes, the configuration obtained by multiplying an elemental configuration containing a whole tube in its centre and four quarter of tubes in the corners. The elemental configuration contains about 13 million (13M) cells (the 2-D cross-section has 100,040 cells and 128 layers are used in the third direction).

The two PRACE Tier-0 machines available within Task 7.2., i.e. Jugene and Curie have diverse architectures, the first one with 4 cores and 2GB RAM per node and the second one with 32 cores per node and 128GB RAM per node. Unfortunately the size of the jobs to be run on Curie is currently limited to a maximum of 2,048 processors, which prevents testing the code at scale. For these reasons, it has been decided to first compare Code_Saturne performance on Jugene and Curie, before running the largest simulations on HECToR (a Cray XE6 machine), where runs of up to 65,536 cores can be undertaken.

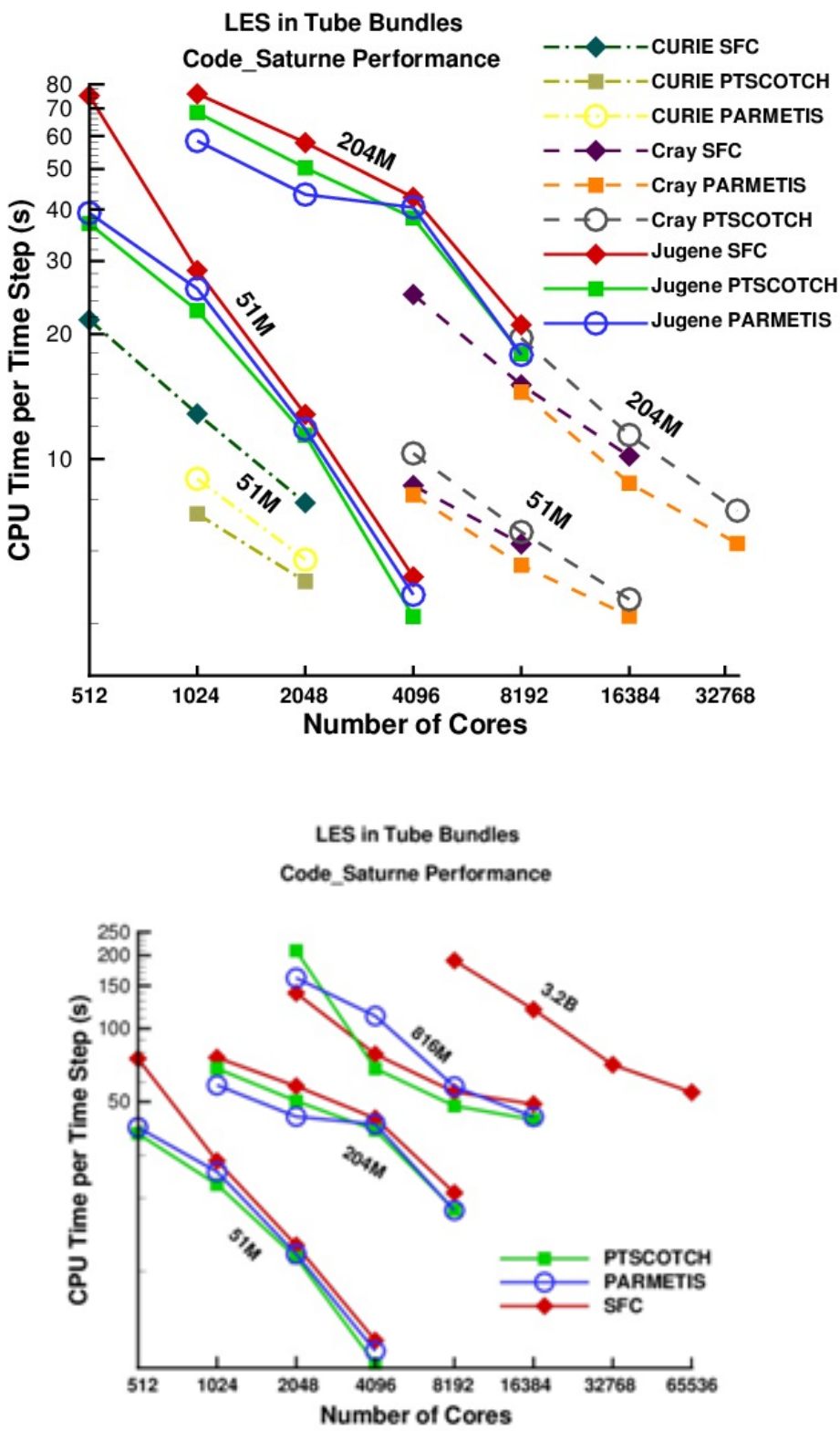

Figure 23: Code_Saturne performance on Jugene, Curie and HECToR
(a - top) Code_Saturne performance on Jugene, Curie and HECToR for 51M and 204M meshes and (b - bottom) performance for 51M, 204M and 816M meshes when PT-SCOTCH, ParMETIS and SFC are used as partitioners .

The performance of Code_Saturne is assessed on the three machines for the 51M and 204M cell mesh cases and results are presented in Figure 23 (a). A performance speed-up is observed in all the cases, even when going from 16,384 to 32,768 cores on Jugene. This

comparison also demonstrates Code_Saturne's portability on machines with various architectures, and compilers.
Figure 23 (b) shows the performance for the 51M, 204M, 816M and 3.2B cell meshes on the Cray XE6. Overall, best performance is usually achieved by PT-SCOTCH for the 51M, 204M and 816M case. Up-to-now only the SFC partitioning tool could be used for the 3.2B case. Code_Saturne was run on that case up to 65,536 cores and a speed-up was observed going from 32,768 to 65,536 cores.

6.2.2 *Parallel Mesh Multiplication for Code_Saturne*

