SEVENTH FRAMEWORK PROGRAMME
Research Infrastructures

INFRA-2012-2.3.1 – Third Implementation Phase of the European High Performance Computing (HPC) service PRACE

PRACE-3IP

PRACE Third Implementation Phase Project

Grant Agreement Number: RI-312763

D7.1.2
Final Report on Applications Enabling

Final

Version:	1.0
Author(s):	Alexander Schnurpfeil, FZJ; Xu Guo, EPCC; Maciej Szpindler, ICM
Date:	23.06.2014

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: RI-312763**	
	Project Title: PRACE Third Implementation Phase Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: **D7.1.2**	
	Deliverable Nature: Report	
	Deliverable Level: PU / *	**Contractual Date of Delivery:** 30 / June / 2014
		Actual Date of Delivery: 30 / June / 2014
	EC Project Officer: Leonardo Flores Añover	

* - 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:** PRACE Third Implementation Phase Project	
	ID: **D7.1.2**	
	Version: 1.0	**Status:** Final
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2010	
	File(s): D7.1.2.docx	
Authorship	**Written by:**	Alexander Schnurpfeil, FZJ; Xu Guo, EPCC; Maciej Szpindler, ICM

	Contributors:	Petri Nikunen, CSC Markus Rampp, RZG Evghenii Gaburov, SURFSara Cevdet Aykanat, BILKENT Eva Casoni, BSC Jacques David, CEA Gunduz V. Demirci, BILKENT John Donners, SURFsara Andrew Emerson, CINECA David Emerson, STFC Hans Eide, UiO Georgios Fanourgakis, CASTORC Damyan Grancharov, NCSA Massimiliano Guarrasi, CINECA Guillaume Houzeaux, BSC Nevena Ilieva, NCSA Maria Francesca Iozzi,UiO Mohammad Jowkar, BSC Christos Kannas, CASTORC Tomáš Karásek, VSB Klaus Klingmueller, CASTORC Soon-Heum Ko, LiU Elena Lilkova, NCSA Leandar Litov, NCSA Stoyan Markov, NCSA Charles Moulinec, STFC Peicho Petkov, NCSA Martin Plummer, STFC Thomas Ponweiser, JKU Thomas Röblitz, UiO Ole Widar Saastad, UiO Katerina Michalickova, UiO Nikolay Aleksandrov Vazov, UiO Georgios Magklaras, UiO Reha Oğuz Selvitopi, BILKENT Peter Stadelmeyer, JKU Andrew Sunderland, STFC Ilian Todorov, STFC Ata Turk, BILKENT Francesco, Savladore, CINECA Maciej Cytowski, ICM Mikael Rännar, SNIC-UmU Carlo Cavazzoni, CINECA Volker Weinberg, LRZ Anupam Karmakar, LRZ Luis Fazendeiro, SNIC-Chalmers Jeroen Engelberts, SURFsara Vegard Eide, NTNU Alexandra Charalampidou, GRNET Marcin, Lawenda, ICM, Joerg Hertzer, HLRS Miroslaw Kupczyk, PSNC

	Reviewed by:	Iva Nikolova, NCSA; Dietmar Erwin, FZJ
	Approved by:	MB/TB

Document Status Sheet

Version	**Date**	**Status**	**Comments**
0.1	07/May/2014	Draft	Set up structure of this Document
0.2	20/May/2014	Draft	Chapters 2, 3 and 4 included
0.3	27/May/2014	Draft	Projects reports included
0.4	03/June/2014	Draft	After task-internal reviews
0.5	22/June/2014	Draft	After project-internal review
1.0	23/June/2014	Final Version	

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure

Table of Contents

List of Figures

List of Tables

References and Applicable Documents

[1] http://www.prace-project.eu

[2] http://www.prace-project.eu/IMG/pdf/d7.1.pdf

[3] http://www.prace-ri.eu/IMG/pdf/wp121.pdf

[4] http://www.prace-ri.eu/IMG/pdf/wp122.pdf

[5] http://www.prace-ri.eu/IMG/pdf/wp123.pdf

[6] http://www.prace-ri.eu/IMG/pdf/wp124.pdf

[7] http://www.prace-ri.eu/IMG/pdf/wp125.pdf

[8] http://www.prace-ri.eu/IMG/pdf/wp126.pdf

[9] http://www.prace-ri.eu/IMG/pdf/wp127.pdf

[10] http://www.prace-ri.eu/IMG/pdf/wp115.pdf

[11] http://www.prace-ri.eu/IMG/pdf/wp116.pdf

[12] PRACE RI webpage for white papers: http://www.prace-ri.eu/white-papers?lang=en

[13] http://www.prace-ri.eu/IMG/pdf/wp167.pdf

[14] http://www.prace-ri.eu/IMG/pdf/wp168.pdf

[15] http://www.prace-ri.eu/IMG/pdf/wp169.pdf

[16] http://www.prace-ri.eu/IMG/pdf/wp170.pdf

[17] http://www.prace-ri.eu/IMG/pdf/wp171.pdf

[18] http://www.prace-ri.eu/IMG/pdf/wp172.pdf

[19] http://www.prace-ri.eu/IMG/pdf/wp173.pdf

[20] http://www.prace-ri.eu/IMG/pdf/wp174.pdf

[21] http://www.prace-ri.eu/IMG/pdf/wp175.pdf

[22] http://www.prace-ri.eu/IMG/pdf/wp177.pdf

[23] https://software.intel.com/sites/products/documentation/hpc/mkl/mklman/GUID-2C5C30E9-2B33-4F4E-AD5D-94398CA57FAD.htm

[24] D7.1.1 Applications Addressing Major Socio-economic Challenges (25.05.2013), http://www.prace-ri.eu/IMG/pdf/d7.1.1.pdf

[25] UKRmol: J. M. Carr, P. G. Galiatsatos, J. D. Gorfinkiel, A. G. Harvey, M. A. Lysaght, D. Madden, Z. Mašín, M. Plummer, J. Tennyson, and H. N. Varambhia, Eur. Phys. J. D 66, 58 (2012)

[26] A. Dora, L. Bryjko, T. van Mourik and J. Tennyson, J. Chem. Phys. 146 (2012) 024324

[27] Z. Masin and J. D. Gorfinkiel, J. Chem. Phys. 137, 204312 (2012).

[28] Atomic, Molecular, Optical and Positron Physics Group, University College London, http://www.ucl.ac.uk/phys/amopp

[29] P. Petkov, S. Markov, I. Todorov, "Development of AGBNP2 Implicit Solvent Model Library for MD Simulations", PRACE White Paper, http://www.prace-ri.eu/IMG/pdf/wp111.pdf

[30] G. Houzeaux, J. Principe, "A Variational Subgrid Scale Model for Transient Incompressible Flows", Int. J. Comp. Fluid Dyn., 22(3), 135–152, 2008.

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

[32] R. Lohner, F. Mut, J. Cebral, R. Aubry, G. Houzeaux, “Deflated Preconditioned Conjugate Gradient Solvers for the Pressure-Poisson Equation: Extensions and Improvements”, Int. J. Num. Meth. Eng., 87(1-5), 2-14, 2010.

[33] K. S. Mujumdar and V. V. Ranade, “Simulation of rotary cement kilns using a one dimensional model”, Chem. Eng. Res. Des., vol. 84, pp. 165–177, 2006.

[34] C. K. Birdsall, A. B. Langdon, Plasma physics via computer simulation, Taylor & Francis, 2004.

[35] Y. Saad, Iterative methods for sparse linear systems, Society for Industrial and Applied Mathematics, 2003.

[36] G. Lapenta, J. Brackbill, P. Ricci, Kinetic approach to microscopic-macroscopic coupling in space and laboratory plasmas, Physics of plasmas 13 (2006) 055904.

[37]] H. Vu, J. Brackbill, Celest1d: an implicit, fully kinetic model for low-frequency, electromagnetic plasma simulation, Computer physics communications 69 (1992) 253-276.

[38] A. Beck, M. Innocenti, G. Lapenta, S. Markidis, Multi-level multi-domain algorithm implementation for two-dimensional multiscale particle in cell simulations, Journal of Computational Physics (2013).

[39] https://code.google.com/p/likwid/

List of Acronyms and Abbreviations

AISBL	Association International Sans But Lucratif (legal form of the PRACE-RI)
AMD	Advanced Micro Devices, Inc.
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BILKENT	Bilkent University (Turkey)
BLAS	Basic Linear Algebra Subprograms
BSC	BarcelonaSupercomputing Center (Spain)
BSCW	Basic Support for Cooperative Work; a web based system that offers shared workspaces
CaSToRC	Computation-based Science and Technology Research Center (Cyprus)
CEA	Commissariat à l'EnergieAtomique (represented in PRACE by GENCI, France)
CHALMERS	Chalmers University of Technology (Sweden)
CaSToRC	Computation-based Science and Technology Research Center
CINECA	Consorzio Interuniversitario, the largestItaliancomputing centre (Italy)
CINES	Centre Informatique National de l'Enseignement Supérieur (represented in PRACE by GENCI, France)
CPU	Central Processing Unit
CSC	Finnish IT Centre for Science (Finland)
CSCS	The Swiss National Supercomputing Centre (represented in PRACE by ETHZ, Switzerland)
CUDA	Compute Unified Device Architecture
DECI	Distributed European Computing Initiative
DEISA	Distributed European Infrastructure for Supercomputing Applications. EU project by leading national HPC centres.
DGEMM	Double precision General Matrix Multiply
DP	Double Precision, usually 64-bit floating point numbers
DPMDB	DECI Project Management Database; a web-based application to view and edit details of DECI proposals and projects
EC	European Community
EPCC	Edinburg Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)
FFT	Fast Fourier Transform
FPU	Floating-Point Unit
FZJ	Forschungszentrum Jülich (Germany)
GB	Giga (= 2^{30} ~ 10^9) Bytes (= 8 bits), also GByte
GB/s	Giga (= 10^9) Bytes (= 8 bits) per second, also GByte/s
GCS	Gauss Centre for Supercomputing (Germany)
GÉANT	Collaboration between National Research and Education Networks to build a multi-gigabit pan-European network, managed by DANTE. GÉANT2 is the follow-up as of 2004.
GENCI	Grand Equipement National de CalculIntensif (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
GNU	GNU's not Unix, a free OS
GPGPU	General Purpose GPU
GPU	Graphic Processing Unit

GRNET	Greek Research and Technology Network (Greek)
HDF5	Hierarchical Data Format
HLRS	Höchstleistungsrechenzentrum Stuttgart (Germany)
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
IBM	Formerly known as International Business Machines
ICHEC	Irish Centre for High-End Computing (represented in PRACE by NUI Galway)
ICM	Interdisciplinary Centre for Mathematical and Computational Modelling (Warsaw, Poland)
IDRIS	Institut du Développement et des Ressources en Informatique Scientifique (represented in PRACE by GENCI, France)
I/O	Input/Output
IP	Implementation Project
IPB	Institute of Physics, Belgrad (Serbia)
ISC	International Supercomputing Conference; European equivalent to the US based SC0x conference. Held annually in Germany.
JKU	Johannes Kepler University (Linz, Austria)
JSC	Jülich Supercomputing Centre (FZJ, Germany)
KB	Kilo (= 2^{10} ~10^3) Bytes (= 8 bits), also KByte
KTH	Kungliga Tekniska Högskolan (represented in PRACE by SNIC, Sweden)
LiU	Linköping University (Sweden)
LQCD	Lattice QCD
LRZ	Leibniz Supercomputing Centre (Garching, Germany)
LU	Lund University (Sweden)
MB	Mega (= 2^{20} ~ 10^6) Bytes (= 8 bits), also MByte
MB/s	Mega (= 10^6) Bytes (= 8 bits) per second, also MByte/s
MFlop/s	Mega (= 10^6) Floating point operations (usually in 64-bit, i.e. DP) per second, also MF/s
MHz	Mega (= 10^6) Hertz, frequency =10^6 periods or clock cycles per second
MKL	Math Kernel Library (Intel)
MPI	Message Passing Interface
NCSA	National Centre for Supercomputing Applications (Sofia, Bulgaria)
NCF	Netherlands Computing Facilities (Netherlands)
NIIF	National Information Infrastructure Development Institute (Hungary)
NTNU	Norges teknisk-naturvitenskapelige universitet (Norway)
NUI	National University of Ireland
NUMA	Non-Uniform Memory Access or Architecture
OpenCL	Open Computing Language
OpenMP	Open Multi-Processing
OS	Operating System
PA	Preparatory Access
PA C	Preparatory Access Type C
PDC	Center for High Performance Computing, at KTH (represented in PRACE by SNIC, Sweden)
PETSc	Portable, Extensible Toolkit for Scientific Computation
PGI	Portland Group, Inc.
PI	Principal Investigator
PM	Person month
POSIX	Portable OS Interface for UNIX

PPM	Portable pixmap format
PPR	Project Proposal and Reporting
PRACE	Partnership for Advanced Computing in Europe; Project Acronym
PRACE-2IP	Second implementation phase of PRACE
PRACE-3IP	Third implementation phase of PRACE
PRACE-RI	PRACE Research Infrastructure
PSNC	Poznan Supercomputing and Networking Centre (Poland)
QCD	Quantum Chromodynamics
RAM	Random Access Memory
RI	Research Infrastructure
ROMIO	High-Performance, Portable MPI-IO Implementation
RZG	Rechenzentrum Garching (Garching Computing Centre, of the Max Planck Society, represented in PRACE by GCS, Germany)
SE	Scientific Evaluation
SHMEM	Share Memory access library (Cray)
SMP	Symmetric MultiProcessing
SNIC	Swedish National Infrastructure for Computing (Sweden)
SPH	Smoothed Particle Hydrodynamics
SSC	PRACE Scientific Steering Committee
STFC	Science and Technology Facilities Council (represented in PRACE by EPSRC, United Kingdom)
SURFsara	Dutch national High Performance Computing & e-Science Support Center
TB	Tera (= 240 ~ 1012) Bytes (= 8 bits), also TByte
TDDFT	Time-Dependent Density Functional Theory
TE	Technical Evaluation
TFlop/s	Tera (= 1012) Floating-point operations (usually in 64-bit, i.e. DP) per second, also TF/s
Tier-0	Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1
Tier-1	See Tier-0
UHeM	Istanbul Technical University National Center for High Performance Computing (=UYBHM)
UiO	Universitetet i Oslo (represented in PRACE by SIGMA, Norway)
UmU	Umea universitet (Sweden)
UU	Uppsala University (Sweden)
VSB-TUO	Vysoká škola báňská – Technická univerzita Ostrava (Technical University of Ostrava, Czech Republic)
WCSS	Wroclaw Centre for Networking and Supercomputing (Poland)
WP	Work Package
VTK	The Visualization Toolkit

Executive Summary

Task T7.1 “Scaling and Optimisation of Applications Codes” in Work Package 7 of PRACE-3IP aims to provide application enabling support for the HPC applications which are important for the European researchers to ensure the applications can effectively exploit multi-petaflop systems. There were three activities in T7.1:

T7.1.A Petascaling & Optimisation Support for Preparatory Access Projects: this activity provided code enabling and optimisation to European researchers as well as industrial projects to make their applications ready for Tier-0 systems. Projects can continuously apply for such services via the Preparatory Access Call Type C (PA C) with a cut-off every three months for evaluation of the proposals. Five Preparatory Access Calls have been carried out in PRACE-3IP and a total of 10 PA C projects have finished their work. The report focused on the optimization work and results gained by the completed projects in PRACE-3IP. The statistics about the PA C calls in PRACE-3IP as well as a description of the call organization itself is also included. The results of the completed projects have also been documented in white papers which were published on the PRACE-RI website [12].

T7.1.B Applications Support for DECI Projects: this activity in PRACE-3IP continued providing the technical support to DECI projects on Tier-1 systems since August 2013 after the PRACE-2IP ended. 15 partner sites were involved providing 73 PMs in total for the DECI technical support. In the 2nd year of PRACE-3IP, 180 Technical Evaluations (TEs) were performed, 119 for DECI-11 and 61 for DECI-12 applications. Technical support was provided for 31 DECI-9 and 37 DECI-10 projects which started in PRACE-2IP and were continued in PRACE-3IP. Full support was provided for 52 DECI-11 projects, and the starting assistance was provided for the 34 DECI-12 projects. Application enabling support was provided for four of the DECI projects and the enabling work done are reported in this deliverable.

T7.1.C Major socio-economic challenges and associated applications: this activity focused on enabling applications addressing key socio-economic challenges. The identified challenges included safe and environmental-friendly energy production, rational drug design, sustainable food supply, future aircraft transportation, ‘big data’ management and processing, understanding of climate change and natural environment protection. Eleven projects have been completed to address the given challenges and support over 20 recognised community codes. The enabling work focused on: application suites enabling for a multi-discipline modelling with coupling separate codes on the selected HPC platforms, improving computational kernels and scalable implementations. Following on the D7.1.1 in PRACE-3IP which reported on the identification of the challenges and codes, this deliverable includes the reporting on the completed projects as well as indications for future improvements and potential benefits for the application communities. Results and outcomes of these projects have also been presented with more technical details in the dedicated white papers.

1 Introduction

Computational simulations have proved to be a promising way to find answers to research problems from a wide range of scientific fields. However, complex problems often imply high demands regarding the needed computation time which cannot be satisfied by conventional computer systems. Instead, supercomputers are the method of choice in today’s simulations.

PRACE offers a wide range of different Tier-0 and Tier-1 architectures to the scientific community as well as to industrial projects. The efficient usage of such systems places high demands on the used software packages and in many cases advanced optimization work has to be applied to the codes to make best use of the provided supercomputers. The complexity of supercomputers requires a high level of experience and advanced knowledge of different kinds of concepts regarding programming techniques, parallelization strategies, etc. Such demands often cannot be covered by the applicants themselves but special assistance of supercomputing experts is needed. PRACE offers such a service through the Preparatory Access Call type C (PA C) for Tier-0 systems. PA C is managed by Task 7.1.A "Petascaling and Optimization Support for Preparatory Access Projects". This includes: the evaluation of the PA C proposals and the assignment of PRACE experts to these proposals. Furthermore, the support work itself is performed and monitored within this task. Section 2.1 gives a more detailed description and shows statistics on the usage of PA C in PRACE-3IP. The review process, the assignment of PRACE experts to the projects, and the monitoring of the support work is explained. Finally, the work done within the projects along with the outcome of the optimization work is presented in Section 2.8, Section 2.9 and Section 2.10.

In order to cover the users' needs regarding computation time and application support on Tier-1 systems the "Distributed European Computing Initiative (DECI)" has been integrated into PRACE since PRACE-2IP as a follow-on activity to the previous successful "DEISA Extreme Computing Initiative". This is managed by task 7.1.B "Applications Support for DECI Projects". T7.1.B continues the support for DECI projects after PRACE-2IP ended since August 2013, and mainly focuses on providing the technical/enabling support for the accepted DECI projects. The work done and the outcome is summarized in Section 0, including a brief overview of T7.1.B (Section 3.1), the technical support and applications enabling reporting for the DECI-9/DECI-10/DECI-11 enabling projects (Section 3.2, Section 3.3, and Section 3.4). Furthermore, general information on the following DECI calls is given in this section.

Finally, the current deliverable describes task 7.1.C "Socio-economic challenges" of WP7.1. This task focuses on the identification of applications which could potentially benefit from using HPC systems whereby it addresses key socio-economic challenges. It concentrates on enabling identified applications for efficient HPC systems usage or improving performance and scalability of the codes already used on HPC platforms for a specific purpose that answers the need of a given socio-economic problem. In addition to that, major socio-economic challenges which should be supported with PRACE infrastructure have been defined. These challenges are representing the fields having substantial impact on our live, society and economy. The following areas have been defined: Energy Sources and Management, Life Sciences and Medicine, Climate Change, Big Data, Environment Protection, and Engineering. Section 4 contains the description of the application support projects, the final report on the enabling of the selected applications and summarizes overall achievements and outcomes from the technical perspective and benefits for addressed scientific problem representing socio-economic challenge. Section 4.4 covers the overall description of the addressed socio-economic challenges including the summary of the selection process and reports in details the enabling projects associated with the selected set of application codes.

2 T7.1.A Petascaling & Optimisation Support for Preparatory Access Projects – Preparatory Access Calls

Access to PRACE Tier-0 systems is managed through PRACE regular calls which are issued twice a year. To apply for Tier-0 resources the application must meet technical criteria concerning scaling capability, memory demands, and runtime set up. There are many important scientific and industrial applications which do not meet these criteria. To support the researchers PRACE offers the opportunity to test and optimize their applications on the envisaged Tier-0 system prior to applying for a regular production project. This is the purpose of the Preparatory Access Call. The PA Call is a continuous call with a cut-off every three months for evaluation of the proposals. Therefore, new projects obtain access for preparatory purposes to PRACE Tier-0 systems each quarter. It is possible to choose between three different types of access:

- Type A is meant for code scalability tests to include the outcome in the proposal for a future PRACE Regular Call. Users receive a limited number of core hours, the allocation period is two months.
- Type B is intended for code development and optimization by the user. Users get also a small number of core hours; the allocation period is 6 months.
- Type C is also designed for code development and optimization with the core hours and the allocation period being the same as for Type B. The important difference is that Type C projects get special support by PRACE to support the optimization requests. In addition to access to the Tier-0 systems the applicants apply for 1 to 6 PMs of supporting work to be performed by PRACE experts.

All Tier-0 systems are available for PA. Currently these are the following systems:

- CURIE, BULL Bullx cluster at GENCI-CEA, France (thin, fat, and hybrid nodes are available)
- FERMI, IBM Blue Gene/Q at CINECA, Italy
- HERMIT, CRAY XE6 at GCS-HLRS, Germany
- JUQUEEN, IBM Blue Gene/Q at GCS-JSC, Germany
- MARENOSTRUM, IBM System X iDataplex at BSC, Spain
- SUPERMUC, IBM System X iDataplex at GCS-LRZ, Germany

2.1 Cut-off statistics

In PRACE-3IP four cut-offs for PA took place. Here, suitable projects are identified by a technical review process and experts from the PRACE project are assigned to these projects to support the optimization work. Despite the fact that Cut-off March 2013 took place in PRACE-2IP it is included in the presented statistics because the projects were taken over from PRACE-3IP by the end of PRACE-2IP and a considerable amount of work was carried out in the frame of the current implementation phase.