Supported by: Pavla Kabelikova, Ales Ronovsky, Vit Vondrak (VSB, Czech Republic)

The main bottleneck to enable petascaling of Code_Saturne is the time required to generate large meshes. This is the case even for relatively modest sizes, e.g. 10 million cells, if the geometry is very complex and boundary layers have to be meshed. Therefore it is obvious that mesh generation of billion of cell meshes has to be parallelized, but no open-source parallel mesh generators are yet available. Therefore other routes must be followed and the one proposed by Code_Saturne developers deals with parallel global mesh refinement (or mesh multiplication), i.e. an initial mesh of about 100 million cells would be read by Code_Saturne and then each of its cells would be split. This process could be repeated several times in order for the Navier-Stokes solver to run on a several billion cell mesh, while post-processing would be carried out on the initial 100 million cell mesh.

This project has developed a parallel mesh multiplication package and integrated this to Code_Saturne in order to extend its capability to generate more than billion cell meshes. In the corresponding whitepaper the basic ingredients of implemented parallel mesh multiplication package are detailed and its performance and scalability tests on a local cluster at VSB-TU Ostrava (ComSio) and on Curie system are presented.

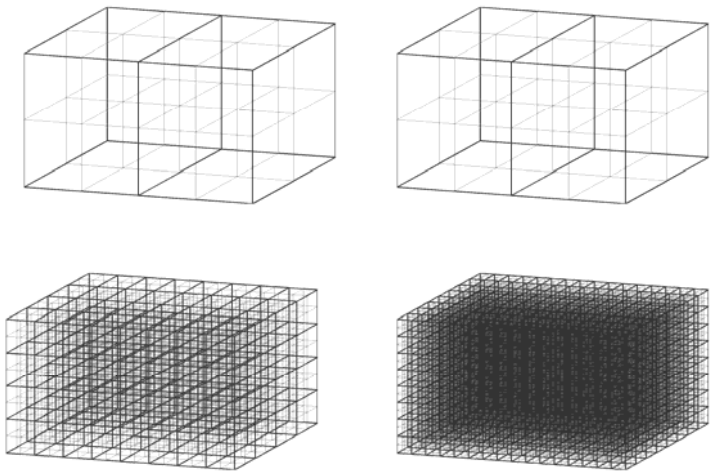

Figure 24: Subdomain mesh refinement

The subdomain mesh multiplication, shown in Figure 24, has been processed by our multiplication algorithm. The mesh refinement is executed in parallel on distributed parts of the mesh and it is done in local variables. During the final re-indexation of global mesh nodes indices, the special care has to be concentrated on faces on the subdomain interfaces that are shared with more processors. Very fine meshes can be obtained applying implemented mesh multiplication algorithm recursively. If required, the next levels of refinement can be processed in the same way as the first level.

Parameters of given mesh			Number of used cores					
level of refinement	no. of cells	no. of vertices	32 cores		64 cores		128 cores	
			time [s]	no. of vertices	time [s]	no. of vertices	time [s]	no. of vertices
0	16320	18070	<0.1	850	<0.1	550	<0.1	280
3	8.4M	8.5M	<0.2	280k	<0.1	150k	<0.1	76k
4	66.8	66M	1.5	2.2M	<1	1.1M	<1	600k
5	0.5B	0.5B	12	17M	6	8.6M	3.8	4.4M
6	4.3B	4.3B	---	---	---	---	28	34M

Table 11: Parallel Performance of Mesh Multiplication Method

The performance tests summarised in Table 11 demonstrate the good parallel scalability of the presented solution. Using the Curie cluster, we have obtained a refined mesh with more than 1 billion cells, vertices also, in less than 30 seconds.

Future improvements to optimize global indexation using other Code_Saturne internal structures and/or methods can lead to more effective computation of very fine meshes in smaller memory and communication requirements than now.

In the future work, the multiplication of other types of meshes would need to be implemented to complete the full mesh multiplication functionality. While the multiplication of tetrahedral and prismatic meshes is straightforward and can be done very easily extending implemented package with tetrahedral and pyramid multiplication, the refinement of pyramidal or hybrid meshes needs some more effort.

6.2.3 *Code Saturne - Optimizations in Preprocessing Step*

Supported by: A. Turk, C. Moulinec, A.G. Sunderland, C. Aykanat

In this work, the efforts towards understanding and improving the pre-processing subsystem of Code_Saturne are described, and to this end, the performance of different mesh partitioning software packages that can be used are investigated.

Previous studies show that sequential graph partitioner MeTiS [14] provide the best results in terms of reducing the average time spent in a timestep of Code_Saturne. However, MeTiS has problems in partitioning into 64K parts or more. Furthermore, for sufficiently large problems, the memory requirements of MeTiS far surpasses the memory available, even in fat nodes. In this document, the efforts towards understanding and improving the preprocessing subsystem of Code Saturne are described, and to this end, the performance of different mesh partitioning software packages that can be used have been investigated.

Analysis conducted during the studies reported in this document reveal that, for medium sized meshes, if the time spent for partitioning is not important, the usage of sequential SCOTCH [16] can be considered, since it provides reasonably good partitions with relatively low

memory requirements and can easily scale to 128K and beyond on a single fat node. As the mesh sizes that are to be partitioned reaches to billion-cell meshes, even the mesh generation has to be performed in parallel via parallel mesh generators. Thus, both in order to adjust to memory constraints and to avoid migration of data, the partitioning has to be done in parallel as well. To fit such large meshes in the memory of cluster nodes, the mesh has to be partitioned into very large number of cores. Unfortunately, the partitioning performance of parallel graph partitioning packages such as Par-MeTiS [15] and PT-SCOTCH decrease with increasing number of cores used in the partitioning process. To address this problem we propose to utilize a two-level hierarchical partitioning scheme that enables the usage of different partitioning schemes in each level. We investigate the different partitioning packages that are supported in Code Saturne and also propose a hierarchical partitioning scheme utilizing Zoltan [17]. We analyze aspects such as memory requirements, load- balancing performance of partitioners and the effect of partitioning quality over the runtime of Code Saturne.

A two-level hierarchical partitioning scheme that enables the usage of different partitioning schemes in each level is investigated as an alternative to the partitioning tools supported by Code Saturne. The proposed hierarchical partitioning scheme utilizes the Zoltan partitioning framework for combining different partitioning schemes. In this scheme, the graph is first partitioned into a quotient of the desired number of parts in the first level, and then, in the second level, each subgraph is separately partitioned further to obtain the final desired number of parts. In the first level of the hierarchical framework, the domain is partitioned using all the available cores into the number of available nodes via a graph partitioning tool such as Par-MeTiS or PT-SCOTCH. This way, all the memory in the nodes can be used. In the second level of the hierarchy, each node independently partitions the data allocated to itself into the number of cores in the node. In this second level partitioning, graph partitioning tools such as Par-MeTiS or PT-SCOTCH can be utilized as well as cheap geometrical or random partitioning schemes since the communication between the cores of a node is much faster.