This section gives an overview on the outcome of the projects at each cut-off. The five cut-offs took place on the following dates: March 1st 2013, June 3rd 2013, September 2nd 2013, December 2nd 2013 and March 3rd 2014.

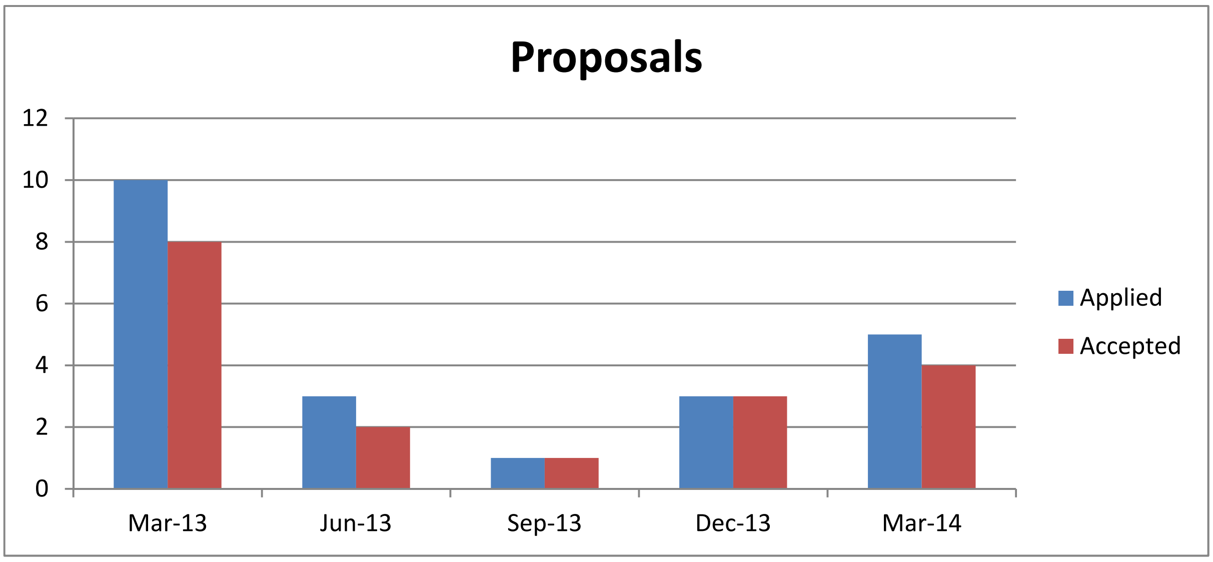


Figure 1: Number of Submitted and accepted proposals for PA type C per Cut-off.

Figure 1 presents the number of proposals and the number of projects which have been accepted for each Cut-off. In total 18 out of 22 proposals were accepted.

In Figure 2 the amount of PMs assigned to the projects per Cut-off is shown. In total 53 PMs are made available to the projects in PRACE-3IP.

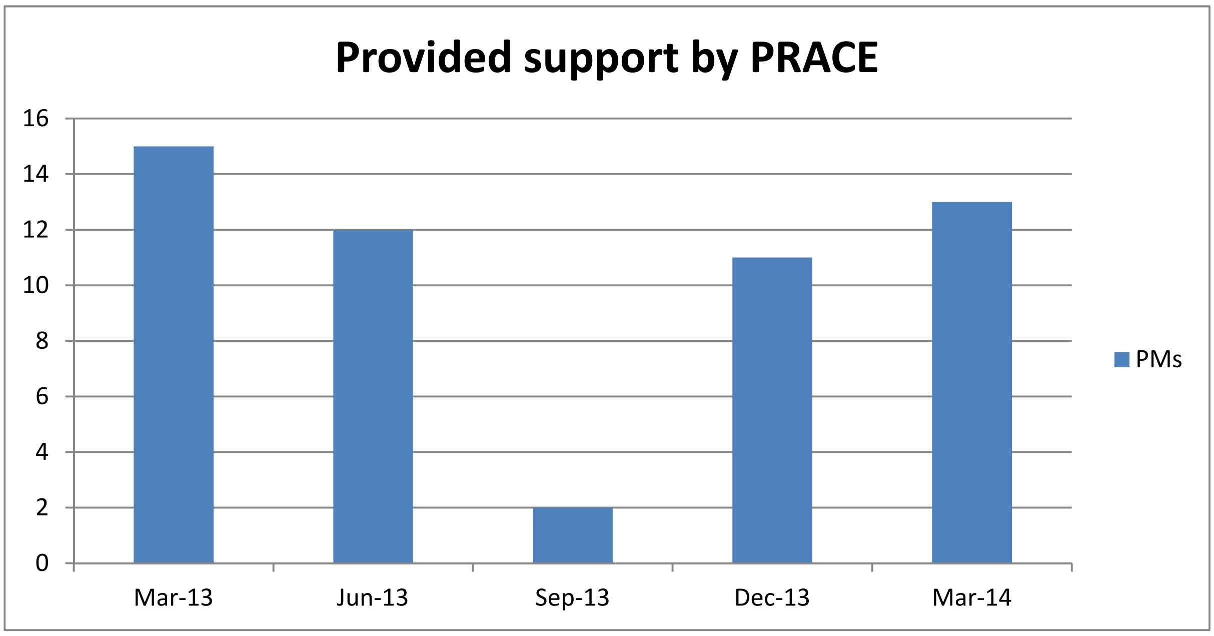


Figure 2: Amount of PMs assigned to PA type C projects per Cut-off.

Since projects from Cut-off December 2013 and June 2014 go beyond the lifetime of

The original schedule foresaw that WP7 would end in project month M24. To bridge the gap till the start of a planned follow-on project it was decided and included in a Contract Amendment that the work will be extended to M31 within the available resources. The period after M24 will be referred to a 3IP extension period - for short, although no extension for the PRACE project as such is needed; it has a duration of 48 month.

PMs dedicated to these projects will partly be used up in PRACE-3IP as well as in the 3IP extension phase.

Finally, it is also worth to give an overview about the scientific areas which are covered by the supported projects. This information is given in Figure 3.

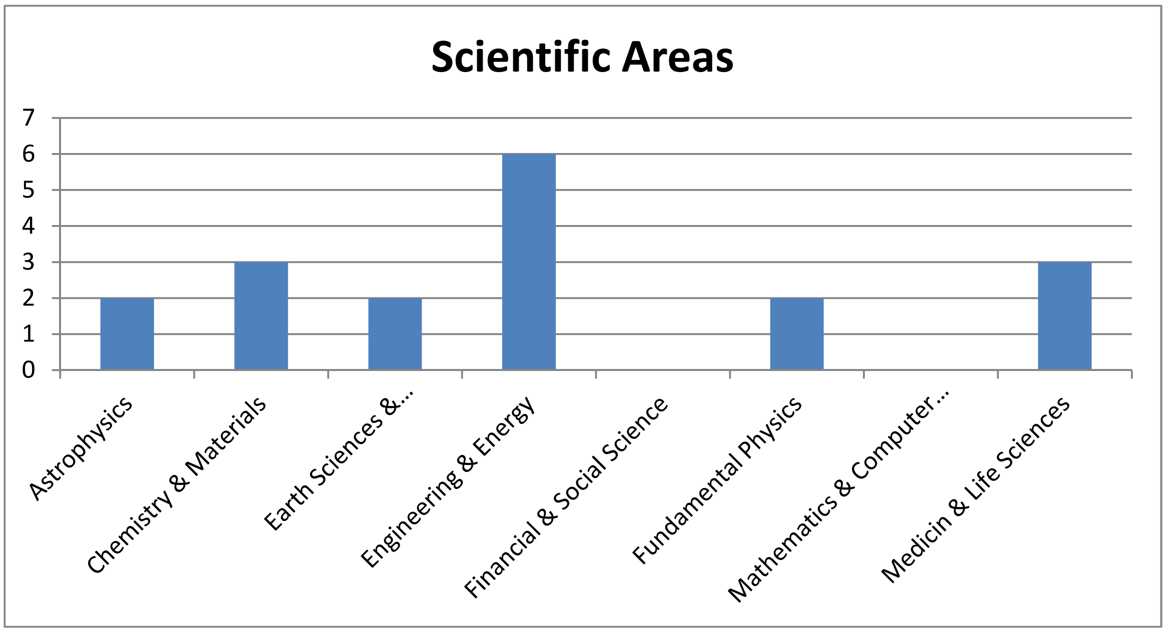


Figure 3: Number of projects per scientific field.

2.2 Review Process

The management of the review procedure, the assignment of PRACE collaborators and the supervision of the PA C projects are handled by task 7.1.A. In this section the review process for the preparatory access proposals of Type C is explained.

All preparatory access proposals undergo a technical review performed by technical staff of the hosting sites to ensure that the underlying codes are principally able to run on the requested system. In parallel, all projects are additionally reviewed by work package 7 in order to assess their optimization requests. Each proposal is assigned to two WP7 reviewers. The review is performed by PRACE partners who all have a strong background in supercomputing. Currently a list of 37 experts is maintained and the task leader has the responsibility to contact them to launch the review process. As the procedure of reviewing proposals and establishing the collaboration of submitted projects and PRACE experts takes place four times a year it is necessary to keep the review process timely and efficient. A close collaboration between AISBL, T7.1.A and the hosting sites is important in this context. The review process for technical and WP7 review is limited to two weeks. In close collaboration with AISBL and the hosting sites the whole procedure from PA cut-off to project start on PRACE supercomputing systems is carried out in less than six weeks.

Based on the proposals the Type C reviewers need to focus on the following aspects:

- Does the project require support for achieving production runs in the chosen architecture?
- Are the performance problems and their underlying reasons well understood by the applicant?
- Is the amount of support requested reasonable for the proposed goals?
- Will the code optimisation be useful for a broader community, and is it possible to integrate the development results achieved during the project in the main release of the code(s)?
- Will there be restrictions in disseminating the results achieved during the project?

Additionally, the task leader decides on the question whether the level and type of support requested is still available within PRACE. Finally the recommendation from WP7 to accept or reject the proposal is made.

Based on the provided information from the reviewers the Board of Directors has the final decision on whether proposals are approved or rejected. The outcome is communicated to the applicant through AISBL. Approved proposals receive the contact data of the assigned PRACE collaborators, refused projects are provided with further advice on how to address the rejection reasons.

2.3 Assigning of PRACE collaborators

To ensure the success of the projects it is essential to assign suitable experts from the PRACE project. This means based on the described optimization issues and support requests from the proposal experts are chosen who are most familiar with the subject.

This is done in two steps: First, summaries of the proposals describing the main optimization issues are distributed via corresponding mailing list. Here, personal data are explicitly removed from the reports to keep the anonymity of the applicants. Interested experts can get in touch with the task leader offering to work on one or more projects.

In case the response is not sufficient to cover the support requests of the projects, the task leader contacts the experts directly and asks them to contribute. In order to identify suitable collaborators a list of experts is maintained along with their special fields of expertise.

There is one exception to the procedure in the case when a proposal has a close connection to a PRACE site which e.g. already worked on the code: In this case this site is asked first if they are able to continue this collaboration in the context of the new PA C project.

This procedure has proved to be extremely successful; only one proposal from Cut-off March 2014 had to be refused due to lack of suitable support so far.

To be able to manage the whole review process within six weeks the assignment of PRACE experts takes place concurrently with the review process. This has shown to be a suitable approach. The overhead resulting in the assignment of projects that are rejected in the end is negligible.

After the review process described in section 2.2 is finished the support experts are introduced to the PIs and can start the work on the projects. The role of the PRACE collaborator includes the following tasks:

- Preparing a detailed work plan together with the applicant,
- Participating in the optimization work,
- Reporting the status report in the phone conference every second month,
- Participating in the writing of the final report together with the PI (the PI has the main responsibility for this report), due at project end and requested by the PRACE office,
- Writing a white paper containing the results which is published on the PRACE web site.

2.4 Monitoring of projects

Another task is the supervision of the Type C projects. This turns out to be a challenge as the projects' lifetimes (six months) and the intervals of the cut-offs (3 months) differ. This means that projects do not necessarily start and end at the same time but overlaps exist, i.e. at each point in time different projects might be in different states. Therefore, a phone conference take place in task 7.1.A every two month to discuss the status of running projects, to give advice on how to proceed with new projects and to manage the finalization and reporting of finished projects.

The conference call addresses all PRACE collaborators who are involved in these projects. All the project relevant information is maintained on a PRACE wiki page which is available to all PRACE collaborators.

Additionally the T7.1.A task leader is available to address urgent problems and additional phone conferences are held in such cases.

Bi-yearly a WP7 Face-to-Face meeting is scheduled. This meeting gives all involved collaborators the opportunity to discuss the status of the projects and to exchange their experiences.

2.5 Hand-over between PRACE-2IP and PRACE-3IP PA type C projects

The support for Preparatory Access Type C projects has been and is part of all PRACE projects (PRACE-1IP, -2IP, -3IP, 3IP Extension). For the hand-over between the projects the tasks decided on the following exact dates.

The hand-over between PRACE-2IP and PRACE-3IP PA type C projects took place right at the end of PRACE-2IP, August 31, 2013. The cut-off which took place in March 2013 was still under the responsibility of T7.1 in PRACE-2IP. Because the approved projects ran until October 2013, i.e. beyond PRACE-2IP, PRACE-3IP-experts were also needed to support these projects. The optimization work was started in PRACE-2IP but finished in PRACE-3IP. For these projects an interim report was given in deliverable D7.1 [2]. The final results are described in this deliverable.

2.6 PRACE Preparatory Access type C projects covered in the 3IP extension

Projects from Cut-off December 2013 and beyond will not finish their work within PRACE-3IP as they were only recently established. Instead the subsequent deliverable D7.1.3 will cover the final reports of these projects. Currently projects from Cut-off December 2013 and Cut-off March 2014 are affected by this arrangement. These projects are listed in Table 1.

Cut-offs December 2013/March 2014	
Title	Development of a package for computer aided drug design
Project leader	Miroslav Rangelov
PRACE expert	Joerg Hertzer
PRACE facility	HERMIT, JUQUEEN
PA number	2010PA2141
Project's start	03-Feb-14
Project's end	02-Aug-14
Title	Very high resolution Earth System Model (CESM) energy flux and wind stress sensitivity experiments
Project leader	Markus Jochum

Cut-offs December 2013/March 2014	
PRACE expert	Mads Ruben Burgdorff Kristensen
PRACE facility	FERMI, JUQUEEN
PA number	2010PA2066
Project's start	03-Feb-14
Project's end	02-Aug-14
Title	Improving the scalability of the overlapping fragments method code for electronic structure of organic materials
Project leader	Nenad Vukmirovic
PRACE expert	Petar Jovanovic
PRACE facility	CURIE TN, HERMIT
PA number	2010PA2132
Project's start	03-Feb-14
Project's end	02-Aug-14
Title	Parallel mesh partitioning in Alya
Project leader	Guillaume Houzeaux
PRACE expert	Mohammad Jowkar
PRACE facility	MARENOSTRUM
PA number	2010PA2171
Project's start	15-Apr-14
Project's end	14-Oct-14
Title	HORSE: High-order method for a new generatiOn of LaRge eddy Simulation solvEr
Project leader	Jean-François Boussuge
PRACE expert	Adrien Assange
PRACE facility	CURIE TN
PA number	2010PA2194
Project's start	21-Apr-14
Project's end	20-Oct-14
Title	Performance of the post-Wannier Berry-phase code for the

Cut-offs December 2013/March 2014	
	anomalous Hall conductivity calculations
Project leader	Malgorzata Wierzbowska
PRACE expert	Thomas Ponweiser
PRACE facility	SUPERMUC
PA number	2010PA2231
Project's start	30-Apr-14
Project's end	29-Oct-14
Title	Memory optimization for the Octopus scientific code
Project leader	Angel Rubio
PRACE expert	Alexandra Charalampidou
PRACE facility	MARENOSTRUM
PA number	2010PA2216
Project's start	15-Apr-14
Project's end	14-Oct-14

Table 1: Projects which were established in PRACE-3IP but will be finalised in the extension phase of PRACE-3IP.

2.7 Dissemination

The task uses different channels for dissemination. For each Preparatory Access call the PRACE sites are asked to distribute an email to their users to advertise preparatory access and especially the possibility of dedicated support via PA C. A template for this email was created in PRACE-2IP.

In the PRACE annual report for 2013 Preparatory Access Type C again was highlighted as unique opportunity to improve code performance and for getting ready for production usage on PRACE Tier-0 resources.

Also each successfully completed project should be made known to the public and therefore the PRACE collaborators are asked to write a white paper about the optimization work carried out. These white papers are published on the PRACE web site [1] and are also referenced in this deliverable.

2.8 Cut-off March 2013

This and the two following sections describe the optimization work carried out on the Preparatory Projects type C. The projects are listed in accordance with the cut-off dates they appeared. General information regarding the optimization work done in addition to the gained results is presented here using the recommended evaluation form. The application evaluation form ensures a consistent and coherent presentation not only of those projects which were managed in the frame of PA C but also those ones which ran under DECI (T7.1.B) and socio-economic challenges (7.1.C) as well. In addition to that the white papers created by these projects are referenced so that the interested reader is able to get further information.

Projects from the March 2013 cut-off started in PRACE-2IP, but the final work has been taken over by PRACE-3IP after the end of PRACE-2IP.

2.8.1 *Enabling Xnavis (URANS solver for fluid-dynamics) for massively parallel simulations of wind farms, 2010PA1461*

Code general features

Name	*χnavis*
Scientific field	Computational Fluid Dynamics
Short code description	*χnavis* is a general purpose solver for Computational Fluid Dynamics (CFD) developed at CNR-INSEAN. The solver is based on the finite volume discretization of the unsteady incompressible Navier-Stokes equations. The code is based on a multi-block approach; complex geometries and multiple bodies in relative motion are handled employing an in-house dynamical overlapping grid approach. Different CFD simulation types are implemented (RANS, LES, DES, DDES) as well as many of the most popular turbulence models (e.g. Boussinesq, k-ε, k-ϖ, Spalart-Almaras). The code has been validated in a number of simulations, mainly in the framework of naval hydrodynamics i , aerodynamics ii and renewable energyiii and it has been proven to scale when running on order of hundreds of cores (i.e. Tier-1 platforms).
Programming language	Fortran
Supported compilers	Intel, GNU, IBM XL
Parallel implementation	Hybrid MPI and OpenMP: the blocks are statically assigned to the available MPI processes while the computational work within each block is spread among the available OpenMP threads.
Accelerator support	N/A
Libraries	N/A
Building procedure	Makefile (different options available, e.g. compiler, optimization and debugging enabling)
Web site	N/A
Licence	Restricted (subject to an agreement with CNR-INSEAN that regulates the use of the code and the extension of the license)

Table 2: Code general features for χnavis.

Main objectives:

The objective of this work is to make the entire operating sequence required to run χnavis capable of exploiting massively-parallel architectures, such as PRACE Tier-0 ones. The main tasks are summarized as follows.

1) Load Balancing pre-processor: development of an automatic tool to assign blocks to processes

2) Overset pre-processor: refactoring to minimize the usage of memory and MPI parallelization

3) Parallel I/O: implementing MPI-I/O to reduce the number of files and possibly improve the I/O performances

4) Main solver performances: assess the scalability of the main *χnavis* solver on a Tier-0 architecture (Blue Gene/Q)

Accomplished work:

1) Load Balancing pre-processor: an automatic load balance pre-processor has been developed from scratch using an object-oriented programming style (Fortran 2003). The tool allows to handle the blocks/processes assignment procedure and it is crucial when dealing with huge grids.
2) Overset: the problem was again the use of large grids, which causes troubles in terms of memory requirement and wall clock time required. To overcome these problems an efficient parallel version of the overset pre-processor has been successfully developed.
3) MPI-I/O: Effort has been put for implementing the MPI - I/O operation onto the Fortran code. Now all processors may access a single file for reading its own partition and writing the solution at those blocks.
4) Analysis of the scalability properties of the parallel code on large and medium grids, with fixed or moving grids, has been performed. This work is mandatory for the application to a regular PRACE call.

Main results:

The automatic load balance pre-processor *LoadBalance* has been proven to be effective and reliable. Large number of grid blocks as well as high number of MPI processes may be efficiently handled and the procedure may be completely automatic. When running the tool, additional options may be set to optimize the block split/assignment algorithm, though. In particular, *LoadBalance* has been able to achieve satisfactory balancing (below 5%) with a total workload ranging from 10 to 80 million of cells, with a total number of blocks ranging from 50 to 500 and with a number of (MPI) processes up to 256. It is worth noting that that the load balance tool enabled a very efficient use of the Blue Gene/Q architecture. In fact, *LoadBalance* has been able to successfully balance a production-sized numerical grid over 256 MPI processes that allow the use of 256*16 = 4096 cores. During the strong and weak scaling analysis *LoadBalance* has been extensively used providing well balanced distribution as the satisfactory scaling performances demonstrate.

The parallelization of the overset pre-processor allowed to reach two targets which are fundamental when running very large simulations. Using the refactored version of the overset code, since it can run on several processors and since each process is now asked to construct topology and cell dependencies only for a subset of the grid, we have circumvented the problem of the huge amount of memory requested during the pre-processing phase. This was a problem that cannot be solved on a serial machine (unless the use of a machine with order of hundred GBs of RAM). Secondly, the required wall clock time requested on this phase may be significantly reduced increasing the number of computing nodes which may be employed by the final MPI version of the overset pre-processor.

The implemented parallel I/O allowed the use of single input and output files, avoiding the generation of input and output files for each processor, which can be unaffordable when a large number of processors is used. Unfortunately, the current implementation does not show any significant improvement in terms of I/O CPU time, some issues must be investigated and will be a matter of near future activities.

We performed a first strong scaling analysis based on an 80 million of grid point case with fixed grids. The analysis has been executed on FERMI (Blue Gene/Q at CINECA). Since each node of FERMI has 16 cores we used 16 OpenMP threads (using 32 or 64 threads to exploit FERMI hardware threads gives no significant performance gain using *χnavis*) while the MPI processes are distributed one per each FERMI node. As a test case, we simulated 10 temporal iterations and used 3 multi-grid levels. The results of the analysis are provided in the figures below reporting the elapsed times (Figure 4) and the efficiencies (Figure 5) against the number of computing nodes.