Two different sets of experiments on two different datasets were conducted to compare the partitioning schemes and tools. In the first set of experiments we check the partitioning results of the tolls and in the second set of experiments we compare the execution time of Code Saturne when utilizing these partitions (see Table 12) using the SMALL dataset with 5.8M vertices. The tests for hierarchical scheme (HIER) are conducted on the BGP in STFC Daresbury since we wanted to preserve our quota in Jugene for the time we will have much larger (e.g. 2 Billion) meshes. In the runs for HIER, PAR-MeTiS is used in the first level to divide the graph in Number of parts/4 and then in the second level, Number of parts/4 separate calls are made to PAR-MeTiS to divide each subgraph into 4.

# of parts	HIER (s)	MeTIS (s)	SCOTCH (s)	ParMeTiS (s)	PT-SCOTCH (s)
256	13.45	12.37	12.10	12.76	12.51
512	7.64	5.92	7.04	7.23	7.14
1024	4.96	3.87	4.64	4.91	5.01
2048	3.59	3.21	3.26	3.38	3.45
4096	3.07	2.52	2.52	2.71	2.60

Table 12: Runtime per timestep (seconds) of Code Saturne
The partitions are obtained by partitioning the SMALL dataset with HIER, MeTiS, SCOTCH, PARMeTiS and PT-SCOTCH.

An analysis of Table 12 shows that the hierarchical scheme has slightly higher runtime per timestep values when compared to other schemes, but it has the potential to be able scale to

much larger number of cores then PAR-MeTiS and PT-SCOTCH so it is still interesting to investigate this scheme.

Collected experimental results indicate that proposed hierarchical scheme performs slightly worse than classical schemes but its advantages in partitioning into larger number of parts still make it a viable approach. Our future work includes improvements in the quality of the partitions obtained via our hierarchical scheme and analysis on much larger number of cores.

6.2.4 *Some Conclusions*

This project on Code_Saturne has demonstrated that the code is ready to run on petascale machines, as shown by the speed-up observed going from 32,768 to 65,536 cores for a 3.2B. The objectives of this project have been fulfilled, as 2B cell meshes had to be handled by the code and its performance has been assessed on 60,000+ cores. This work has also strongly benefited from two other projects carried out within PRACE-1IP WP7.6, the first one lead by VSB concerning mesh multiplication, and the second one by Bilkent University based on developing a hierarchical partitioning tool. Future work will involve testing the code on larger meshes, of up to 12-15B cell and assessing Code_Saturne scalability in the mixed mode MPI/OpenMP.

This work will enable the use of higher Reynolds numbers in simulations using a highly accurate LES model, allowing the study of significantly more realistic flows (closer to industrial ones). The functionality in the code remains available for both very large simulations and small ones (even serial), and can still be used by researchers who have only limited access to HPC computational resources.

7 Collaboration with Communities

Task 7.2 established a strong cooperation with the scientific communities to cooperate to enable new challenging HPC applications of their interest and that, once optimised, can be put at disposal of the whole scientific community. The activity was issued with both *independent* and *structured* communities.

7.1 Independent Communities

The independent communities are identified by groups of scientists investigating specific research topics, characterised by the use of similar application codes and adopting a common investigation methodology. Identifying these communities around Europe is not easy as usually specific governance rules and research programmes are not defined between the groups of scientists involved in the communities. In many cases these communities are identified with the suites of computational codes used. In this sense, task 7.2 working with the owners of codes GROMACS, Quantum ESPRESSO, GPAW and CP2K, sought to form collaborations with the Material Science and Life Science communities. The activity on DL-POLY and DALTON allowed relationships with the computational chemistry community to be created. Other communities represented were those of fluid dynamics and plasma physics.

During the activity of the project, Task 7.2 continued to further improve contacts with other independent communities astrophysics and computational cosmology, in particular. A small set of numerical HPC codes for cosmology has emerged in Europe: Dr. Volker Springel and the Virgo Consortium have developed a code named Gadget-2 which belongs to the PRACE benchmark suite. The version of the code available to the public is not up-to-date, so it was decided to not put effort in this application. Currently, the full-physics state-of-the-art version of the code (Gadget-3) is essentially distributed among the Virgo Consortium collaboration, which is the main organized body in the Computational Cosmology community. Currently many different versions of Gadget-3 are distributed, each of which having some different physical modules. Contacts have been established in order to try to set up a fruitful collaboration, but till now only few groups of Gadget-3 users have agreed to cooperate closely with us. Thus, it was not possible to form an explicit collaboration with the community in time to actively work on the code before the end of the Task 7.2.

The lack of time and the difficulties in finding the right representative of the community did not allow further collaborations with other independent communities , like the Fundamental Physics community, to be consolidated.

7.1 Structured Communities

Task 7.2 established collaborations with structured communities to investigate their applications for exploiting efficiently the PRACE Tier-0 infrastructure. The collaboration was issued firstly with the IS-ENES community (https://is.enes.org) and the MAPPER project (http://www.mapper-project.eu) then with the ScalaLife project [7] and the VERCE project [8].

7.1.1 *IS-ENES*

IS-ENES [5], promotes the development of a common distributed modeling research infrastructure in Europe in order to facilitate the development and exploitation of climate models and better fulfill the societal needs with regards to climate change issues. The

cooperation with the IS-ENES community was fruitful allowing to improve the scalability of the EC-Earth 3 suite, based on the IFS (atmosphere) and NEMO (ocean) models, coupled through the OASIS coupler code. The cooperation was a great occasion to allow IS-ENES researchers to work with PRACE experts of HPC methodologies. The activity and the results reached are reported in Chapter 5. Some of these codes are not well structured to scale efficiently on Tier-0 systems, so it was agreed to refactoring some codes in the perspective of using advanced HPC systems, continuing the collaboration with IS-ENES in PRACE-2IP.

7.1.2 *MAPPER*

MAPPER is a project funded by the EC in FP7 to develop computational strategies for multiscale simulations across disciplines, exploiting existing European e-Infrastructures. MAPPER has the objective to deploy a computational science environment for multiscale computing, cooperating with other research infrastructures and EU Projects. The cooperation between the MAPPER project and PRACE was issued firstly with WP6 to set up a test bed distributed infrastructure based on Tier-0, Tier-1 and grid infrastructures to test multi-scale applications of interest for MAPPER. Then some cooperation started with WP7 to evaluate possible forms of collaboration between the two projects in the context of scaling applications. The multiscale scientific applications of interest for the MAPPER community span over five challenging scientific domains: fusion, clinical decision making, systems biology, nanoscience and engineering. Task 7.2 analysed the codes proposed by MAPPER, some of them are already suitable for a Tier-0 environment and was suggest to apply for the PRACE Regular calls to get the computational resources.

7.1.3 *ScalaLife*

The ScalaLife project has the objective to develop new hierarchical parallelization approaches for codes of interest of the life science community. ScalaLife created a Competence Centre for scalable life science software. The cooperation with ScalaLife involved the petascaling activity in DALTON and GROMACS and allowed to further reinforce the common activity between the PRACE HPC experts and the main developers of the GROMACS and DALTON application codes. The petascaling activity and the results reached are described in details in Chapters 2 and 3.

7.1.4 *VERCE and EPOS*

The Earth Sciences community and, specifically the Geophysics community, are quite diverse and spreads across different disciplines and interests from earthquakes, to volcanology, surface dynamics and plate tectonics. Different research centres and national institutes are involved in specific research programs involving simulation activities using different models and applications. The earthquake and seismology research addresses both fundamental problems in understanding Earth's internal wave sources and structures, and augmenting applications for societal concerns about natural hazards, energy resources, environmental change, and national security. This community is central in the European Plate Observing System (EPOS), the ESFRI initiative in solid Earth sciences, recently consolidated in a research infrastructure.

VERCE is an European project aiming at providing to the earthquake and seismology research community in Europe, a data-intensive e-Science and HPC environment. VERCE was interested in cooperating to the petascaling activity of seismological code like SPECFEM3D. The VERCE Community is cooperating with the EPOS RI.