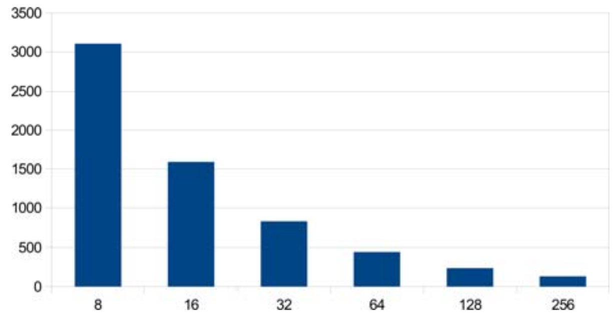


Figure 4: Elapsed times vs. number of compute nodes.

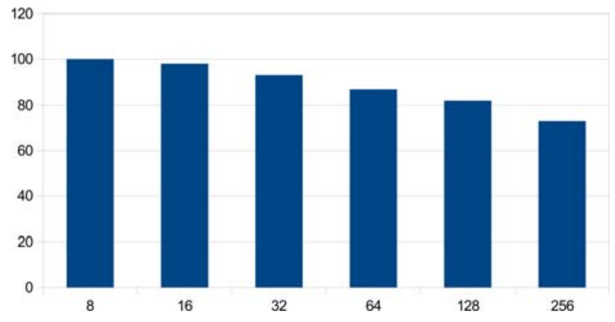


Figure 5: Efficiency vs. number of compute nodes.

The elapsed times are given in seconds. The efficiency is based on the 8 nodes case. The value of efficiency is very good when using 32 nodes (512 cores), good at 128 nodes (2048 cores) and fairly good at 256 nodes (4096 cores). In this test case, the moving grids algorithm has not been activated. We decided to repeat the scaling analysis considering a moving grid case: the largest size having moving grids we simulated was an 11 million grid points. The results are very close to that of the fixed grid case. We also performed a weak scaling analysis to evaluate the performance trend when enlarging the grids. The results of the weak scaling analysis are very good, since the elapsed times are nearly constant as expected in the ideal case.

To have a complete overview of the code/machine performances, we also addressed a scalability analysis using Intel Sandy - Bridge architectures; this test can be of paramount importance for take into consideration different Tier-0 architectures. We tested the code with two different types of Sandy-Bridge nodes: the first type is a 2 eight - core Intel(R) Xeon(R) CPU E5 - 2658 2.10 GHz Dual socket, the second one is a 2 eight - core Intel(R) Xeon(R) CPU E5 - 2687W 3. 10 GHz. The performances of the runs highlighted a performance gain close to 10 for the 3.1 GHz nodes and 6.5 for the 2.1GHz.

Machine	Elapsed Time (seconds)
FERMI Blue Gene/Q	264
Dual Sandy-Bridge E5-2687W 3.1 GHz	26

Dual Sandy-Bridge CPU E5-2687W 2.1 GHz	39

Table 3: Scalability analysis using different architectures. Detailed information is given in the text.

Finally, we proved the scalability of *χnavis* for the current sizes to be sufficient to allow for an efficient usage of a Tier-0 architecture.

The project also published a white paper which can be found online under [3].

2.8.2 *Scalability analysis, OpenMP hybridization and I/O optimization of a code for Direct Numerical Simulation of a real wing, 2010PA1454*

Code general features

Name	Direct Numerical Simulation of a real wing code (no acronym)
Scientific field	Engineering and energy
Short code description	The code used in this work is an in-house, highly validated, compressible Navier-Stokes solver based on a finite volume second-order space discretization with an optional hybrid WENO discretization of the convective terms that is activated in computations involving the presence of shock waves. A third-order low-storage explicit Runge-Kutta scheme is used to advance the equations in time. The code is written in Fortran programming language. The key feature of the code is its capability of preserving the global kinetic energy at discrete level in the limit case of vanishing viscosity and exact time integration, which makes the computation extremely robust without the addition of any form of artificial viscosity or filtering procedure. The initial parallelization relied on a Cartesian decomposition strategy, whereby the computational domain was split in the three coordinate directions. Communication between neighbouring blocks was handled by means of an efficient MPI parallelization.
Programming language	Fortran
Supported compilers	GNU, IBM
Parallel implementation	MPI/OpenMP
Accelerator support	None
Libraries	None
Building procedure	Basic Makefile compilation
Web site	None
Licence	not assigned

Table 4: Code general features for the Direct Numerical Simulation of a real wing code.

Main objectives:

The main objective of the project was to develop a new MPI/OpenMP hybrid version of a MPI parallelized code for direct numerical simulations of a real wing. The code was already ported to the IBM Blue Gene/Q architecture and showed good scalability. The main bottleneck recognized by the PI of the project was related to limited in-node performance of the code while using all available hardware threads (64) per node. The hybridization of the code was expected to overcome those scalability issues.

Accomplished work:
The main goal was fully achieved – the full MPI/OpenMP parallelization of the code was developed. The increase of performance was confirmed with a series of scalability tests with various problem sizes on the FERMI system. The new hybrid MPI/OpenMP version of the code is currently in use for production runs.

Main results:
We have successfully implemented a hybrid MPI/OpenMP version of an existing flow solver for direct numerical simulations of turbulent flows. The final version of the code was extensively tested and is now used for production runs on the FERMI system. The hybrid MPI/OpenMP version of the code outperformed the pure-MPI implementation. The code can now take full advantage of the SMT mechanisms of the IBM Power A2 processor architecture. This was confirmed by a series of performance tests, from which the largest and most representative one was constructed on a grid of size 4096x384x2048. Table 5 below shows the achieved performance.

Number of nodes	MPI 64 MPI / node		MPI/OpenMP (4 MPI x 16 OpenMP) / node	
	Walltime	Speed-up vs. 256 nodes, MPI	Walltime	Speed-up vs. 256 nodes, MPI
256	21.56s	1.00x	10.49s	2.05x
512	10.97s	1.96x	5.42s	3.97x
1024	6.27s	3.43x	3.38s	6.37x

Table 5: Scalability and performance comparison of pure-MPI and MPI/OpenMP versions of the code.

The project also published a white paper which can be found online under [4].

2.8.3 *Next generation pan-European coupled Climate-Ocean Model - Phase 1 (ECOM-I), 2010PA1470*

Code general features

Name	*HBM*
Scientific field	Earth Sciences and Environment
Short code description	*HBM* (HIROMB-BOOS Model) is the ocean circulation model actively developed by the Danish Meteorological Institute (DMI). The application code is proprietary to DMI and not publicly available. The model code is written in free format standard Fortran95 and has been parallelized using OpenMP, MPI and more recently also OpenACC.
Programming language	FORTRAN 90
Supported compilers	xlf90, gfortran

Parallel implementation	MPI, OpenMP
Accelerator support	No support for GPU
Libraries	
Building procedure	autoconf, make
Web site	No public version available
Licence	MoU agreement between DMI and respective PRACE partner

Table 6: Code general features for *HBM*.

Main objectives:

The goals of the ECOM-I project was to reduce the runtime of 10-day forecast from 16 hours (current level) to 2 hours. The scalability should go from current 900 cores to as many cores as possible. The goals will be reached by optimizing parallel computing and I/O, two-nesting as well as halo-communication in MPI.

Accomplished work:

PRACE experts working with the ECOM-I project were contributing to the benchmarking of the model code and scaling performance on the selected PRACE Tier-0 systems, particularly the Blue Gene/Q architecture.

A module for the small benchmark was created with routines for saving and retrieving data for each of the two target routines. The save routines automatically save all needed fields into separate files for each domain and the arguments to the routine are an id and a number to indicate in which time step to save. The benchmark can be used for several tasks, including validating new versions of the routines and tuning of different settings.

The application code has been ported to the Blue Gene/Q system and both threaded and mixed threaded parallel versions scaling have been tested.

Main results:

The model application is implemented and proved to preserve bit-reproducible results with a serial, threaded (OpenMP), parallelized (MPI) and mixed (OpenMP+MPI) builds and across different architectures. After initial difficulties managing bit-reproducibility on the Blue Gene/Q system, eventually all the problems have been fixed and the code successfully ported to the FERMI system and other smaller systems to assert versatility. Both XL and GNU compiled versions of the application preserve bit-reproducibility. Although it has been shown that the results on the Blue Gene/Q platform are not bit-wise consistent with the x86-based systems it has been agreed that the differences are related to hardware floating point implementation and are acceptable after discussion with the model developers.

Figure 6 shows results of the OpenMP thread scaling on the one compute node of the Blue Gene/Q system. The twofold scaling curve reflects the underlying hardware configuration of the system. When the application is running with 1 to 16 threads, each thread is using a physical CPU core exclusively. Further increasing the number of threads employs SMT hardware support with threads sharing the CPU resources. For this reason performance for more than 16 threads is lower but the application still benefits from larger thread count. This is visible on the scaling curve. For up to 16 threads almost linear scaling is achieved and further scaling results in the speedup of a factor 2.5 in the application runtime for 64 threads compared to 16 threads (the XL compiler case).

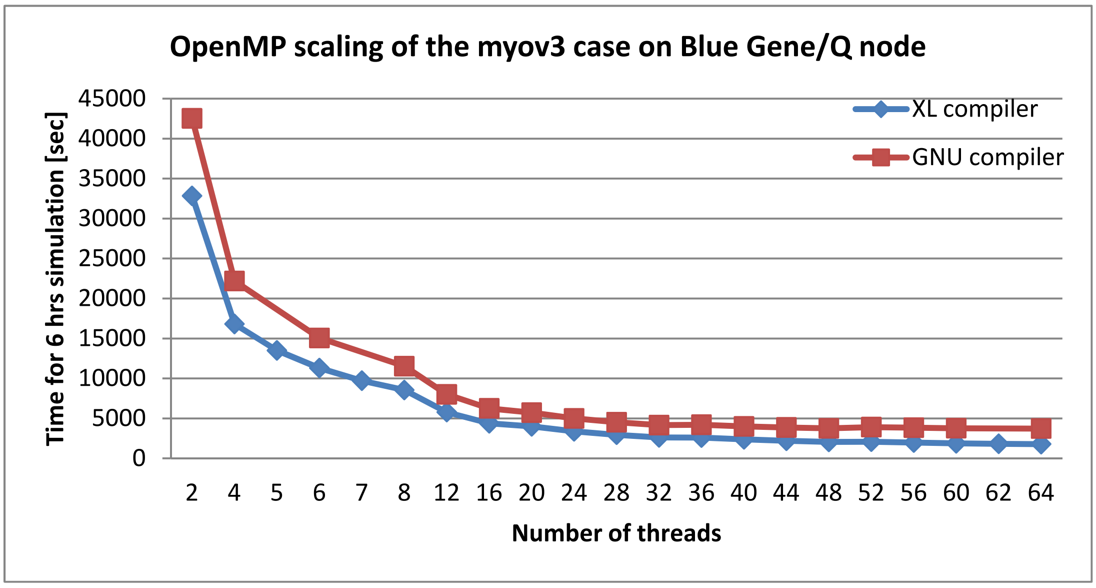


Figure 6: Thread scaling of the *HBM* model 2.8-v2, OpenMP version of the application tested with myov3 case on the one node of Blue Gene/Q. Performance comparison between XL Fortran and GNU Fortran based applications.

The threaded parallel (OpenMP+MPI) version of the code has been compiled with the XL compiler only because the GNU Fortran compiler shows lower performance of the model code with OpenMP enabled on Blue Gene/Q platform.

For the OpenMP+MPI runtime mode it was decided to use one MPI process per Blue Gene/Q node and 16 OpenMP threads on each node. This approach provides optimal resource utilization because of the Blue Gene/Q memory node allocation limitations and demonstrated good performance of the OpenMP threaded version of the model code. While each MPI application process allocates substantial amounts of memory it is not possible to use the maximal thread number per node without the code rearrangement. Initial scaling tests with improved code version for threaded parallel (MPI+OpenMP) mode show also optimal scaling for small node counts.

The project also published a white paper which can be found online under [5].

2.8.4 *Increasing the QUANTUM ESPRESSO capabilities II: towards the TDDFT simulation of metallic nanoparticles, 2010PA0633*

Code general features

Name	*Quantum ESPRESSO*
Scientific field	Material Science, Quantum Chemistry
Short code description	*QUANTUM ESPRESSO* is an integrated suite of computer codes for electronic-structure calculations and materials modelling, based on density-functional theory, plane waves, and pseudopotentials (norm-conserving, ultrasoft, and projector-augmented wave). QUANTUM ESPRESSO stands for opEn Source Package for Research in Electronic Structure, Simulation, and Optimization. It is freely available to researchers around the world under the terms of the GNU General Public License. *QUANTUM ESPRESSO* builds upon newly restructured electronic-structure codes that have been developed and tested by

	some of the original authors of novel electronic-structure algorithms and applied in the last twenty years by some of the leading materials modeling groups worldwide. Innovation and efficiency are still its main focus, with special attention paid to massively parallel architectures, and a great effort being devoted to user friendliness. *QUANTUM ESPRESSO* is evolving towards a distribution of independent and inter-operable codes in the spirit of an open-source project, where researchers active in the field of electronic-structure calculations are encouraged to participate in the project by contributing their own codes or by implementing their own ideas into existing codes.
Programming language	Mainly Fortran 95 with some ancillary subroutines in C
Supported compilers	Any working Fortran 95 and C compiler (compilation is regularly checked against gnu, pgi, intel, IBM xl compiler suites)
Parallel implementation	MPI and OpenMP
Accelerator support	Yes, for Nvidia card, activities are undergoing to support Xeon PHI
Libraries	Blas, lapack and FFT (with interface driver for fftw, mkl, acml, essl, ecc...)
Building procedure	Autotools suite (configure and make)
Web site	www.quantum-espresso.org
Licence	GPL

Table 7: Code general features for Quantum ESPRESSO.

Main objectives:

Implement novel strategies for reducing the memory requirements and improving the weak scalability of *turboTDDFT*, which is a planewave pseudopotential TDDFT code, included in the *QUANTUM ESPRESSO* package. The final goal is to obtain a net improvement of the code capabilities and to be able to study the plasmonic properties of metal nanoparticle (Ag, Au) and their dependence on the size of the system under test. This project is a preliminary, but necessary step for the simulation of a hybrid systems composed of metal nanoparticles and molecular antenna coupled by plasmonic interactions.

Accomplished work:

- *QUANTUM ESPRESSO* (*QE*) is a suite of inter-operable codes with two main kernels: *pwscf*, for total energy ground state simulations; and *cp*, for Car-Parrinello like simulations. All other kernels are for computation of ground state properties, post-process like analysis or correction to the DFT approximations for more accurate results. TurboTDDFT is one of these kernels, mainly meant to compute optical properties. It requires, as a pre-process step, a pwscf ground simulation for the ground state.

- Usually new parallelization paradigms are first introduced and validated in cp and pwscf main kernels and then ported to other kernels. Following this pattern in this work we have implemented in TurboTDDFT a parallelization technique already validated in cp and pwscf, named "task grouping". Task grouping was first implemented in cp kernel code to exploit the extreme parallelism of the Blue Gene architecture, and then ported on pwscf.

- Task group is a key technique to allow the 3D parallel FFT implemented in plane wave DFT codes like *QE* or CPMD, to scale beyond the number of "planes" in the z direction of the domain. Due to the specific characteristic of the reciprocal basis set (determined by the kinetic energy cut-off) the 3D FFT it is parallelized by scattering z columns (all points in the domain having the same values for x and y coordinate) in reciprocal space, and distributing the z "planes" (all point having the same value for z) in real space. Task group, as the name says, groups together many electronic bands (each one requiring a forward and backward 3D FFT) to be processed at the same time by a group of processor rather than a single task, and allowing for a greater scalability. The groups are implemented using communicators. With this implementation the new limit to the scalability is given by the number of z "planes" multiplied by the number of task group.

Main results:

The scalability we were targeting within this project was the weak scalability on a real dataset, allowing to run large systems with many electrons. We chose two test cases: one molecule (Test1) and one Au nanoparticle (Test2). Test1 involved 59 atoms, 109 electronic bands and 98385 basis vectors. Test2 involved 92 atoms, 506 electronic bands and 219223 basis vector. The number of atoms plays a minor role in determining the size and the wall time, what counts are the number of electronic bands and the number of basis vectors. To make things more complex, the algorithm does not scale linearly with the number of basis vectors and number of bands (but superlinear), so we have only a rough estimate of the weak scalability. If we neglect for the purpose of the present benchmark this non-linearity of the algorithm, we observe (see Table 8) that Test2 is 10.2 times larger than Test1.

Test case name	Test case size	Number of core	Wall time	size / size Test1	cpu hours / Test1 cpu hours
Test1	98385 * 109	1024	43 min	1	1
Test2	219223 * 506	2048	296 min	10.2	13.7

Table 8: Summary of test results on molecular (Test1) and Au nanoparticle (Test 2).

As can be seen from the above results, the computational resources required by the large test case Test2 is 13.7 times larger than the computational resources required by Test1, being Test2 10.2 times larger than Test1, and considering that the algorithm is superlinear, we can conclude that the turboTDDFT code with new task group parallelization scale fairly well with real case datasets.

Data set	Test case size	Number of cores	Wall clock time	Speed-up vs the first one	Number of Nodes	Number of process
Test1	10723965	1024	43min	1	64	256
Test2	110926838	2048	296min	1.5	128	512

Table 9: Weak speed-up obtained.

HERMIT specific issues

Several test runs have been performed (using cp kernel and W256, 256 water molecule, dataset) to find out the best combination of compilers, libraries and execution environment. As the compiler we used Intel suite (through the ftn Cray wrapper) with an optimization level O2 (in *QE* performances mainly depends on libraries); as the library for linear algebra we used Cray scilib, for the FFT we used a custom 3D FFT linked with FFTW (version 2.5). But the most important issues we discover about the performance on HERMIT, is related to the task/thread affinity (*QE* best perform and scale with a combination of 4 to 8 thread per tasks). To make the code perform at best we need to explicitly specify "-cc numa_node" on the aprun command line. In table below we report the performance registered for two run with and without the "-cc numa_node" flag:

Launch command	Recorded time
aprun -n 512 -N 8 -d 4 ./cp.x -ntg 4 -nbgrp 2 -input cp.in > h2o.out.test.11 2>&1	4m19.80s CPU 6m 0.14s WALL
aprun -n 512 -N 8 -d 4 -cc numa_node ./cp.x -ntg 4 -nbgrp 2 -input cp.in > h2o.out.test.11 2>&1	7m54.39s CPU 2m13.18s WALL

Table 10: Performance of the cp kernel with W256 dataset. 2048 cores, 512 tasks, 4 threads per task.

The project also published a white paper which can be found online under [6].

2.8.5 *Scalability of gyrofluid components within a multi-scale framework, 2010PA1505*

Code general features

Name	*GEM*
Scientific field	Plasma Physics
Short code description	*GEM* is a 3D MPI-parallelised gyrofluid code used in theoretical plasma physics at the Max Planck Institute for Plasma Physics, IPP, at Garching. Germany. The code *GEM* addresses electromagnetic turbulence in tokamak plasmas. Its main focus is on the edge layer in which several poorly understood phenomena are observed in experiments. The code is written in Fortran/MPI and is based on the electromagnetic gyrofluid model. It can be used with different geometries depending on the targeted use case. The code has been run up to now on conventional tokamak cases like the ASDEX-Upgrade at the IPP in Garching.
Programming language	Fortran
Supported compilers	Intel, IBM xl and other well-known compilers
Parallel implementation	MPI
Accelerator support	No
Libraries	Fftw3, mkl
Building procedure	make
Web site	http://solps-mdsplus.aug.ipp.mpg.de/wsvn/GEM(Z/R)
Licence	IPP

Table 11: Code general features for GEM.

Main objectives:

The main goal of this project was to improve the parallel weak scalability of the application GEM optimising the communication scheme to larger tokamak cases.

Accomplished work:

Since the main bottleneck of the code is the solver, we have particularly focused on its improvement. Various versions of the MPI-parallelised code have been set up for the weak scaling analysis. In most cases the I/O routines were eliminated to solely concentrate on the solver features. The *Scalasca* utility has been used to analyse the runtime behaviour of the code and to filter out the suboptimal routines. The *dummy* version functioning as a baseline for our measurements uses no boundary or sum information. The *CG* version implements a conjugate-gradient method for the solver iteration. Furthermore, two different Multigrid versions have been analysed: the *MGV* version (Multigrid with V-cycle scheme) and the *MGU* version (Multigrid with U-cycle scheme) which has been developed within this project.