The contacts between PRACE, VERCE and EPOS started only some months ago when the work on SPECFEM3D was almost completed. The scalability of SPECFEM 3D represents a major interest for VERCE and EPOS. Some information on the scalability of the code have been discussed with VERCE with the intent to support the community to apply for Tier-0 resources in PRACE.

8 Summary

Task 7.2 had the main goal of identifying opportunities to enable applications of interest of selected scientific communities and specific application projects. During the two-years life of the project, a long term collaboration was established with scientific communities interested in computational methodologies to facilitate their exploitation of PRACE Tier-0 systems. The collaboration involved both *independent communities* (groups of scientists investigating specific research topics and characterised by using similar application codes and adopting a common investigation methodology) and *structured communities* (groups of researchers carrying out research in the same scientific area at a European level, with a common governance, structure and specific research objectives). The latter communities involved are IS-ENES (meteo-climatology), MAPPER (multi-scale problems), ScalaLife (life science) and recently VERCE and EPOS for computational seismology.

Eleven key application codes of value to these main scientific communities at European level have been first selected and then petascaled in Task 7.2. All the codes selected are European and cover the main computational disciplines: Molecular Dynamics, Material Science, Nano-sciences, Computational Chemistry, Plasma Physics Meteo-climatology, Earth Sciences, Engineering and CFD.

The petascaling actions in Task 7.2 covered a broad range of activities, from the identification of the bottlenecks and the lack of performance, to the optimization and restructuring of the algorithms and the data structures, to the implementation of smart parallelization methodologies, to the fine tuning of the codes and scalability tests on the Tier-0 architectures. Furthermore, a synergic cooperation was initiated with Task 7.5 and Task 7.6 to work on the codes with specific actions of their competence including; scalable libraries, hybrid coding on multi-many core systems and accelerators, parallel pre-processing, parallel and hierarchical I/O.

The main chapters of the deliverable summarise the activity done and the achievements reached for each application code. The whole work is documented in 12 white papers directed by Task 7.2, plus other 9 prepared in collaboration from Task 7.5 or Task 7.6 in collaboration with Task 7.2. The white papers will be published on the PRACE RI web site and will be available for the scientific communities [10].

The enabling activity allowed long-term collaborations to be established with the related communities and the owners of the codes, allowing fast progress to enhance new challenging applications that now can be made available to the whole scientific community to realise complex and innovative simulations on Tier-0 Systems. This result would not be possible without the close collaboration of the communities and the developers involved.

Of course the activity carried out in Task 7.2 was not always easy and sometimes did not give the expected results, in terms of performance improvements and efficiency, or required more effort than initially planned. In addition, in some cases it was not trivial to integrate new algorithms and methods in existing production codes.

However, the work resulted from the activity in Task 7.2 often achieved very positive outcomes for the owners of the codes and the communities. In some cases, the petascaling improvements have been incorporated as part of the official releases of the code (i.e. Quantum ESPRESSO, GPAW, DALTON) or the enabled code installed directly on the Tier-0 Systems (GROMACS, CP2K, Quantum ESPRESSO). In some other cases the positive collaboration will continue also after PRACE-1IP to improve and complete the undertaken activities. Some examples of continuing collaborations include:

- the OpenFOAM work on scalability will continue in PRACE 2IP-WP9 *Industrial Application Support*, given the interest for the industrial community to improve the OpenFOAM performances on Tier-0 and Tier-1 systems, based on the analysis and the achievements reached in Task 7.2.
- A refactoring of Quantum ESPRESSO has been agreed in PRACE-2IP WP8 *Community Code Scaling* to address the code toward the next generation of HPC Systems (i.e. Exascale).
- The collaboration with IS-ENES community continues in PRACE-2IP WP8, addressing somewhat different subjects: OASIS3-MCT, advanced features for the NEMO ocean model, etc.

In summary, the activity done in Task 7.2 produced a positive impact allowing eleven European flagship application codes of interest for the scientific communities to be optimised and ready for production on Tier-0 systems. It is worth noting that in different cases applications based on these codes have already been submitted successfully to the 4th PRACE regular access call and some other are under evaluation in the 5thcall, thanks to the work done in Task 7.2.