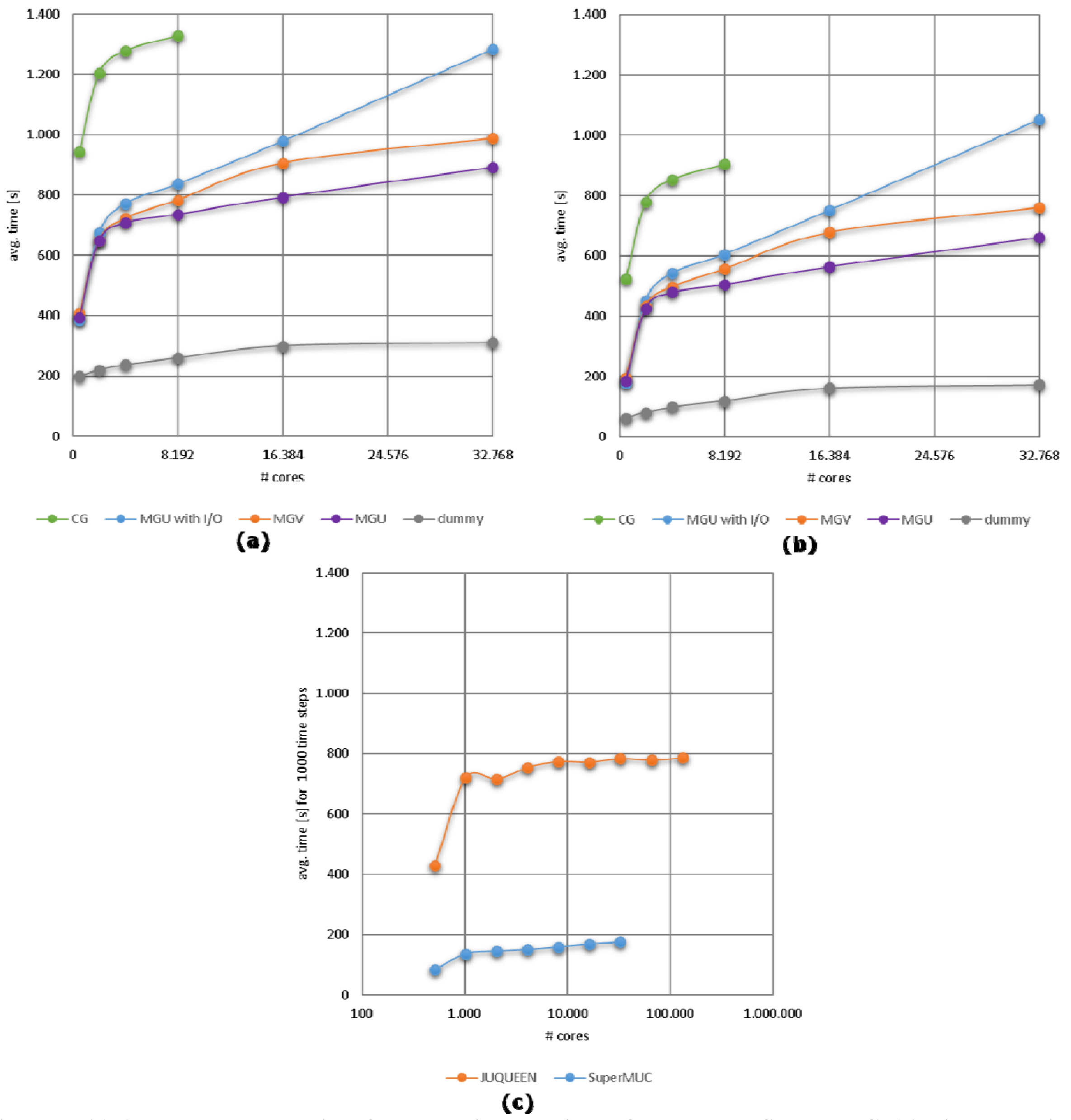


Figure 7: (a) Overall wall-clock time for the various versions of the code on SuperMUC. (b) Time spent in the MPI functions (from the Scalasca analysis). (c) Comparison of the weak scaling of the MGU version of GEM on SuperMUC (LRZ) and on JUQUEEN (JSC).

Main results:

Scaling measurements have been done on the HPC systems SUPERMUC (IBM System x iDataPlex) at LRZ and JUQUEEN (IBM Blue Gene/Q) at Jülich Supercomputing Centre (JSC) to benchmark the performance of *GEM*. A comparison of the performance of the various versions on SUPERMUC is shown in Figure 7 (a). Detailed performance analysis using the Scalasca tool revealed that the scalability of the dominating MPI routines determines the scalability of the entire code, as shown in Figure 7 (b). This is very significant for the CG solver due to its heavy usage of the *MPI_Allreduce* function. Here the *MGU* version shows best scalability. Finally, Figure 7 (c) presents the good weak scaling behaviour of the *MGU* version (excluding the I/O) up to 32.768 cores on SUPERMUC(IBM System x iDataPlex) at LRZ and up to 131.072 cores on JUQUEEN (IBM Blue Gene/Q) at Jülich Supercomputing Centre. As the code has not been optimised for in-order architectures like JUQUEEN and because of differences in clock-speeds, task-binding etc. the overall performance on JUQUEEN is worse than on SUPERMUC. However, scaling is slightly better on JUQUEEN.

The main conclusion is that the goal of running larger systems commensurate with the ITER tokamak edge pedestal layer region with similar wall-clock time with the same number of grid points per core as standard cases was achieved (in principle, since actual runs require production resources). The diagnostic tool Scalasca greatly helped filtering out the suboptimal options. With the help of Scalasca a version of the code has been found for which acceptable weak scaling is confirmed. Currently the physics content of the code is extended to treat real stratification. The relevance of the results obtained is essentially the demonstration of feasibility of planned large-system runs. The improvement of the I/O scheme using systems such as the ADIOS library [http://adiosapi.org/] and possible extension to a hybrid MPI/OpenMP scheme are being explored. This new version of the code is planned to be used in a future large-scale project at LRZ to simulate turbulences in the ITER reactor-plasma experiment currently under construction in Cadarache, France.

The project also published a white paper which can be found online under [7].

2.8.6 *Direct numerical simulation of a high-Reynolds-number homogeneous shear turbulence, 2010PA1492*

Code general features

Name	*SHEAR*
Scientific field	Fluid Dynamics (Turbulence)
Short code description	Turbulence is often induced by shear. One important problem in the study of fluid mechanics is to find the interaction between the mean flow and the kinetic energy of the turbulent fluctuations. The simplest flow in which to analyse this interaction is the so-called homogeneous shear turbulence (HST) which has a constant velocity gradient. The logarithmic layer of general wall-bounded turbulent flows has been investigated for a long time, but the mechanisms of how large-scale motions are generated and collapse into smaller eddies are not well understood. These multi-scale interactions among eddies are of great interest. The objective is to investigate if HST contains the basic energy-production and momentum-transfer characteristics of general turbulent shear flows. To do so, the key point is to achieve a high enough Reynolds number to include a sufficient range of scales. Preliminary tests have been run to determine the necessary test box sizes and Reynolds numbers. To be able to run the required simulations the model code needs to be ported to larger machine systems. Data from the computations will be compared with those of turbulent wall-bound flows at high

	Reynolds numbers, and the results will be made available to the community. The *SHEAR* code is written to do direct numerical simulation of homogeneous shear turbulence of logarithmic layers. The code is a modification of an earlier model code written for turbulent channel flow. In the model, the governing equations of velocities for an incompressible flow are reduced to yield a fourth-order equation for the normal velocity component, and a second-order equation for the normal vorticity component. The third-order explicit Runge-Kutta method is applied and explicit time stepping is adopted. The model uses Fourier-expansions in the streamwise and spanwise directions, and compact finite differences in the shear direction. Domain decompositions are done in two ways: in the xz-plane for performing fast Fourier transformations along the x- and z-axis, and in the xy-plane to perform the derivatives using compact finite differences along the y-axis. In all of the computation, these domains must be transposed to each other by using two collective transposing algorithms. The baseline code is parallelised using MPI and OpenMP. The fast Fourier transformations and computing derivatives are done independently for each MPI process.
Programming language	Fortran90
Supported compilers	gfortran, Intel Fortran, IBM XLF (maybe more; tested with these)
Parallel implementation	Hybrid: OpenMP and MPI
Accelerator support	N/A
Libraries	HDF5, MPI, FFTW and ESSL
Building procedure	Makefile
Web site	http://torroja.dmt.upm.es/
Licence	Private (contact Prof. Javier JIMÉNEZ (jimenez@torroja.dmt.upm.es))

Table 12: Code general features for SHEAR.

Targets and accomplished work:

For this project, the scalability was increased for the Blue Gene/Q. Besides, general improvements to the code were made for parallel I/O, Fourier transformations and communication. These improvements were implemented by Siwei Dong (Fluid Dynamics Group, School of Aeronautics, Universidad Politecnica de Madrid). He was assisted by Vegard Eide (Norwegian University of Science and Technology) and Jeroen Engelberts (SURFsara, Amsterdam, the Netherlands).

Main objectives:

The goal of this project was to improve the performance of the *SHEAR* code on the Blue Gene/Q architecture focusing on the following tasks:

- Implement parallel I/O using the HDF5 library
- Investigate scalability of the communication using MPI_ALLTOALLV when data need to be transposed

- Test scalability of 1-D and 2-D FFT using the FFTW library

Accomplished work:

The hybrid MPI/OpenMP *SHEAR* code is ported to the Blue Gene/Q system architecture. Parallel I/O using HDF5 is implemented for the output of restart data. The parallel I/O performance on JUQUEEN is however not very good. This may be due to I/O-related problems on the system as reported by the Jülich support team. The code shows good intra-node thread scaling running on physical cores but can only obtain approximately a 1.5 speed-up from utilising hyper-threading. Inter-node strong scaling is good up to 1K nodes, while drop-off in efficiency when increasing the number of nodes further can be accounted for by the size of the test case, and also network congestion of the Blue Gene/Q system. Weak scaling is very good.

Main results:

Scaling

Scalability testing is done running a test case of 3072x2048x1536 grid points. Multi-threading performance is tested using 512 nodes, with one MPI task per node, and changing the number of threads from 1 to 64, see Figure 8 (top). The Scalability up to16 threads, i.e. running on physical cores, is nearly linear. Increasing the number of threads further, i.e. utilizing hyper-threading, scalability decreases. Tracing of the code has shown that this is because of the FFTs that obtain only a speedup of x1.3 from x2 or x4 hyper-threading using FFTW.

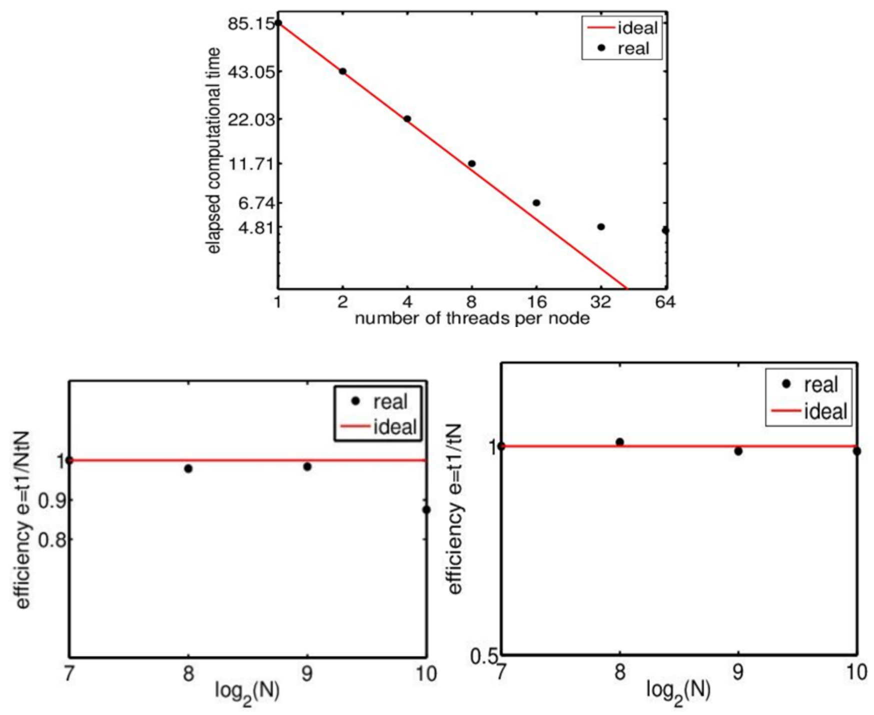


Figure 8: 1 (top), 2 (bottom left) and 3 (bottom right).

Strong scaling is obtained running on 128 to 1K nodes with 1 MPI task and 64 threads per node, see Figure 8 (bottom left). The results show that efficiency drops to 87.5% when running on 1K nodes. This is due to the communication time that does not change going from 512 to 1K nodes, and the amount of computation for the test case may not be large enough to compensate for this.

Weak scaling was tested running on 128, 256, 512 and 1K nodes with a constant work load of 9M grid points per node, showing a nearly perfect scaling, see Figure 8 (bottom right).

Parallel I/O

Parallel I/O has in the scope of this project been implemented using the HDF5 library.

Despite specific Blue Gene/Q tuning settings, initial runs on the JUQUEEN system show that I/O performance is not very good, achieving only 1-2 GB/s when writing data. Additional tuning of the HDF5 I/O was investigated testing different file access settings. Detailed results are reported in [1].

The I/O performance of the *SHEAR* code on JUQUEEN can still not be considered very well. The support team in Jülich has reported that they have been and are still investigating I/O-related problems on JUQUEEN. This is still an open issue.

MPI

The communication when transposing data in SHEAR was done using calls to the MPI_ALLTOALLV function. This function was used since MPI tasks can have different message sizes if the global dataset cannot be evenly distributed among the processes. But test runs on JUQUEEN show that the performance using this function is not good, causing poor scalability of the code. The execution time using MPI_ALLTOALLV has increased dramatically from earlier test runs. A request regarding this has been issued to the Jülich support team. The code was rewritten to use MPI_ALLTOALL instead by padding smaller messages with extra bytes so that each process will send the same amount of data. Detailed results are reported in [1].

FFTW

The Fast Fourier Transforms of the *SHEAR* code can be done by either 1D or 2D FFT using functions from the FFTW library. The 1D FFT can use both the basic and advanced interface implemented in FFTW. Results show that using the basic interface in the 1D FFT is faster than the advanced interface, and also faster than the 2D FFT. The FFT in the SHEAR code is multi-threaded by hand. Testing the OpenMP version of the FFTW library did not improve performance. More details are presented in [1].

Estimation of reworking of the code

The modifications were minor. It is estimated that less than 20% of the code was modified for this enabling and optimization project.

The project also published a white paper which can be found online under [8].

2.8.7 *Massively Parallel Multiple Sequence Alignment Method Based on Artificial Bee Colony, 2010PA1467*

Code general features

Name	*MSA_BG*
Scientific field	Bioinformatics, Life Science
Short code description	Multiple sequence alignment (MSA) is an important method for biological sequences analysis and involves more than two biological sequences, generally of the protein, DNA, or RNA type. The innovative parallel algorithm MSA_BG for multiple alignments of biological sequences was proposed as a result of a previous PRACE study. The MSA_BG algorithm is iterative and based on the concept of Artificial Bee Colony metaheuristics and the concept of

	algorithmic and architectural spaces correlation. The Artificial Bee Colony (ABC) algorithm is an optimization algorithm based on the intelligent foraging behaviour of honey bee swarm.
Programming language	C++
Supported compilers	GNU, IBM XL
Parallel implementation	Hybrid MPI/OpenMP
Accelerator support	No
Libraries	No external dependencies on third party libraries
Building procedure	Makefile (no auto tools)
Web site	-
Licence	-

Table 13: Code general features for MSA_BG.

Main objectives:

The project focuses on performance investigation and improvement of the multiple biological sequence alignment software *MSA_BG* on the Blue Gene/Q supercomputer JUQUEEN. The main objective is refactoring of the parallel implementation by applying hybrid MPI & OpenMP code development on top of the initial MPI implementation.

Accomplished work:

- The application was ported on the JUQUEEN supercomputer and numerous experiments have been conducted. Profiling and benchmark tests were performed in order to evaluate the performance of the application.
- Hybrid MPI/OpenMP parallelization on the top of the MPI only code has been developed.
- The advantages of this approach were showcased through the results of benchmark tests that were performed on JUQUEEN.

The experimental results show that the hybrid parallel implementation provides considerably better performance than the original code.

Main results:

Benchmark tests have been conducted on the JUQUEEN supercomputer in order to measure and tune the performance of the application. The experiments that are presented use various numbers of computing nodes, MPI processes and hybrid MPI/OpenMP configurations. The conditions of termination for these tests are 106 and 107 iterations which refer to attempts for improvement of each sequence alignment quality. The input file that was used contains 149 sequences with a maximum length of 1036 nucleotides.

#NODES	#MPI processes	#OMP threads	Wall time (seconds)	relative speedup
32	2048	-	33.25	1.00
	1024	2	31.69	1.05
	512	4	14.91	2.23
	256	8	15.06	2.21
	128	16	14.23	2.34
64	4096	-	20.30	1.00
	2048	2	26.39	0.77
	1024	4	9.57	2.12
	512	8	8.89	2.28

	256	16	8.75	2.32
128	8192	-	16.13	1.00
	4096	2	24.95	0.65
	2048	4	7.42	2.18
	1024	8	6.55	2.46
	512	16	6.21	2.60
256	16384	-	13.03	1.00
	4096	4	6.06	2.15
	2048	8	5.14	2.53
	1024	16	4.89	2.66

Table 14: Relative speedup of hybrid implementation for problem size of 10^6 iterations.

#NODES	#MPI processes	#OMP threads	Wall time (seconds)	relative speedup
32	2048	-	269.75	1.00
	1024	2	127.22	2.12
	512	4	111.31	2.42
	256	8	110.58	2.44
	128	16	113.39	2.38
64	4096	-	138.44	1.00
	2048	2	74.12	1.87
	1024	4	57.73	2.40
	512	8	57.09	2.43
	256	16	58.29	2.37
128	8192	-	75.23	1.00
	4096	2	48.79	1.54
	2048	4	31.05	2.42
	1024	8	30.41	2.47
	512	16	30.92	1.38
256	16384	-	42.71	1.00
	4096	4	18.18	2.35
	2048	8	17.34	2.46
	1024	16	17.24	2.48

Table 15: Relative speedup of hybrid implementation for problem size of 10^7 iterations.

Table 14 and Table 15 present the speedup achieved when allocating a constant number of nodes with a different set of MPI ranks per node and OpenMP threads. The speedup is normalized to the MPI only version corresponding wall time for the same number of nodes, using 64 MPI tasks per node.

The smallest allocation unit on the JUQUEEN system is 32 compute nodes (512 processor cores) and the maximum number of ranks per node is 64. The number of tasks (ranks per node) needs to be chosen as a power of 2. The set up that was used for benchmark tests using the hybrid MPI/OpenMP implementation has been chosen in accordance to the configurations listed in Table 14.

Configuration Number	**#Ranks per Node**	**#OpenMP Threads**
1	32	2
2	16	4
3	8	8
4	4	16

Table 16: Configuration alternatives for benchmarking the hybrid MPI/OpenMP implementation.

According to the results obtained from the benchmark tests, the setup that uses 16 MPI processes per node with 4 OpenMP threads (configuration number 2) seems to indicate a threshold regarding performance. When increasing the number of threads further for a constant number of nodes, performance may still increase, but not significantly.

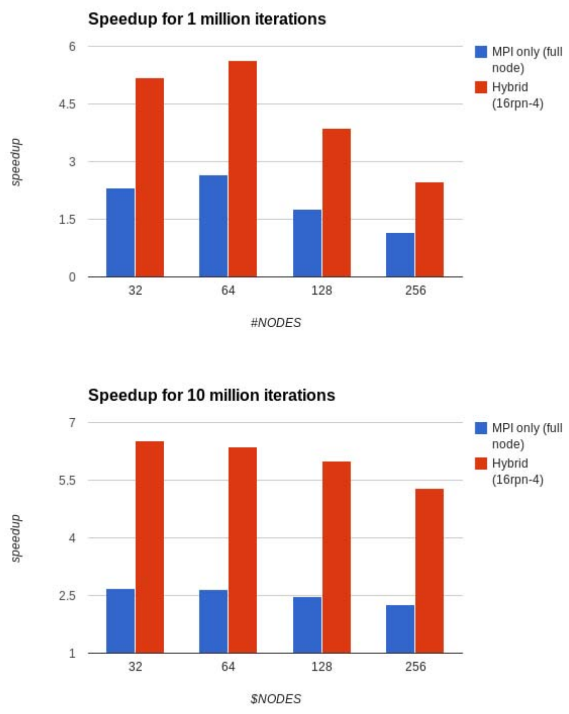


Figure 9: Relative speedups obtained via MPI only and hybrid MPI/OpenMP implementation. 10^6 compared to 10^7 iterations.

Figure 9 presents the gain of using the hybrid MPI/OpenMP implementation in comparison to exploiting the full node using only MPI processes. The red bars represent runs using the hybrid MPI/OpenMP implementation (with configuration number 2), while the blue bars refer to runs using the MPI only implementation, occupying 64 processes per node. Corresponding runs that use the 16 cores available on each node are used as a base in order to normalize the relative speedup. Therefore, in case a configuration is used that occupies 16 MPI tasks per node, there would be two options: either to use 4 times more MPI processes or exploiting the remaining tasks on each node with OpenMP threads.

The project also published a white paper which can be found online under [9].

2.9 Cut-off June 2013

2.9.1 *Optimization of PIERNIK for the multiscale simulations of high-redshift disk galaxies, 2010PA1757*

Code general features

Name	Optimization of *PIERNIK* for the multiscale simulations of high-redshift disk galaxies -
Scientific field	radio astronomy
Short code description	Main aim of this project was to prepare *PIERNIK MHD* code for large scale simulations of multiphysics phenomena in gaseous disks of galaxies, active galactic nuclei, and planet forming circumstellar disks. In our studies we focused on the case of galactic disks, which

	involve many physical ingredients and processes, such as magnetic field generation, cosmic-ray transport and gravitational instability induced star formation. In such cases we need to resolve multiscale environment ranging from parsec scale, gravitationally bound star forming regions, up to tenths of kpc long cosmic-ray driven outflows.
Programming language	C/C++, Fortran, python
Supported compilers	gcc 4.5, Cray compiler openmpi-1.4.2 hdf5-1.8.5 fortran 2003 compiler (>=gfortran-4.7, >=ifort-13.1)
Parallel implementation	Message Passing Interface (MPI)
Accelerator support	Code optimized for Cray compiler
Libraries	git python 2.7 HDF5 (>=1.8.8, --enable-shared --enable-fortran --enable-fortran2003 --enable-parallel) yt for visualization FFTW (>=3.0, optional, for selfgravity) Lapack (optional, for selfgravity) matplotlib (optional, visualization) IDL (optional, visualization)
Building procedure	The following applications are required by PIERNIK: gcc 4.5, openmpi-1.4.2, hdf5-1.8.5 The following tools are required by gcc 4.5: ampfr-3.0.0, mpc-0.8.2, gmp-5.0.1 **gmp-5.0.1 installation:** ./configure --prefix=${HOME}/SVN/local --host=x86_64-redhat-linux –build=x86_64-redhat-linux make make install **mpfr-3.0.0 installation:** ./configure --prefix=${HOME}/SVN/local --host=x86_64-redhat-linux –build=x86_64-redhat-linux --with-gmp=${HOME}/SVN/local make make install **mpc-0.8.2 installation:** ./configure --prefix=${HOME}/SVN/local --host=x86_64-redhat-linux --build=x86_64-redhat-linux --with-gmp=${HOME}/SVN/local --with-mpfr=${HOME}/SVN/local --with-mpc=${HOME}/SVN/local make make install

	gcc 4.5 installation: export PATH=$PATH:${HOME}/SVN/local/bin export LD_LIBRARY_PATH=${HOME}/SVN/local/lib ./configure --prefix=${HOME}/SVN/local --host=x86_64-redhat-linux –build=x86_64-redhat-linux --with-gmp=${HOME}/SVN/local --with-mpfr=${HOME}/SVN/local –with-mpc=${HOME}/SVN/local --disable-altivec --disable-fixed-point --without-ppl –without-cloog --disable-lto --enable-nls --without-included-gettext --with-system-zlib –disable- checking --disable-werror --enable-secureplt --enable-multilib --enable-libmudflap –disable- libssp --enable-libgomp --enable-cld --disable-libgcj --enable-languages=c,c++,fortran --enable-shared --enable-threads=posix --enable-__cxa_atexit --enable-clocale=gnu make -j3 make install **openmpi-1.4.2 installation:** ./configure --prefix=${HOME}/SVN/local --with-openib --with-tm –enable-fortran --enable-io-romio CC=gcc-4.5.0 CXX=g++-4.5.0 F77=gfortran-4.5.0 FC=gfortran-4.5.0 make -j3 make install **hdf5 installation:** ./configure --enable-shared --enable-parallel --enable-hl --enable-fortran CC=/home/users/trojan/reef/SVN/local/bin/mpicc FC=/home/users/trojan/reef/SVN/local/bin/mpif90 --prefix=${HOME}/SVN/local make make install
Web site	http://piernik.astri.umk.pl/doku.php
Licence	GNU General Public License

Table 17: Code general features for PIERNIK.

Main objectives:

The aim of this project was to increase the performance of the *PIERNIK* code in a case where the computational domain is decomposed into large number of smaller grids and each concurrent process is assigned a significant number of those grids. These optimizations enable the *PIERNIK* to efficiently run on Tier-0 machines.

Accomplished work:

The following actions were undertaken to optimize the code:

1. The first step in reducing the MPI overhead relies on the identification group of processes running on the same computational node and converting MPI calls into direct memory access.
2. We implemented coalescing of MPI messages wherever it was applicable, i.e. all messages exchanging in one step between a pair of processes are now put into the common buffer and only one message is sent. This implementation significantly decreased fragmentation of the communication.

3. Additionally, in order to decrease the number of MPI messages we have implemented domain decomposition using the Morton space-filling curve (SFC), which provides high "localization", i.e. neighbouring grids are located on the same processes as much as possible. When carefully implemented, the properties of SFC can allow for fast neighbour searching. This is essential for reducing costs of AMR bookkeeping.
4. Finally, we changed AMR so that it became more selective. It doesn't refine the full block at once, but only the required regions are covered by finer grid blocks. This greatly improves the performance of initial iterations of the grid structure and saves some blocks from unnecessary refinements during the regular refinement update.

Main results:

Optimization results

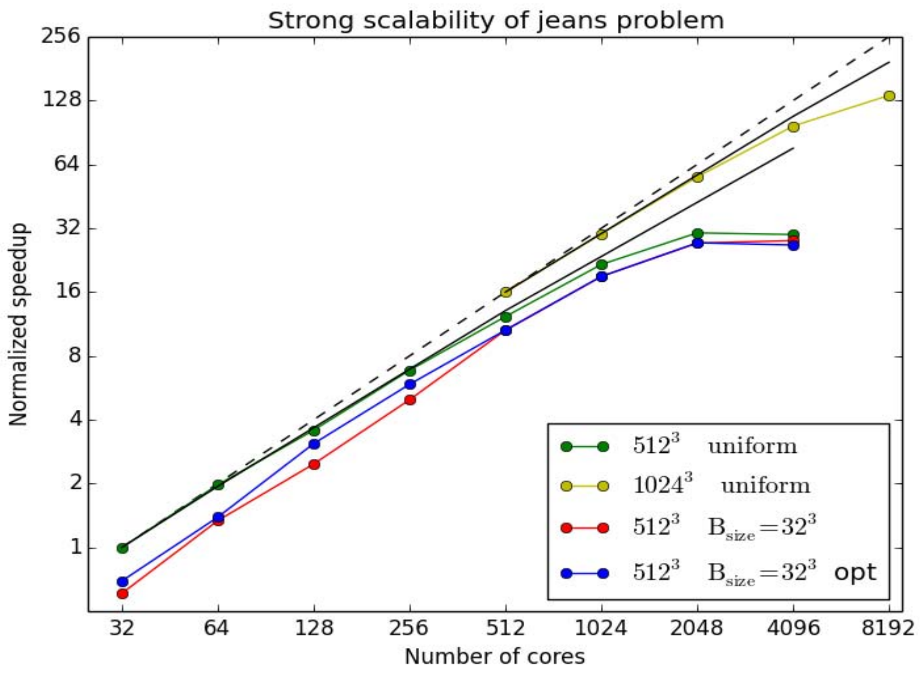


Figure 10: Strong scalability of jeans problem. The blue curve shows the effect of our optimization.

Strong scalability curves (Figure 10) for the uniform grid of moderate sizes (5123 red, green and blue, 10243 yellow) are taken as reference for optimization. Black lines are ideal scaling curves for each of those runs taking into account the ratio of the total number of cells (including guardcell layers) to the number of physically valid cells. Dashed black line shows ideal speed-up. The red curve shows *PIERNIK*'s performance for many grids (of equal cell size without AMR) per computational process before the optimization, the blue curve reflects results of the optimization. The main improvement apparent for the runs on 32 to 256 cores results from the conversion of MPI calls into direct memory access within processes running on the same node. The reduced efficiency of computations on 2048-4096 cores for the 5123 run should be understood as limitation for the total size of the grid processed by a single core. The present computationally inexpensive MHD algorithm the code scales very well on up to 8192 cores, taking into consideration that one single core processes more than 643 cells, even if they are distributed in many blocks.

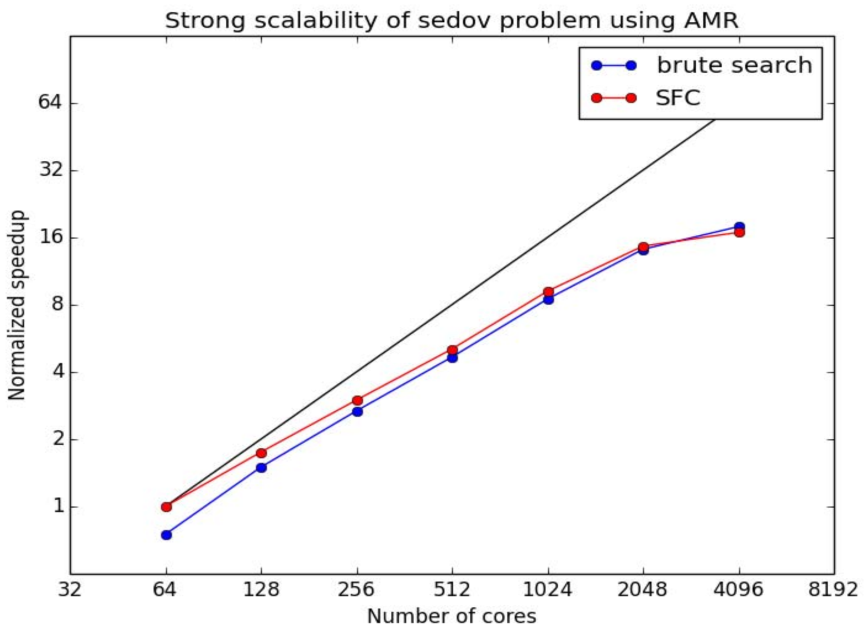


Figure 11: Strong scalability of sedov problem using AMR method.

The strong scaling curve (Figure 11) in the AMR run showing improvement in performance after implementation of domain decomposition using the Morton space-filling curve. Figure 12 displays the wall-time spent on grid operation (total time minus time spent on the hydro algorithm).

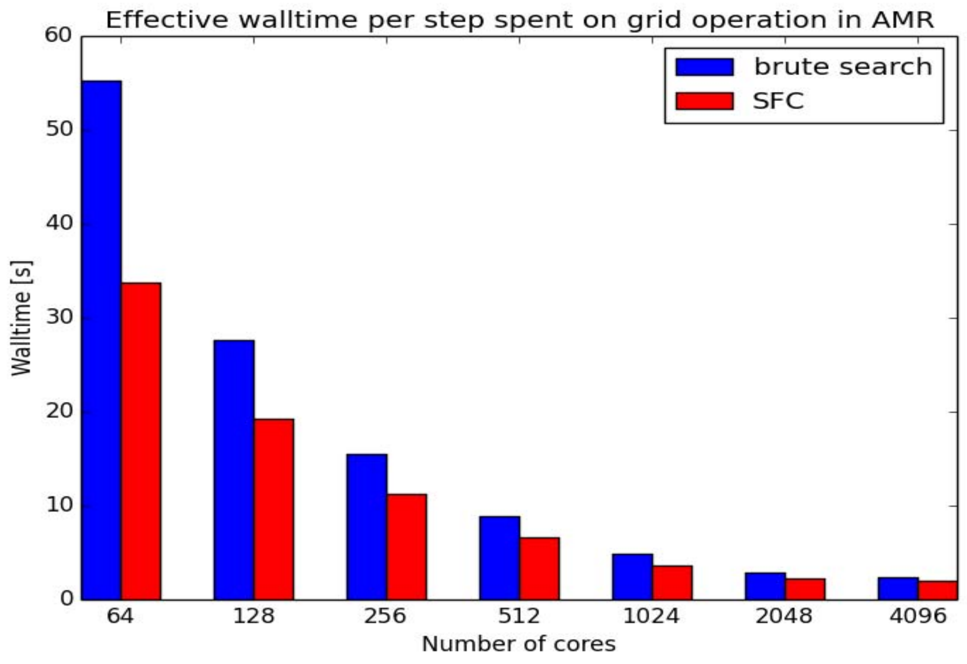


Figure 12: Performance improvement obtained by using SFC for domain decomposition.

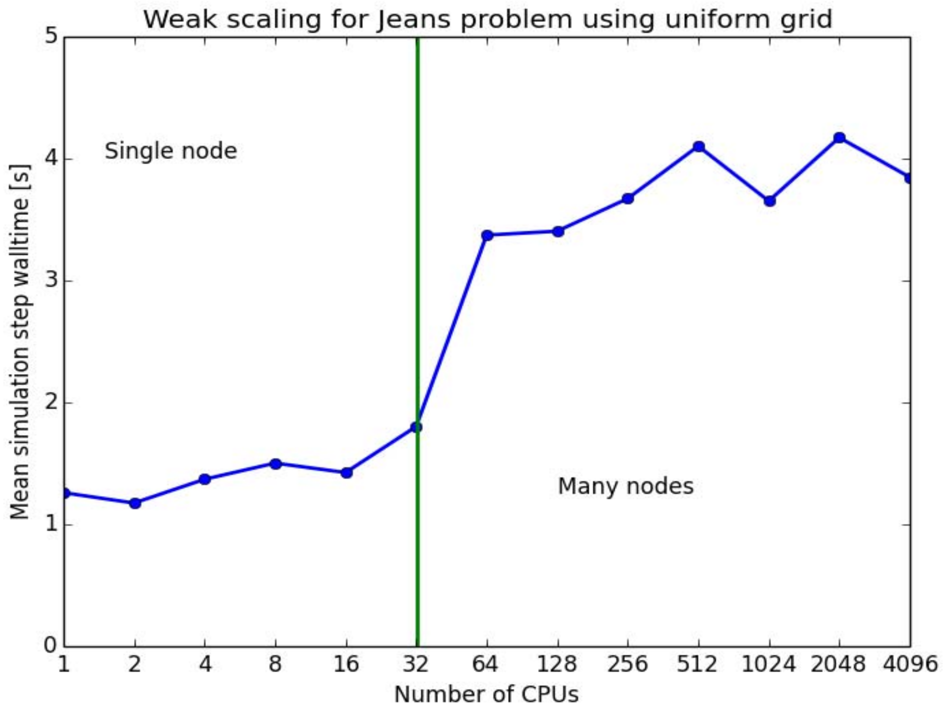


Figure 13: Weak scaling curve for Jeans problem using 64^3 cells per process.

- As was shown in Figure 11, Figure 12 and Figure 13 several improvements were achieved. Overall the performance of non-uniform and adaptive meshes, that are strictly necessary for the fulfilling scientific goals of the project, was significantly improved. Additionally we have ported the code to the latest Cray compilers which will allow to utilize *PIERNIK* on the broader range of HPC sites.
- Performed optimization greatly improved scalability of the code nearly reaching the reference performance in the situation when each computational process is assigned only one big chunk of the computational domain. Moreover, utilization of the Morton Space Filling Curve resulted in significant reduction of the time spent on grid operation, which was dominant in simulations using Adaptive Mesh Refinement.
- The scalability of the *PIERNIK* code is predominantly dependent on the number of grid cells attributed to every MPI process. For big meshes of the overall size 10243 the code scales very well with respect to the ideal scaling curve (including the overhead of boundary conditions) up to 4096 CPU cores and shows further speed-up by 50% at 8192 cores. The essential gain in scalability has been achieved for meshes divided into a large number of small blocks, typical for intense use of the AMR technique in multi-scale astrophysical simulations. The strong scalability improvement resulting from the work performed within the current project varies in the range of 10% - 20% for 32-1024 CPU cores.

The project also published a white paper which can be found online under [10].

2.9.2 *URANIE, 2010PA1527*

Code general features

Name	*URANIE*
Scientific field	Information analysis, machine learning, nuclear power

Short code description	*URANIE* is a sensitivity and uncertainty analysis platform based on the ROOT framework (http://root.cern.ch) . It is developed at CEA, the French Atomic Energy Commission (http://www.cea.fr).
Programming language	C/C++
Supported compilers	Gcc
Parallel implementation	MPI (OpenMPI)
Accelerator support	No
Libraries	ROOT, nlopt, pcl, swig, parmentis, cppunit, libxslt, opt
Building procedure	make {-with params}
Web site	http://sourceforge.net/projects/uranie
Licence	GNU Library or Lesser General Public License version 3.0 (LGPLv3)

Table 18: Code general features for URANIE.

Main objectives:

Based on the *URANIE*'s inner architecture where a single simulation (job) is indivisible the project's main objective was to investigate and possibly solve the threads bottleneck problem in one of the *URANIE*'s launching strategies (system-based strategy).

In *URANIE*, the number of possible simulations is linearly correlated with the number of processor cores used. In this way increasing the range of input variables causes the number of hardware resources required also needs to be increased. The most used ratio is either 1 or 2 (a single simulation uses 2 cores).

In the system-based strategy based on the fork mechanism provided by the Linux kernel:

- A single job is allocated.
- The master node runs a control process.
- The control process launches children of the control process, each child running a *mpirun* script which runs one computation in the DOE (Design of Experiments), the control process checks for the state of the children in order to decide when to run new children.

This gives good flexibility and good performance on hundreds of processors, but the bottleneck on the master node can become a problem on large runs. Also, this strategy gives little control on the placement of processes in the context of coupled simulations.

Accomplished work:

The main work was carried out on CURIE and focused on providing mechanism to lower the single simulation execution time.

In the meantime the source code tracking revealed some serious errors that changed the logic in one of the most important parts of the application.

With system-based strategy we conducted many test runs in which the odd situation was observed. After increasing the number of jobs above the experimental limit (approximately 450) many launched child processes started to become zombies, so that master process which normally waited for their children to complete, could wait forever because it would never receive the proper terminate signal from them.

The above was strongly related to the problem tracked in the source code. Originally the software authors did not check the return value and error code given by fork() function which was called by master process in the loop limited by number of DOEs. In such cases, when -1 was returned (fork() did not succeeded) the processes with such IDs were still added to the list of running processes, whereas they should have not. This was, in turn, causing other troubles when calling waitpid() function on running threads invalid PIDs (-1) which were completely changing the logic of this part of the code.

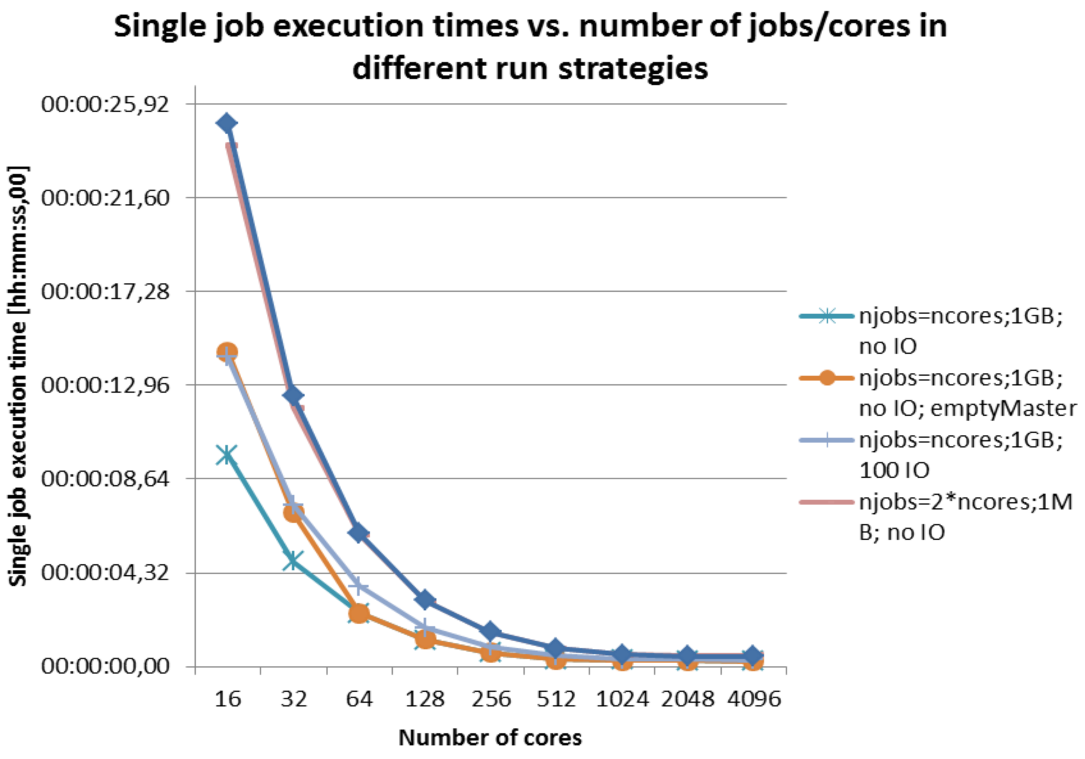


Figure 14: Single job execution times in different run strategies.

It turned out spawning children processes above the limit caused fork() to return errno equal to EAGAIN which literally meant it was not possible to create a new process because the caller's RLIMIT_NPROC resource limit was encountered. In this case it was decided to programmatically call setrlimit function increasing maximum number of threads (setrlimit(RLIMIT_NPROC, …)) in one hand and decreasing the maximum size of the process stack setrlimit(RLIMIT_STACK,…) on the other hand. This helped sometimes a bit to increase the initial experimental limit but not much and not in each test-run case.

Main results:

After analysis of the source code and architecture we found out there is strong linear correlation between number of jobs (simulations) and number of cores used. Then, we focused on optimizing a single job execution time which finally succeeded. Some major logical errors have also been spotted and fixed. In the result we can observe the single job execution time is minimized when number of cores used is maximized at the same time.

The timing characteristics are presented on Figure 14. It is difficult to calculate the speedup for given configurations because, as it was written above, the number of simulations is strictly related to the number of processor cores (and also threads) used. For a single simulation there would be only a single processor used. However, we believe this will not make any sense. Moreover, the speedup and efficiency that can be observed in this specific case are not consequent to the strict definitions and formulas known in HPC. Due to *URANIE*’s internal architecture it was decided to normalize the above times to receive a single job execution (for different setups) time. Afterwards it can be noticed that increasing number of simulations (jobs) along with number of cores make the single job execution time shorter.

The project also published a white paper which can be found online under [11].

2.10 Cut-off September 2013

2.10.1 *Parsek2D-MLMD, 2010PA1802*

Code general features

Name	*Parsek2D-MLMD*
Scientific field	Astrophysics
Short code description	*Parsek2D-MLMD* allows the simulation of astrophysical and space plasmas. It is based on the Implicit Moment Particle In Cell (PIC) code *Parsek2D* and implements a novel adaptive technique, the Multi Level Multi Domain (MLMD) method, for electromagnetic plasma simulations.
Programming language	C / C++
Supported compilers	GNU, Intel
Parallel implementation	MPI
Accelerator support	(none)
Libraries	HDF5
Building procedure	make
Web site	https://github.com/KulMari/Parsek2D_MLMD
Licence	

Table 19: Code general features for Parsek2D-MLMD.

Main objectives:

The field of the application is space plasma physics. Kinetic simulations of plasmas are extremely challenging because processes develop on multiple scales, from the ion (large, slow) to the electron (small, fast) scales. To our knowledge, *Parsek2D-MLMD* is the only semi-implicit adaptive code for plasma simulations. The main scientific case of interest is magnetic reconnection in space. A higher resolution grid is required to resolve the electron-scale physics developing in a small region centred around the X point. We routinely perform realistic mass ratio simulations of reconnection, while usually low mass ratios are used to artificially reduce the problem size. According to tests, our MLMD simulations are 70 time faster than simulations done using the higher resolution on the entire domain.

The technical goal of the project was to restructure the code in order to be able to perform in an acceptable amount of time a simulation of the desired size (about 100x100 ion skin depths) and with high Refinement Factors (the jump in resolution between the grids) of the order of 12-14 (to have appropriate resolution on the refined grid). We have achieved a very satisfactory speed up of about a factor 10 between the code version used at the beginning of the project and the current one.

Accomplished work:

During the project, two main lines of work emerged: one more related to the specifics of the MLMD algorithm and another more technical and more clearly related to HPC optimization activities.

The two main algorithmic developments have been understanding the role of the smoothing in the MLMD system and implementing time sub cycling. This last activity required a big effort for implementation and testing; both activities had deep impact for speed up.

Smoothing is a very powerful tool in Particle-In-Cell simulations, where it is usually used to suppress numerical instabilities and in particular the Finite Grid Instability, an aliasing instability which arises when physical quantities are sampled on low-resolution grids [34]. It became recently clear that smoothing is fundamental in the MLMD system to control numerical noise on the refined grid. Smoothing has to be applied in a specific sequence with respect to MLMD operations to avoid the formation of artefacts on the coarse grid. Starting using smoothing had the positive effect of allowing to increase the stopping criterion in the GMRES iterative field solver [35] (from 10^{-6} to 10^{-3}) and to decrease the number of inner iterations used in the particle mover (from 7 to 3) [36]. A very demanding stopping criterion for the field solver and a very high number of inner iterations for the mover were used to try to reduce the noise on the refined grid and to limit the development of artefacts on the coarse grid. Using smoothing as an alternative to allowed to reduce both the communication requirements (the GMRES solver performs most of the communication, both Point to Point and global, in single-grid PIC codes) and, most importantly, the execution times (the particle mover is the most computationally expensive block in single grid PIC codes) of the code.

Sub-cycling reduces the execution time of simulations and improves scalability with respect to cases when it is not applied. Sub-cycling means using a different time step for the coarse and the refined grid. In the original version of the code, the time step common to all the levels of the MLMD simulation was chosen to resolve the faster physics captured by the refined grid, at the cost of over-resolving the coarse grid. Using a higher time step on the coarse grid does not degrade the physical outcome of the simulation. Actually, it has the positive effect of allowing both of the grids to work in a "healthier" regime. Implicit Moment Method Particle In Cell simulations are supposed to respect the following constraint: $\varepsilon < v_{th,e}\,\Delta t/\Delta x < 1$, where $\varepsilon \approx 0.1$ (but it can be lower if smoothing is applied), $v_{th,e}$ is the average thermal velocity for electrons, Δx is the spatial resolution and Δt is the time step. A grid over-resolved in time may develop the Finite Grid Instability. Conversely, a grid under-resolved in time may incur into accuracy problems [37]. Using high Refinement Factors between the grids expose the coarse grid to the former risk, the refined grid to the latter. For this reason, the time step has to be different on two levels simulated. In the new version of the code, the Time Ratio TR between the time steps in two grids can be chosen as an input parameter. TRs up to six have been successfully used in magnetic reconnection simulations. The refined grid executes TR cycles for each cycle ("iteration") that the coarse grid executes. In this way the number of computational operations on the coarse grid decreases. The reason for the better performance with sub-cycling lies in the reduction of the necessary MLMD operations: C2R (Coarse to Refined) and R2C (Refined to Coarse) communication is executed only one time every TR cycles, rather than every cycle. This reduces communication overhead and waiting times on the refined grid. In particular, PRA particle splitting and communication operations on the refined grid (very time consuming operations, as shown by HPCToolkit profiles) are executed only one time each TR cycles, with a positive effect on scalability and execution time. The physical significance of the simulation is preserved, execution time is saved and scalability is improved; on the negative side, however, it has to be remarked that the coarse grid cores wait idle for quite a long amount of time while the refined grid executes higher frequency cycles. Sub-cycling has thus the effect of improving code scalability and performances at the expense of a higher load imbalance. Load imbalance can however be reduced with massive code restructuring operations which has not been undertaken (for lack of CURIE time and also because the project outcome was already very satisfactory) under the current project.

The main technical optimization improvements done in this project have been the elimination of unnecessary synchronization points within the levels and between the levels, the resizing of vectors used for MLMD operations, the improvement of particle communication, the elimination of indirection (mostly in the mover), the introduction of a new structure for field and particle moment arrays and of pre-screening before the execution of expensive function. Notice that most of the optimization activities performed affect intra-grid rather than inter-grid operations; for this reason, also the execution times of simulations with one single level have been notably speeded up.

Main results

The main result obtained within the project is for sure the speed up of the code, which is almost of a factor ten with respect with the original execution time. To show the code speed up, the same approach adopted in Beck et al. [38] is used to evaluate code performances: the execution time of a two level MLMD simulation with a certain RF between the grids is compared with the execution time of a single-level simulation having the same domain size and entirely solved with the MLMD refined grid resolution. The same test is repeated for the code before and after optimization. Since the aim of the MLMD algorithm is to reduce resource consumption while delivering the same level of relevant physical information, this is our most relevant performance metric. The same test cases as in Beck et al. are used here. The MLMD simulations are quite small in size to make the comparison with the fully resolved simulations achievable with moderate resources. Two PRA cells per side are used in the MLMD simulation with the new code version, while one was used in Beck et al. for the old code version. Using 2 PRA cells rather than one makes the tests with the new code version more challenging.

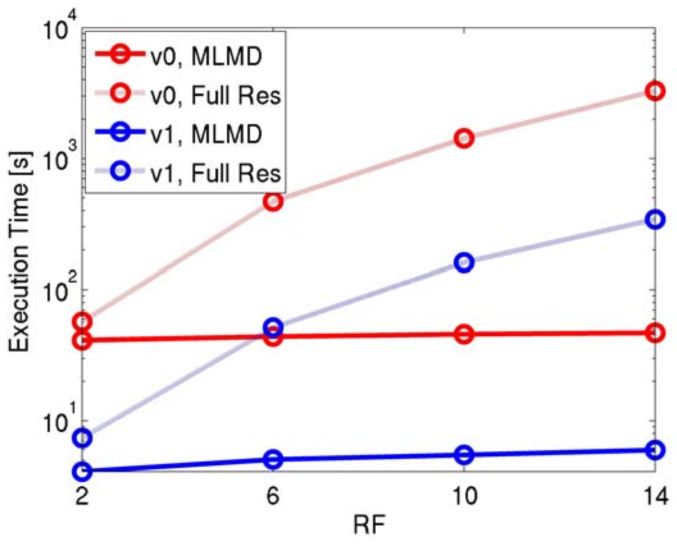


Figure 15: Execution time in seconds for two-level MLMD simulations (straight line) with the old (v0) and new (v1) version of the code. The execution times of one-level simulations with resolution equal to the refined grid resolution are plotted in dashed lines. The y-axis is in log scale: notice the code speed up between v0 and v1.

Figure 15 shows the execution times in seconds on a logarithmic scale as a function of the RF for the full resolution (dash-dotted line) and the MLMD case (straight line). Red is the old code version, blue is the new one. Notice that a significant amount of time is saved if a MLMD rather than a full resolution simulation is done. Notice also the significant speed-up between v0 and v1, achieved during this project, which amounts to almost one order of magnitude.

The second result obtained is that the memory consumption of the application, one of the main concerns in the proposal, has been reduced. Now, we are not forced anymore to use a very high number of cores for our simulations due to high memory requirements.

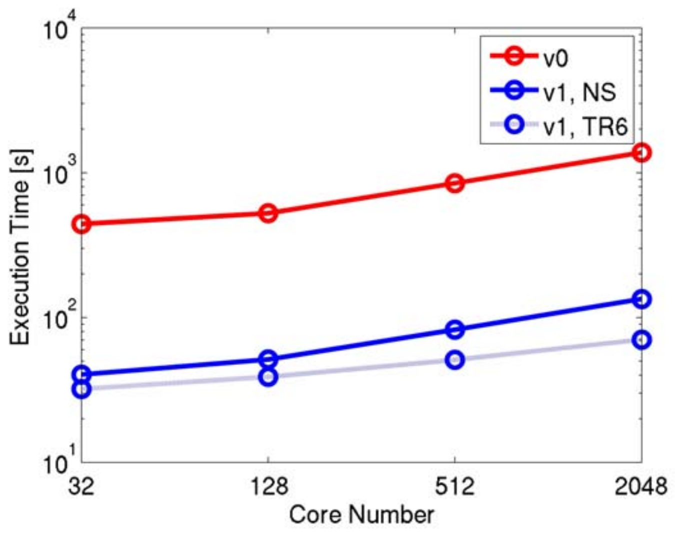


Figure 16: Weak scaling tests for the old code version (v0), the new code version without sub-cycling (v1, NS) and the new code version with sub-cycling and Time Ratio between the grid TR=6 (v1, TR6). A logarithmic scale is used in the y-axis. Notice the code speed up of a factor 10 between the old and the new version of the code. Sub-cycling improves scalability.

Third, sub-cycling has been implemented. This has been the biggest algorithmic improvement. Sub-cycling allows to reduce again the execution time of simulations without losing physical information and also improves code scalability, which however was not the main aim of the current project (see Figure 16).

The project is currently preparing a white paper which will be published online at [1].

3 T7.1.B Application Support for DECI Projects

Distributed European Computing Initiative (DECI), the follow-on activity to the previous DEISA Extreme Computing Initiative, has been integrated into PRACE since PRACE-2IP and continued in PRACE-3IP. The purpose of DECI is to provide the European researchers with the opportunities to access the significant resources on Tier-1 systems across different European countries for their world leading scientific research and simulations. Whilst the DECI process management was covered in WP2, T7.1.B in WP7 was responsible for providing the technical support for DECI projects and focused on the application enabling support for the DECI projects which stated such requirements in their proposals. This section provides an overview of the work and the outcome in the 2nd year of PRACE-3IP. It also includes the application enabling for the DECI projects during this period.

3.1 T7.1.B Overview in the 2nd Year of PRACE-3IP

T7.1.B provided the technical support for the DECI projects of the following DECI calls in the 2nd year of PRACE-3IP.

DECI Calls	Open Date	Closing Date	Allocation Starting Date	Received Proposals (TEs)	Accepted Projects	Enabling Projects
DECI-9 (Started in PRACE-2IP)	17 Apr 2012	30 May 2012	1 Nov 2012	45	31	1
DECI-10 (Started in PRACE-2IP)	5 Nov 2012	14 Dec 2012	1 May 2013	85	37	2
DECI-11	8 May 2013	10 Jun 2013	1 Nov 2013	119	52	1
DECI-12	18 Dec 2013	20 Jan 2014	1 May 2014	61	34	No enabling project

Table 20: DECI calls in PRACE-3IP.

3.1.1 *Technical Evaluations*

In the 2nd year of PRACE-3IP, T7.1.B continued providing the Technical Evaluations for the DECI proposals, including 119 TEs for DECI-11 and 61 TEs for DECI-12.

Technical Evaluations (TEs) aimed to provide technical inputs during the review process for the DECI proposals, including the evaluation of whether the applications were suitable to run on the Tier-1 systems, as well as the identification of the technical requirements from the proposed DECI projects. The DECI TEs started as soon as possible after the DECI calls were closed and were usually completed within around 2-3 weeks depending on the amount of the received DECI proposals. The TE for each proposal was to be completed by their DECI home site, which is usually the DECI site from their own country. If the proposal was from a country which has not been involved in PRACE/DECI, one DECI partner site would be assigned to be the home site for this proposal in such case and responsible for its TE.

In the 2nd year of PRACE-3IP, all the TEs were completed using the online PPR (*Project Proposal and Report)* tool for DECI. When the TEs were completed, all the DECI applications together with the TE reports were then available for the next step, the Scientific Evaluations (SEs). At the same time, the DECI home sites were also responsible to input and update the proposal information in the DECI Process Management DataBase (DPMDB) where the status of all the DECI projects and proposals are kept. The system assignments for the accepted DECI projects were based on the TE results and comments.

3.1.2 *Technical Support for the Accepted DECI Projects*

T7.1.B continued providing the technical support for the accepted DECI projects in the 2nd year of PRACE-3IP. This included the continued support for DECI-9 and DECI-10 projects which have been started in PRACE-2IP and also the full support for all the DECI-11 projects. Although DECI-12 projects have started within PRACE-3IP and the TEs and the starting assistances have been completed for the DECI-12 projects, there will be no enabling support for DECI-12 projects after the PRACE-3IP ended.

Similar to the previous DECI support in PRACE-2IP, T7.1.B in PRACE-3IP helped the DECI users to get access to their assigned Tier-1 systems at the starting stage of their DECI project. The home sites arranged initial meetings with their PIs, to explain all necessary information to the DECI PIs and users. It was also the stage at which the DECI sites should capture the user requirements of the environments, software installed, etc. on the assigned system and provide technical assistance where needed. The initial meetings were in the forms of face-to-face meetings, telcons, videoconferences, or email guidance only, as requested by the DECI PIs and users. A full instruction on the initial meetings and assistance guidance for helping the DECI users to get start with their DECI projects has been listed on the PRACE WP7 wiki page.

T7.1.B provided the technical supports for all the ongoing DECI projects where there were any issues raised via the PRACE support system for DECI users (TTS), or if there were any queries received directly by the DECI sites. Besides, T7.1.B focused on providing the enabling support to the DECI projects which required such assistance in their proposals. The enabling support, which was usually involved 1-6 PMs from the DECI sites, included all aspects of applications enabling such as porting code to the Tier-1 systems, performance benchmarking and profiling, optimisation, and further code development, etc. In the 2nd year of PRACE-3IP. The enabling support for the DECI projects is shown in the Table below. Detailed reporting on the applications enabling can be found in the following sections 3.2, 3.2 and 3.4.

	Project	**Home site**	**Exec Site**	**Enabling Site**	**Architectures**	**Enabling Efforts**
DECI-9	Planck-LFI2	CSC	CSC	CSC	Louhi XT@CSC, Sisu@CSC	1-3 PMs
DECI-10	DNSTF	PDC	EPCC	PDC, SURFsara	HECToR XE6@EPCC, ARCHER@EPCC	1-3 PMs
DECI-10	HYDRAD	RZG	RZG, VSB-TUO	RZG	Hydra@RZG, Anselm@VSB-TUO	1-3 PMs
DECI-11	Planck-LFI3	CSC	CSC	CSC	Sisu@CSC	1-3 PMs

Table 21: DECI enabling projects in the 2nd year of PRACE-3IP.

Before the end of each DECI project, the home site will send the closing instructions to the PIs. A final report was expected for each DECI project from the DECI PI, to summarise the scientific, technical, and algorithmic results of the DECI project, and document the relevance of the PRACE infrastructure. All final reports were expected from the DECI PIs within 3 months after the original deadline of the projects. Any data collection should be completed within the 3-month period as well.

Besides the final reports from the DECI PIs, the technical support for DECI by was also reported. Brief summaries of the technical support for DECI-9 / DECI-10 / DECI-11 are included in this deliverable. White papers were usually contributed by the DECI sites which implemented the applications enabling for DECI projects, to share more technical details and experience of the work done. The white papers are available on the PRACE website [12]. In the 2nd year of PRACE-3IP, one white paper for the DECI-10 enabling project, DNSTF, is under preparation during this deliverable writing. The white paper will be available on the PRACE website in June 2014.

3.1.3 *Coordination and Collaboration*

Fifteen partner sites are involved providing 73 PMs in total for the DECI technical support[1]. In order to track the DECI support progress and discuss plans or issues met. T7.1.B held monthly telcons on every 3rd Friday of each month. The monthly telcons were merged with the WP2-T7.1.B videoconferences since March 2014, in order to avoid duplication of effort.

T7.1.B worked in close collaboration with WP2. Whilst WP2 was responsible for the DECI process management, T7.1.B focused on providing the technical support. As mentioned in section 3.1.1, the DECI proposals together with the TEs would be passed to WP2 for SEs. The final system assignment was done by WP2. It was the responsibility of WP2 to track the DECI projects' progress, but in the case that any issues were met by DECI projects, WP2 would raise the issues with T7.1.B for providing the technical support for the technical questions.

There were several tools used for DECI support and DECI process management:

- BSCW-DECI folders: to keep documents for DECI available for PRACE-3IP partners. There are certain folders for the final PIs' reports storage.
- DPMDB: the database used for the DECI projects information storage and progress tracking. T7.1.B was responsible to input the DECI projects information after TEs.
- WP7 PRACE Wiki: the task internal wiki page to provide useful information on the DECI support, such as workflow, guidance on the initial meetings with PIs, etc.
- TTS: the query tickets system which DECI users are encouraged to submit queries to. This aims to help users to get responses more efficiently and effectively.

In the 2nd year of PRACE-3IP, T7.1.B had two sessions at the WP7 Face-to-Face meetings in Dublin (Dec, 2013) and Daresbury (May, 2014), to report on the progress and discuss the next step work plans, etc.

3.2 Technical Support for DECI-9 in the 2nd Year of PRACE-3IP

All the 31 DECI-9 projects started from 1 November 2012 in PRACE-2IP. T7.1.B continued the support for all DECI-9 projects since August 2013 after the end of PRACE-2IP. There was one DECI-9 enabling project, Planck-LFI2, as reported in the following subsection.

[1] There are more DECI sites in total, but part of them only provides the Tier-1 resources or only involved in the DECI process management task in WP2.

3.2.1 *Planck-LFI2*

The DECI-9 project, Planck-LFI2 (Planck LFI data analysis for second release), has CSC as the home site / exec site / enabling site. The project was assigned to Louhi XT@CSC and Sisu@CSC. This project required 1-3 PMs for the enabling efforts. The enabling work done by CSC was for the two main codes, CosmoMC and LevelS.

Applications Reporting for Planck-LFI2 - CosmoMC

Code general features

Name	*CosmoMC*
Scientific field	Astro Sciences (cosmology)
Short code description	*CosmoMC* is a parallel sampling code for exploring cosmological parameter space. It uses the Markov Chain Monte Carlo method to find which theoretical model fits best to the given cosmological data.
Programming language	Fortran 2003/2008
Supported compilers	Intel
Parallel implementation	MPI, OpenMP
Accelerator support	None
Libraries	GSL optional
Building procedure	Edit makefile, then run “make”.
Web site	http://cosmologist.info/cosmomc/
Licence	Free of charge until January 2015 on conditions that: (1) Any publication using results of the code are submitted to arXiv at the same time as, or before, submitting to a journal. arXiv must be updated with a version equivalent to that accepted by the journal on journal acceptance. (2) If the user identifies any bugs they must be reported as soon as confirmed.

Table 22: Code general features for CosmoMC.

Targets and accomplished work

There were two issues that needed our attention: building *CosmoMC* to use the Gnu Scientific Library (GSL) and to enable OpenMP threads.

Main objectives:

The initial plan was to only build *CosmoMC* to use the Gnu Scientific Library (GSL). Few months after that was done, the version of the CosmoMC changed, and the new version had to be built for the execution machine. This didn’t succeed until the execution machine was changed. On the new machine there was a new problem: the parallel launcher, *aprun*, cannot place threads well into cores if a program is built with the Intel compiler (as it is in this case). Different ways to bind threads with CPU cores were tried to relieve the problem.

Accomplished work:

First, in October 2012, *CosmoMC* was configured to use the GSL library. It took some time to realise that none of the GSL versions available on execution platform (Cray XT Louhi) were compatible with the compiler that was in use (Intel). The solution was to install a custom version of GSL in the user's application directory.

Second, in December 2012, it turned out that the newest version of *CosmoMC* required a newer version of the Intel compiler than was available. The requirement set in because the new version of the software was written in Fortran 2008. Using a different compiler collection wasn't an option because they don't support the application's requirements any better. This was a difficult problem because no further compiler updates were to be installed on the old system. The solution was to move *CosmoMC* simulations to the successor of Louhi (Cray XC30 Sisu) once it was available. The new system had the newest possible compiler collections from all vendors, so the problem was easily solved that way. CSC gave assistance with this in January 2013. In March 2013, a problem turned up: *CosmoMC* could not be run in hybrid MPI–OpenMP mode. The parallel launcher, aprun, cannot place threads well into cores if a program is built with the Intel compiler (as it is in this case). Different ways to bind threads with CPU cores were tried, but the performance is still far from satisfactory. The problem is acknowledged by Cray. In April 2013, a workaround was developed, which solved the problem in case of one MPI task per socket (two per node). This meant that *CosmoMC* can be run in hybrid MPI–OpenMP mode, which speeds up the investigators' work. More MPI tasks wouldn't have made the same effect: they only add more statistics, but do not make computations any faster. Until the end of the project, the *CosmoMC* computations were run using eight OpenMP threads per one MPI task. This was not optimal according to earlier experience: four threads would have performed better, but could not be used.

Main results:

Performance data was not recorded. The only objective was to build the software, with the features requested by the research team.

Application Reporting for Planck-LFI2 - LevelS

Code general features

Name	*LevelS*
Scientific field	Astro Sciences (cosmology)
Short code description	LevelS is a code for simulating cosmological data that will be collected by the detectors in the Planck HFI and LFI instruments.
Programming language	C++, Fortran 95
Supported compilers	All
Parallel implementation	mpibatch script
Accelerator support	None
Libraries	cfitsio, FFTW
Building procedure	Edit makefile, then run "make".
Web site	None
Licence	Proprietary to Planck collaboration

Table 23: Code general features for LevelS.

Targets and accomplished work

In May 2013 one of the applications (*LevelS*) started to fail although it worked fine in February of the same year.

Main objectives:
The objective was to make *LevelS* work again.

Accomplished work:
Substantial of effort was used to spot the error, first by CSC and later by Cray's applications support. The cause of the problem was finally simple: the application was not started with the parallel launcher aprun, but with the "system" command on the compute node, through the *mpibatch* script. In such a case, there are three ways to compile the application:

- load the craype-target-native module before compiling the code;
- add the -target=native option to the cc, CC, or ftn command; or
- call the underlying compiler directly (gcc, gfortran, icc, ifort, craycc, crayCC, crayftn).

Main results:
Performance data was not recorded.

3.3 Technical Support for DECI-10 in the 2nd Year of PRACE-3IP

All 37 DECI-10 projects started from 1 May 2013 in PRACE-2IP. T7.1.B continued the support for all DECI-10 projects since August 2013 after the end of PRACE-2IP. There were two DECI-10 enabling projects, DNSTF and HYDRAD, as reported in the following subsection.

3.3.1 *DNSTF*

The DECI-10 project DNSTF (Direct numerical simulation of finite size fibres in turbulent flow) has PDC as the home site and EPCC as the exec site. Both PDC and SURFsara contributed to the enabling support. The Tier-1 systems assigned to this project included HECToR @ EPCC, and ARCHER @ EPCC after HECToR finishing the service time. The enabling effort requested was 1-3 PMs.

Code general features

Name	*Slilab*
Scientific field	Lattice-Boltzmann method for the Solid-liquid interaction
Short code description	The *Slilab* code is based on the Palabos library, an open-source based on the lattice Boltzmann method. It has been developed for years, partly at the University of Geneva, and is proven to simulate accurately flow in complex geometries.
Programming language	C++
Supported compilers	GNU, Intel
Parallel implementation	MPI
Accelerator support	NA
Libraries	Boost, Eigen3, XmlTiny
Building procedure	NA
Web site	www.palabos.org
Licence	GNU Affero General Public License

Table 24: Code general features for Slilab.

Main objectives:

The objective is to parallelize *Slilab* code with OpenMP for a shared-memory execution model, when focusing on the multiphase phase flow simulation, as fiber suspensions in turbulent channel flow etc. In such problems the motion of the “second phase - fibre” is frequently crossed over the distributed domain boundary of the “first phase - fluid”, which in turn reduces the balance of MPI processing. Addition of OpenMP parallelization allows to minimize the number of MPI ranks in favor of single-node parallelism, therefore mitigating MPI imbalance.

Accomplished work:

We successfully parallelized *Slilab* code with OpenMP for a shared-memory execution model, when combined with its MPI parallelism allows us to run the code with hybrid parallelisation strategies to maximise the performance.

We identified serial bottlenecks with the help of Intel VTune Amplifier that required parallelization with OpenMP. All but one of the bottlenecks—the hotspot—allowed straightforward parallelisation with “#pragma omp parallel [collapse(2)]” clauses. Some of the functions make use of “static” variable which generated race conditions, but these were lifted by promoting these variables to become private for each thread. In addition, in the hotspot loop we also attempted the 2D parallelization strategy instead of original 1D which would, theoretically, allow it to scale to hundreds of thread, for example, on Intel XeonPhi. Finally, we also made the application NUMA-aware, thus allowing it to scale on multiple socket/CPUs on a single computing node.

Main results:

1. Compiler nuances

Most of the OpenMP parallelization, once the hotspots were identified, was straightforward. However, there were several nuances due to a partial compiler supports for OpenMP thread private variables. In particular, some of the functions used “static” variables to store data between invocations. However, when these functions were called from within parallel loops, race conditions produced erroneous results. A natural way to solve this was to use “#pragma omp threadprivate” clause for these variables. However, the variables were not of a plain-old-data type but rather C++ template classes. We found that while Intel C++ compiler [14.0.2] could digest such thread private variables, the GNU C++ compiler [4.8.1] generated a compiler error. A work around was to use C++11 “thread_local” keyword with GNU C++, which proved to work fine. However, Intel C++ compiler was unable to process this, and generated compiler errors instead. As a result, we opted for a compile-time conditional rules, such that with Intel C++ compiler the code used “#pragma omp threadprivate” clause after variable declaration, while with GNU C++ compiler the code takes advantage of C++11 “thread_local” keyword during variable declaration.

2. Scalability results

Here we present single-node code performance and scalability. Due to the bandwidth-bound nature of the application, it makes no sense to study its speed-up with the core count. The reason is that memory bandwidth is a resource shared by all cores on a CPU, and adding cores to the application above certain threshold wouldn’t provide any additional bandwidth. As such there is not any fundamental reason to expect linear application speed-up with the core count. It is important, however, to study application scalability as CPUs are added to the application. In particular, we test the code scalability on a 4-socket node with Sandy Bridge CPUs. Each CPU adds bandwidth to the system, therefore, in ideal case, the performance of the code should scale linearly with the number of CPU used. We also measure the memory bandwidth the application attains, and observer whether it scales linearly with CPU count. In addition, this bandwidth can be compared to the maximal practically achievable bandwidth which can be obtained with the Stream benchmark. To measure the bandwidth we used *likwid-perfctr* [39].

For the benchmarks we use Intel C++ compiler 14.0.2 and Intel MPI 4.1.0.24. We tested the code on a system equipped with 4x Sandy Bridge E5-4650 CPUs and four-channel 1600MHz DDR3 memory with peak bandwidth of 51.2GB/s per CPU. However, the maximal practically achievable bandwidth, which we measured with the Stream benchmark, is 35.2, 69.8, 104 and 138 GB/s for 1, 2, 3 and 4 CPUs.

We use the following environmental variables:

```
export OMP_SCHEDULE=static
export KMP_AFFINITY=compact,verbose
export I_MPI_PIN=1
export I_MPI_PIN_DOMAIN=omp:compact
```

These settings guarantee that the threads and processes will be closely packed together, and not spread across different CPUs. In particular, with 8 cores and 8 threads per CPU (8c8t for short), which also means that the system has no hyperthreading (HT) enabled, an application using with 8 threads and processes in total will only utilize resources of the CPU0. However, with 16 and 24 threads and processes, the application will also use CPU1, and CPU1 & CPU2 respectively.

To execute a hybrid version of the code, we use the following execution command:

```
OMP_NUM_THREADS=#threads mpirun –np #ranks ./cavity3d
#dimension
```

Here, "`#threads`" is the number of OpenMP threads per MPI rank, "`#rank`" is the number of MPI ranks, and "`#dimension`" is the problem size meaning that a three-dimensional box of size (`#dimension`)^3 is simulated.

Here we report strong scaling results for problem size of 128 and 256, running on from 1 to 4 CPUs.

`#dimensions = 256:`

#threads	#ranks	core/rank/Socket	Bandwidth [GB/s]	Mups[2]	Speed-up
8	1	8c1r1S	22.0	32.7	1.00
1	8	8c8r1S	22.3	30.9	0.94
16	1	16c1r2S	44.1	63.6	1.95
1	16	16c16r2S	45.4	58.0	1.77
24	1	24c1r3S	41.3	54.8	1.68
1	24	24c24r3S	70.0	83.7	2.55
4	6	24c6r3S	65.4	85.9	2.62
32	1	32c1r4S	84.8	119	3.63
1	32	32c32r4S	89.7	120	3.67
4	8	32c8r4S	87.9	118	3.61

Table 25: Strong scaling results of Slilab with #dimensions=256.

`#dimensions = 128:`

[2] Mups: Mega-updates per second, i.e. the so-called "science rate" of the application. This is a metric how many cells are updated per second in the lattice-boltzman method.

#threads	#ranks	core/rank/Socket	Bandwidth [GB/s]	Mups	Speed-up
8	1	8c1r1S	21.5	30.6	1.00
1	8	8c8r1S	18.4	31.4	1.03
16	1	16c1r2S	37.8	54.6	1.78
1	16	16c16r2S	37.1	55.5	1.81
24	1	24c1r3S	34.5	48.3	1.58
1	24	24c24r3S	57.5	76.4	2.50
4	6	24c6r3S	55.7	77.5	2.53
32	1	32c1r4S	67.7	94.5	3.09
1	32	32c32r4S	76.9	103	3.37
4	8	32c8r4S	63.1	115	3.76

Table 26: Strong scaling results of Slilab with #dimensions=128.

4. Discussion

We successfully parallelized *Slilab* with OpenMP for shared-memory execution model. Since the application is already MPI parallel, it allows us to maximise application performance by exploring treads/rank parameter space. We see from the tables that a hybrid parallelization approach allows us to achieve over 90% of parallel efficiency when scaled to 4 sockets on single node. Furthermore, if the problem per node is large enough, as with `#dimensions=256`, the pure OpenMP shows best result, which also demonstrates NUMA-awareness of the code; in particular, the bandwidth utilized by the code on 4S is nearly 4x of the 1S bandwidth for `#dimensions=256` case. The pure OpenMP performance is less satisfactory with smaller problem sizes. The primary reason is due to the remaining serial bottlenecks in the code that become further exposed. In principle, with additional efforts, these bottlenecks could also be parallelised to improve pure OpenMP scalability on smaller problems. However, we observe that is not required because when combined with MPI, the results demonstrate satisfactory scalability even on smaller problem sizes.

We were also able to test-drive this application on Intel XeonPhi with pure OpenMP using 240 threads on 60 Xeon Phi core. The performance was comparable to a single Sandy Bridge CPU. From our experience, this is not a bad result for an out-of-the-box code; however, this means there is at least 2x potential for improvement. The two basic reasons for this result is due to a) serial bottleneck of the OpenMP code which is much more exposed on XeonPhi than Xeon, and b) the compiler is unable to auto-vectorize the code that further aggravate single-threaded code; a XeonPhi core can be up to 10x slower than a Xeon core. We expect up to 3x speed-up on XeonPhi compared to a single Sandy Bridge-class CPU once these problems are solved.

3.3.2 *HYDRAD*

The DECI-10 project HYDRAD (Hydrodynamic stability of rotating flows in accretion disks) has RZG as the home site / exec site / enabling site. It also has VSB-TUO as one other exec site. The tier-1 systems assigned to this project included Hydra @ RZG and Anselm @ VSB-TUO. The enabling effort requested was 1-3 PMs.

Code general features

Name	*NSCOUETTE*
Scientific field	Hydrodynamics
Short code description	The *NSCOUETTE* code implements a hybrid-parallel direct-numerical-simulation (DNS) method for turbulent Taylor-Couette

	flow. The Navier-Stokes equations are discretized in cylindrical coordinates with the spectral Fourier-Galerkin method in the axial and azimuthal directions, and high-order finite differences in the radial direction. Time is advanced by a second-order, semi-implicit projection scheme, which requires the solution of five Helmholtz/Poisson equations, avoids staggered grids and renders very small slip velocities. Nonlinear terms are computed with the pseudo-spectral method. The code is parallelized using a hybrid MPI-OpenMP strategy, which is simpler to implement, reduces inter-node communications and can more efficient, compared to a flat MPI parallelization concerning memory, message sizes (latency) and MPI overhead at large core counts. A strong scaling study shows that the hybrid code maintains very good scalability up to more than 20,000 processor cores and thus allows to perform simulations at higher resolutions than previously feasible, and opens up the possibility to simulate turbulent Taylor-Couette flows at Reynolds numbers up to $O(10^5)$. This enables to probe hydrodynamic turbulence in Keplerian flows in experimentally relevant regimes, e.g. in accretion processes in weakly-ionized astrophysical disks. (see arXiv:1311.2481 for a detailed description of the code and benchmarks)
Programming language	FORTRAN
Supported compilers	ifort (Intel), gfortran (GCC), XLF (IBM)
Parallel implementation	hybrid MPI/OpenMP
Accelerator support	no
Libraries	BLAS, LAPACK (MKL, ACML, ...), FFTW (MKL or FFTW), pHDF5
Building procedure	make
Web site	-
Licence	-

Table 27: Code general features of NSCOUETTE.

Accomplished work:

- Implementation, validation and benchmarking of the new, fully thread-safe DFTs from MKL/11.1 versus the original FFTW routines (leading to a 5%-10% performance improvement, depending on the setups)
- Benchmarking of MKL transposition routines mkl_?omatcopy [3] versus the FORTRAN90 intrinsic transpose (a performance anomaly of was detected in mkl_?omatcopy from MKL/11.0 and was reported to Intel. The problem has been fixed in MKL/11.1 so that mkl_?omatcopy is now consistently at least as fast as the FORTRAN90 intrinsic)

[3] '?' denotes the datatypes: s=real, d=double precision, c=complex, and z=double complex[23]

- Algorithmic optimization of the computational performance and memory requirements of NSCOUETTE (leading to optimization of the "single-thread" performance and hence time-to-solution by a factor of 2 to 4, depending on the setup)
- Visualization of simulation results (basic enabling of the user for the VisIT and Paraview tools and remote visualization, production of a movie for analysis and dissemination)

Main results:

1) High-level optimization

For project HYDRAD a number of high-level optimizations were performed for the NSCOUETTE code in addition to standard application enabling. First, for the discrete Fourier Transform (DFT) the performance of the new, fully thread-safe implementation (which is a requirement for usage in NSCOUETTE) from the latest Intel Math Kernel Library (MKL, release 11.1) was evaluated against FFTW which has been used by default. With the DFTs from MKL, the overall performance was increased by 5% to 10%, depending on the setup.

The major task was the reorganization of the NSCOUETTE code such that LU decompositions for solving the five linear systems (which consumed more than 50% of the total runtime) are pre-computed once, at the beginning of a simulation. By this measure the overall computational performance per core could be nearly doubled for the typical matrix dimensions used (in the order of a thousand). The new implementation was thoroughly validated also by the original authors of the code and it has been used in production since the end of 2013. The optimizations enabled the scientists to run production applications at a higher radial resolution (factor 2-4). This, in turn, revealed a number of new "hotspots" in the profiling which were targeted for optimization as a next step. Specifically, a number of matrix-vector operations on the (banded) finite-difference coefficient matrices now take advantage of banded BLAS routines, replacing dense matrix-vector operations which were originally implemented and which had played a sub-dominant role for the overall computing time budget in the original version of the code and with smaller matrices.

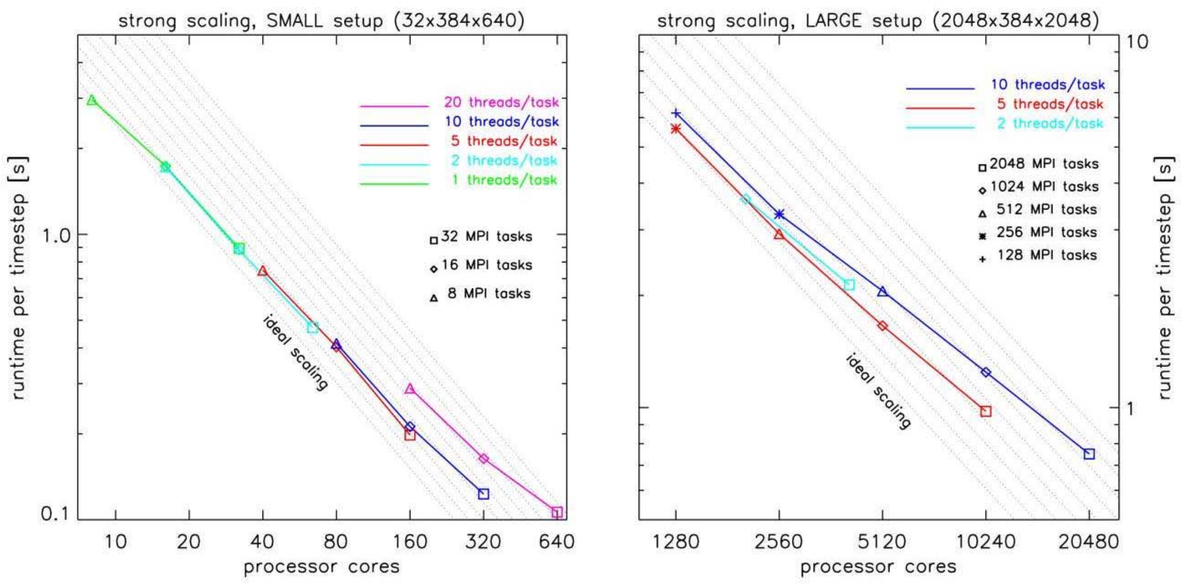


Figure 17: NSCOUETTE scaling up to 20,000 cores on Hydra @ RZG.

These algorithmic optimizations make the global transposition (implemented with MPI_Alltoall and local transposes) become even more dominant in the overall computing time and hence formal scalability at high core counts has actually *deteriorated* compared to the original code version. Nevertheless, the new version has been shown to scale up to more than 20,000 cores on Hydra @RZG (cf. Figure 17). Most importantly, however, the overall computational performance (in terms of time to solution for a given number of cores) could be improved by more than a factor of four for a given numerical resolution as a result of the optimizations reported here. This also allows the scientists to perform simulations with a significantly larger number of grid points with the same computing time budget.

2) Visualization and dissemination

For one of the highly resolved *NSCOUETTE* simulations of turbulence in Keplerian flows a movie was produced from more than 500 output files, generated by parallel HDF5 I/O. The visualization shows the evolution of turbulent vortices (Figure 18 is a snapshot from this movie showing a volume rendering of the streamwise vorticity, as a common measure for the strength of turbulence, produced with VisIT). The material was provided to PRACE for dissemination at ISC'14, SC'14, and for the PRACE web.

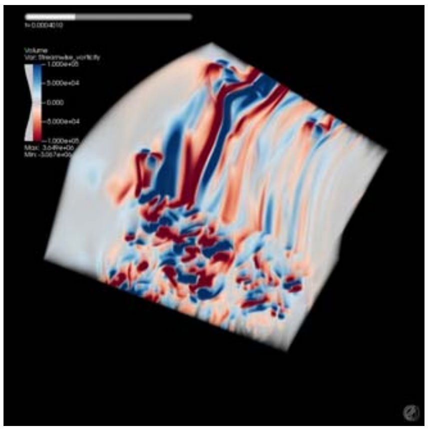


Figure 18: Snapshot of NSCOUETTE simulations of turbulence in Keplerian flows.

3.4 Technical Support for DECI-11

The DECI-11 call was launched on 8 May 2013 and closed on 10 June 2013. There were 119 proposals received and T7.1.B provided the TEs for all the DECI-11 proposals using the online PRACE PPR tool for DECI. 52 out of the 119 DECI-11 proposals were accepted. The accepted projects started from 1 November 2013 with a length of 1 year for the project time, i.e. the due date for the DECI-11 projects will be 30 October 2014. There was one DECI-11 enabling project, Planck-LFI3, as reported in the following subsection.

3.4.1 *Planck-LFI3*

Planck-LFI3 is the continued project for the DECI-9 project Planck-LFI2. CSC is the home site / exec site / enabling site for this project. The Tier-1 system assigned to this project is Sisu @ CSC. The enabling effort requested was 1-3 PMs. There has been no enabling progress implemented so far, but the project is planning to check the performance of the codes after the second phase of the Sisu@CSC upgrad in July, and optimise the code where necessary.

3.5 The following DECI calls

The DECI-12 call was launched on 18 December 2013 and closed on 20 January 2014. There were 61 proposals received and T7.1.B provided all the TEs for the DECI-12 proposals using the online PRACE PPR tool for DECI. 34 out of the 61 DECI-12 proposals were accepted. The accepted DECI-12 projects started from 1 May 2013 with a length of 1 year for the project time, i.e. the due date for the DECI-12 projects will be 30 April 2015. Due to the time scale of DECI-12 and PRACE-3IP, there will be no enabling effort available to support DECI-12 projects. T7.1.B was only responsible for providing the starting assistances for the DECI-12 projects.

At the time of this deliverable writing, DECI-13 is under planning as well but the timescale has not been confirmed yet. There will be no enabling support from T7.1.B for DECI-13.

4 T7.1.C Applications for Major Socio-economic Challenges

HPC techniques used for scientific computations have enabled several breakthrough research discovery and design innovations. Use of computing capabilities opens also new perspectives in solving key socio-economic challenges. Most of these problems are founded on in complex multi-discipline research areas that require a lot of investments and effective tools to be addressed. The PRACE HPC infrastructure delivers substantial computing capabilities and specialized applications that combined together are key factor for addressing socio-economic challenges that transform our ecosystem, economy, environment and everyday life.

Support for application codes addressing key socio-economic challenges and related scientific problem has been provided as a part of PRACE-3IP Project application support services task of Work Package 7. The overall approach for this task included: socio-economic challenges identification and associated application selection, required technical support for selected codes definition and actual enabling work targeting selected HPC systems. Representative set of applications has been evaluated and selected as a target of enabling for effective use on the PRACE HPC systems. This report contains complete descriptions of the enabling process for the selected application codes addressing identified socio-economic challenges. Report on the challenges identification and selection process has been included in the PRACE-3IP deliverable D7.1.1 *Applications Addressing Major Socio-economic Challenges* [24].

4.1 Work scope

Using numerous PRACE documentation and inputs from different European initiatives target scientific areas representing key socio-economic challenges have been defined. These have included following domains: *Energy Sources and Management*, *Life Sciences and Medicine*, *Climate Change*, *Big Data*, *Environment Protection*, and *Engineering*. With the help of PRACE Scientific Steering Committee evaluation and selection process of scientific problems representing socio-economic challenges have been completed resulting in a set of representative challenges in:

- Safe and Environmental-friendly energy production,
- Rational drug design,
- Sustainable food supply,
- Future aircraft transportation,
- ‘Big data’ management and processing,
- Understanding of climate change,
- Natural environment protection.

For each challenge a representative problem has been defined and an underlying scientific case extracted. These details have been evaluated with respect to scientific importance and representativeness using the input from the PRACE Scientific Steering Committee (SSC) members. For a given scientific case a suitable application or suite of applications has been selected with primary technical issues to be addressed. This leads to technical case formulation. On a basis of high-level socio-economic challenges and precisely defined scientific cases, technical cases for the associated application codes have been proposed. The proposals have been grouped in application enabling projects with project leader from PRACE WP7 members. Proposals were ranked using input from the SSC members. The evaluation ranking has been then approved by the PRACE PMO leading to the list of 12 socio-economic application enabling projects.

4.2 Work organization

Table 28 summarizes all application enabling projects grouped by a respective socio-economic challenge. The application codes are listed together with the project leader responsible for technical case definition and enabling work. Using this table plans for the enabling of the selected codes have been coordinated. For a complete description of identification and selection of these codes and initial goals for the application enabling please refer to the PRACE-3IP deliverable D7.1.1.

Socio-economic challenge	**Project Number**	**Project Leader**	**Associated applications**
Safe and Environmental-friendly energy production	PR2	Charles Moulinec (STFC DL)	TOMAWAC, TELEMAC-3D, Sisyphe (coupled)
	PR4	Jacques David (CEA)	URANIE
Rational drug design	PR5	Soon-Heum Ko (LiU)	LSDALTON
	PR7	Nevena Ilieva-Litova (NCSA)	DL_POLY_4
Sustainable food supply	PR8	Thomas Röblitz (UiO)	Bio Informatics applications workflow
Future aircraft transportation	PR9	Peter Stadelmeyer (JKU), Tomáš Karásek (VSB)	OpenFOAM, Elmer
‘Big data’ management and processing	PR11	Cevdet Aykanat (BILKENT)	Pegasus, Mahout
Multiscale modelling of the human	PR6	Andrew	PFARM

Socio-economic challenge	Project Number	Project Leader	Associated applications
cells and organs		Sunderland (STFC DL)	
	PR12	Eva Casoni (BSC)	Alya Red
	PR13	-	NEST
Understanding of climate change	PR14	Klaus Klingmueller (CASTORC)	EMAC/ ECHAM, MESSy (coupled)
Natural environment protection	PR15	John Donners (SURFsara), Massimiliano Guarrasi (CINECA)	Delft3D, Delft3D-DELWAQ, Delft3D-SWAN (coupled)

Table 28: Socio-economic challenges and associated applications enabling projects.

Each project focused on either one particular application code that need to be enabled for a given scientific case or on enabling a set of codes that need to be coupled for efficient use on HPC systems. All projects, with one exception, have reached the final enabling targets and detailed reports from these projects are included in this document.

Work on the project related to human neural system (project number 13, PR13) and associated application *NEST* (NEural Simulation Tool) has been cancelled. On the early stage of the project, the availability of scientific experts was not guaranteed. Because of high risk of the failure, it was decided to discontinue this project.

For each project technical goals have been defined and subsequent steps periodically updated during the work progress. Around 20 contributors were involved in this application support task. To coordinate work and track progress monthly teleconferences with all project leaders have been organized. Additionally, every quarter the project's progress were evaluated and a written report requested. Accordance to the enabling plans has been discussed with projects leaders. Evaluation steps have been also the subject of WP7 periodic face-to-face meetings where project leaders were delivering presentation on project progress on the dedicated session.

4.2.1 *Contributions*

A number of PRACE partners were contributing to the projects on “socio-economic applications” enabling. Project leaders were chosen from PRACE application experts in WP7 and all other members invited to contribute. Project leaders were coordinating contributions and enabling work. Responsibility for the external communication with experts and application community were on the project leader. It was agreed that leaders also deliver presentations and talks on the face-to-face session, quarterly progress report and a final report. Moreover, it was decided that each project will be completed with detailed technical report in a form of the PRACE white paper. Table 29 lists ten White papers already published on the PRACE web site. Details on the eleventh project focused on GPU accelerated version of the application for climate chemistry (PR14) is reported in the section 4.4 without a dedicated white paper.

	White paper title	Authors
WP167 [13]	Impact and optimum placement of off-shore energy generating platforms	C. Moulinec, D.R. Emerson (STFC DL)
WP168 [14]	Simulation Scaling in Nuclear Energy: Implementing Parallel Execution Framework for URANIE	J. David, V. Bergeaud (CEA)
WP169 [15]	Enabling Large-Molecule Simulations of Biological Interest through LSDALTON's DFT Method	Soon-Heum Ko (LiU)
WP170 [16]	Development of an Advanced Implicit Solvent Model in DL_POLY	N. Ilieva, P. Petkov, I. Todorov, D. Grancharov, E. Lilkova (NCSA)
WP171 [17]	Scaling of Biological Data Workflows to Large HPC Systems - A Case Study in Marine Genomics	T. Röblitz (UiO)
WP172 [18]	Fluid-Structure Simulations with OpenFOAM for Aircraft Designs	P. Stadelmeyer, T. Ponweiser (JKU), T. Karásek (VSB)
WP173 [19]	Petascaling Machine Learning Applications with MR-MPI	C. Aykanat (BILKENT)
WP174 [20]	HPC Programming to Generate Electron-molecule Resonance Data for DNA Radiation Damage Studies	A. Sunderland, M. Plummer (STFC DL)
WP175 [21]	Multidisciplinary Coupling in Cardiac Computational Mechanics	E. Casoni, M. Vázquez, G. Houzeaux (BSC)
WP177 [22]	Optimization of a Coupled Simulation with Delft3D-FLOW and SWAN for Informed Decision Making	J. Donners (SURFsara), M. Guarrasi, A. Emerson (CINECA)

Table 29: White papers on the applications and socio-economic challenges.

Technical support for applications including code porting, performance optimization, profiling, scalability enabling, I/O tuning and identification and fixing of bottlenecks was performed by either project leader or his contributors. There were also substantial efforts on optimized or accelerator enabled computational modules, routines or kernels development reported. Summary of the efforts used for the identification, selection and support work reported by the project leaders is presented in Table 30. A number of external collaboration efforts have been also established including inputs from University scientists and code developers community experts. These contributions have been used to design the projects objectives and verifying the outcomes of technical work performed.

Socio-economic challenge	Project Number	Total PM
Safe and Environmental-friendly energy production	PR2	4 PM (STFC)
	PR4	2 PM (CEA)

Socio-economic challenge	**Project Number**	**Total PM**
Rational drug design	PR5	4.5 PM (LiU)
	PR7	4 PM (NCSA)
Sustainable food supply	PR8	13 PM (UiO)
Future aircraft transportation	PR9	9 PM (JKU) 4 PM (VSB)
'Big data' management and processing	PR11	5 PM (BILKENT)
Multiscale modelling of the human cells and organs	PR6	3.5 PM (STFC)
	PR12	9 PM (BSC)
Understanding of climate change	PR14	6.5 PM (CASTORC)
Natural environment protection	PR15	2 PM (SURFsara) 7 PM (CINECA)
Total (including coordination)		**80 PM**

Table 30: T7.1C efforts summary.

4.2.2 *Tier-0 resources access*

Following internal PRACE guidelines on Tier-0 systems, Preparatory Access framework has been used for Tier-0 systems access, using the Type B procedure. Type B access is granted without support from PRACE experts. It was decided that for each six-month access period a cumulative application for resources will be submitted. The reasons for such approach were: formalities reduction and balance of the core hours share between the projects. Three cut off periods from January 2013 to June 2014 for Tier-0 systems access were used (11th cut off – granted 15th January 2013, 13th cut off – granted 15th July 2013, 15th cut off - granted 15th January 2014), what secured contiguous access to the systems and uninterrupted work progression. Before each cut off project leaders were requested to estimate required core hours and define target systems. Applications for resources were then coordinated by the task leader. Table 31 summarizes granted core hours and systems eventually used for the applications enabling.

System	**Budget [core hours]**		
	11th Cut Off	*13th Cut Off*	*15th Cut Off*
CURIE TN	200.000	365.000	450.000
CURIE FN		35.000	
CURIE HYBRID	-	-	100.000
MARENOSTRUM	-	100.000	100.000
JUQUEEN	250.000	500.000	250.000
FERMI	250.000	250.000	250.000
Total	700.00	1.250.000	1.150.000

Table 31: Tier-0 allocations granted.

For each access period a given core hours allocation was available and the distribution among projects was agreed. While each project used a different work plan and also specific tasks, core hours were distributed not equally but rather on demand. This approach helped to spend core hours practically and respect the needs all of all contributors.

Not only Tier-0 systems were used while significant amount of work was first focused on application refactoring. At this stage, Tier-0 capabilities were not required and smaller Tier-1 or local systems were used. It was not monitored and considered as partners' contribution.

4.3 Approach for socio-economic applications enabling

General approach applied to application codes identified as addressing socio-economic challenges was problem driven. Each project was focused on addressing a particular scientific case rather than entire application scope. Starting point for the enabling work was a precise scientific case and the scope of the application use corresponding to the given challenge. With scientific input and application expertise, project leaders have been then responsible for detailed technical case definition. Additional input from external collaborators including expert scientists, application developers or community experts was used.

Following list presents support approach for considered socio-economic challenges explaining scientific and technical goals and methods applied. Changes to the initial plan and application study, if any, have been explained in the remarks description. Detailed evaluation reports for application codes listed here are included in the next section of this deliverable.

- **Safe and Environmental-friendly energy production** (PR2 and PR4)
  - Scientific case: Study the impact and optimum placement of off-shore energy generating platforms and to simulate the placement of marine turbines in coastal waters.
  - Technical case: Triple coupling of wave propagation, three-dimensional hydrodynamics and sediment transport distribution application modules for large scale simulations of a local (0.1 to 1 meter) and a distant (1 to 10 km) impact.
  - General approach: Application coupling, mesh multiplication, code scaling improvement.
  - Applications supported: *TELEMAC* suite.

  - Scientific case: Study of a nuclear reactors safety parameters using numerical simulations with Verification and Validation and Uncertainties Quantification analysis.
  - Technical case: Scalability and portability improvement for parallel coupled simulations with *URANIE* framework.
  - General approach: Runtime tuning and scaling enabling.
  - Application supported: *URANIE* uncertainty quantification (VVUQ) platform.
- **Rational drug design** (PR5 and PR7)
  - Scientific case: Large scale molecular simulations using linear-scaling Density Functional Theory (DFT).
  - Technical case: Improvement of DFT application code performance with applying density fitting (DF) scheme and auxiliary density matrix method (ADMM).
  - General approach: application refactoring, performance tuning, scaling tests and improvement.
  - Applications supported: *LSLDATON* (Linear Scaling *DALTON*).
  - Scientific case: Enabling large scale periodic boundary-free simulations with implicit solvent model in molecular dynamics of protein salvation processes.

  - Technical case: Integration of advanced implicit solvent model library AGBNP2 with molecular dynamics application *DL_POLY*.
  - General approach: application development and refactoring.
  - Applications supported: *DL_POLY_4*.
- **Sustainable food supply** (PR8)
  - Scientific case: Investigation of the genomic basis of phenotypic traits using fish genomes for analysis of feed efficiency, growth, age at maturation, disease resistance and product quality.
  - Technical case: Implement scalable HPC workflow for the analysis of sequenced data.
  - General approach: Application workflow design and implementation, scalability enabling.
  - Applications supported: Bioinformatics workflow consisting of codes: *bwa*, *cutadapt*, *FastQC*, *GATK*, *mapdamage*, *picard-tools*, *SAMtools*, *sysstat*.
- **Future aircraft transportation** (PR9)
  - Scientific case: Multi-physics simulations for aeronautics design using fluid-structure interactions (FSI) .
  - Technical case: Implementation of a partitioned, strongly coupled solver for transient FSI simulations with independent meshes for the fluid and solid domains using *OpenFOAM* application.
  - General approach: computational kernel development, scalability enabling.
  - Applications supported: *OpenFOAM*.
  - Remarks: General assumption has been made on computing fluid-structure interactions with two independent meshes and this entirely using *OpenFOAM* application. It turned out that the solution of CSM, where *Elmer* application usage was considered, has a very minor influence on the overall runtime or the scalability. The two most important tasks regarding performance are updates of fluid mesh and solution of CFD part. It was decided to concentrate work on those two performance bottlenecks, which are related to *OpenFOAM* only. Using *OpenFOAM's* functionality to handle several independent meshes simultaneously, no change in initial goal of coupling two independent models was required.
- **'Big data' management and processing** (PR11)
  - Scientific case: Study large-scale machine learning problems with Map/Reduce parallel programming paradigm.
  - Technical case: Enabling of processing of terabytes of data on Tier-0 HPC systems using Map/Reduce parallel programming paradigm on fundamental data mining approaches.
  - General approach: Enabling algorithms implementation and scalability test for Tier-0 systems.
  - Applications supported: *MR-MPI* library.
  - Remarks: Initial analysis have investigated Mahout and Pegasus which run in a distributed fashion using MapReduce paradigm. These tools use Hadoop MapReduce and for communication, utilize RPC calls using slow UDP/TCP which hinder scalability in Tier-0 HPC systems. According to later analysis, it has been found that these tools (*Mahout* and *Pegasus*) do not fit HPC systems neither in terms of efficiency nor scaling for more than a few hundreds of cores. For this reason, it was decided to switch to more efficient tool *MR-MPI* which is specifically tailored for such systems, and implemented the above mentioned

machine learning algorithms. That approach enabled scaling of the algorithms up to 2000 processors for the *MR-MPI*-based implementation on Tier-0 HPC system.

- **Multiscale modelling of the human cells and organs** (PR6 and PR12)
  - Scientific case: Study of the DNA oxidative damages and RNA damages caused by strand breaks and associated electrons resonances.
  - Technical case: Replace a serial propagator (coupled PDE solver) with a parallel equivalent module in the electron-molecule collision package for more detailed resonance formation studies. Compute more detailed representations of electrons trapped in quasi-bound resonances which are associated with strand breaks in DNA molecules.
  - General approach: Replace a serial propagator (coupled PDE solver) with a parallel equivalent module in the electron-molecule collision package for more detailed resonance formation studies.
  - Applications supported: Resonance Finding and Resonance Fitting *TIMEDEL* code.
  - Remarks: The original goals of the project were to incorporate parts of the highly parallelised external region *PFARM* code into the *TIMEDEL* modules. However, on subsequent analysis of the serial modules it became clear that this was probably too much of an undertaking for the relatively limited effort associated with this project. A more pragmatic approach subsequently has been undertaken, where more straightforward, yet still effective parallelisation, have been applied to the original *TIMEDEL* application code.

  - Scientific case: Simulation of the cardiac mechanics at organ level with a focus on the cardiovascular mechanisms and their physiological models.
  - Technical case: Development of the fluid-electro-mechanical coupling in the high performance computational biomechanics simulation tool to simulate cardiac mechanics.
  - General approach: Computational modules coupling, application refactoring, scaling improvements.
  - Applications supported: *Alya* and *Iris* mesh generator.
- **Understanding of climate change** (PR14)
  - Scientific case: Simulate the atmospheric chemistry as a part of climate modelling.
  - Technical case: Enable GPU acceleration for atmospheric chemistry sub-model.
  - General approach: Code refactoring.
  - Applications supported: *KPP* (Kinetic PreProcessor), *EMAC* chemistry-climate model, *MECCA* (Module Efficiently Calculating the Chemistry of the Atmosphere).
- **Natural environment protection** (PR15)
  - Scientific case: Simulate Study lake environment with interactive lake design approach.
  - Technical case: Enable a coupled software application to simulate the different aspects of a lake, ranging from the optical properties of the water to safety concerns.
  - General approach: Enabling of the interactive application code for HPC usage, code refactoring and porting.
  - Applications supported: *Delft3D-FLOW*, *Delft3D-WAVE* and *SWAN*.

4.4 Detailed report on application enabling

This section contains complete and unified reports on projects supporting selected application codes. For specific details on technical work done and more in-depth study of the results please refers to the corresponding PRACE white paper (see Table 29:).

4.4.1 *PR2*

Code general features

Name	The *TELEMAC-MASCARET* suite
Scientific field	Hydrodynamics, Wave propagation, Sediment transport
Short code description	*TELEMAC-MASCARET* is an integrated suite of solvers used to simulate free-surface flows, in 1-D, 2-D or 3-D.
Programming language	Fortran 90, Python, Perl
Supported compilers	GNU, Intel, XL
Parallel implementation	MPI
Accelerator support	N/A
Libraries	*METIS* and *ParMETIS*
Building procedure	The code might be built using Python all the way, or Perl and Makefile.
Web site	http://www.opentelemac.org
Licence	GPL

Table 32: Code general features for TELEMAC suite.

Main objectives:

This project aims at preparing the hydrodynamic *TELEMAC-MASCARET* suite of solvers for large scale simulations, involving wave propagation, three-dimensional hydrodynamics and sediment transport distribution. The triple coupling *TOMAWAC-TELEMAC-3D-SISYPHE* has been tested for large meshes (over 10 million elements).

Accomplished work:

The *TELEMAC-MASCARET* suite was ported onto the MARENOSTRUM Tier-0 system (Bull) and the ARCHER Tier-1 system (Cray XC30) using the Intel compiler on both machines. A significant part of the work consisted of testing first *TOMAWAC*'s performance in parallel, before focusing on the triple coupling. An issue was raised in *SISYPHE*, in the *BEDLOAD_DIFFIN* subroutine, where the boundary conditions are set. The case where an element has only one node as a physical boundary but where the edge containing this node is an interface node was not taken into account. This is now fixed.

The *TELEMAC-MASCARET* suite allows serial pre-processing (using a tool call *PARTEL*) or parallel (using tools called *PARTEL_PRELIM* and *PARTEL_PARA*). Serial pre-processing was used except for the largest meshes, where parallel pre-processing was required. The parallel pre-processing stage did not support double precision input before this project. This has been implemented in *PARTEL_PRELIM* and *PARTEL_PARA*.

Main results:

The main results are presented for the largest mesh of 10,454,016 elements and 5,248,620 nodes. The number of targeted processors is 1,024 – 2,048. First results (see Table 33) show that *TOMAWAC*'s speed-up is super-linear up to 2048 cores on MARENOSTRUM (4,17 from 512 to 2,048 cores).

Number of cores	Time (s)	Speed-up	Efficiency
512	8,671	1.00	100
1024	2,936	2.95	147
2048	2,076	4.17	104

Table 33: TOMAWAC: 5th LEVEL – 7,200 time-steps – 6.21 s per time-step (MARENOSTRUM)

Table 34 shows the performance of the triple coupling *TOMAWAC-TELEMAC-3D-SISYPHE* obtained for the same mesh as the one used to test *TOMAWAC*. Results are obtained on ARCHER system. A speed-up of 1.59 is observed from 768 to 1,536 cores, but of 1.60 only going from 768 to 3,072 cores. A CrayPat performance analysis tool showed that the collective *MPI_ALLTOALLV*, used in the method of characteristics, was a dominating scaling obstacle for a large number of cores (3,072). This explained why the speed-up was not better than 1.60 going from 768 to 3,072 cores.

Number of Cores	Time (s)	Speed-up	Efficiency
768	1,819	1.00	100
1,536	1,144	1.59	78
3,072	1,139	1.60	40

Table 34: TOMAWAC-TELEMAC-3D-SISYPHE: 5th LEVEL – 1,200 time-steps (ARCHER)

The coupling between the three codes was only running in serial at the beginning of the project because *TOMAWAC* was not running in parallel. During the project we received a new parallel version provided by the main developers, EDF, and the coupling was claimed to work on 128 processors.

This was tested for small cases, and a few bugs were corrected. Then following the objectives of the project to enable larger simulations (several millions of elements going from a few kilometres to a few centimetres), larger meshes were created. Testing and debugging were carried out to lead to a simulation of up to 10 million 2-D elements on up to 3,072 cores. Demonstrating that this size of mesh is currently achievable by the code, including pre- and post-processing should encourage the users to run the *TELEMAC* suite using at least Tier-1 machines.

4.4.2 *PR4*

Code general features

Name	*URANIE*
Scientific field	Information analysis, machine learning, nuclear power
Short code description	Uranie is a sensitivity and uncertainty analysis platform based on the ROOT framework (http://root.cern.ch) . It is developed at CEA, the French Atomic Energy Commission (http://www.cea.fr).
Programming language	C/C++

Supported compilers	GCC
Parallel implementation	MPI (OpenMPI)
Accelerator support	No
Libraries	ROOT, nlopt, pcl, swig, parmentis, cppunit, libxslt, opt
Building procedure	make {-with params}
Web site	http://sourceforge.net/projects/uranie
Licence	GNU Library or Lesser General Public License version 3.0 (LGPLv3)

Table 35: Code general features for URANIE (2).

Main objectives:
While *URANIE* is well suited for launching many instances of serial codes, it suffers from a lack of scalability and portability when used for coupled simulations and/or parallel codes. Aim is to enhance this launching mechanism to support a wider variety of applications, with target of 100's to 1000's, possibly 10_000's cores.

Accomplished work:
Three strategies are studied:

- "system launcher" mechanisms: using *mpirun* for the initial resource allotment, launching of a monitor, which will manage parallel tasks using low level basic system mechanisms such as fork-exec for secondary launching of the individual simulation-point process,
- generic MPI launching: using *mpirun* plus generic MPI library, including sub-parallelism such as OpenMP or fork-exec as suitable for the individual simulation-point process,
- SLURM "step" launching: using *srun* for sub-partitioning the resources then *mpirun* as above.

With that in mind, two experiments have been done on the flowrate "toy" code. The first experiment was involving a number of jobs equal to the number of allocated cores; this is a straightforward set-up. On the second one, the number of jobs was doubled, in order to enable better compute resources usage by some over-allocation. In both experiments, the output files have a limited size (a few thousand bytes). The number of cores was doubled incrementally from 16 to 2048.

Main results:
On the first experiment, the scalability is good for all three strategies up to 512 cores; it degrades for the SLURM based one beyond that number Figure 19 shows results for "straightforward *njobs=ncores* (weak scalability)" approach of performance for the three strategies (time in minutes - lower is better).

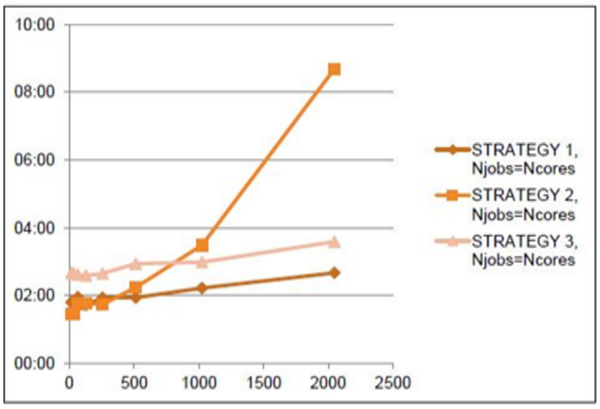


Figure 19: Results of the first experiment.

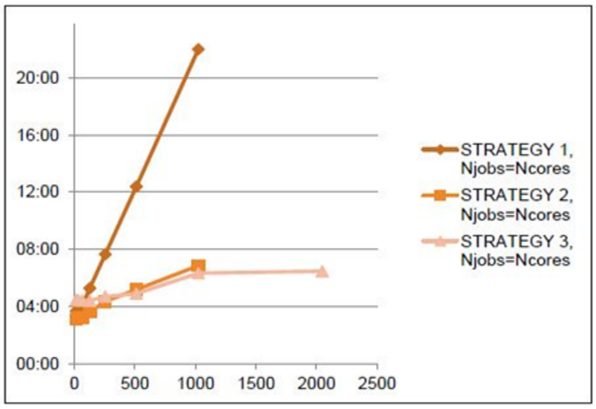


Figure 20: Results of the second experiment.

The second experiment shows the limits of the two first strategies which are based on scheduling on the master node via system-level mechanisms: they require heavy scheduling tasks to define which jobs have stopped and which should be restarted. In that respect, the SLURM based one is more robust than the first strategy. MPI-based strategy which involves message passing for scheduling is much more robust and efficient for this test case, achieving very good scalability up to 2048 nodes. Figure 20 shows "over-allocation *njobs=2*ncores* (weak scalability)" case performance for the three strategies (time in minutes - lower is better)".

Further experiments with larger output files suggest that the key factor to the limitation of the scalability is the intensity of the I/O workload.

4.4.3 *PR5*

Code general features

Name	*LSDALTON* (Linear Scaling *DALTON*)
Scientific field	Quantum Chemistry
Short code description	*LSDALTON* was born with the ultimate research goal of linear-scaling Density Functional Theory (DFT) in order to treat large molecular systems. *LSDALTON* has therefore served as a framework to test and develop new linear-scaling methods. The novelty of the *LSDALTON* program have made it fairly easy to fully exploits the latest Fortran programming schema and tune it for modern computing architectures. *LSDALTON* does not follow point group symmetry, which enables this code to be applied to analyses of large molecular systems. *LSDALTON* does however exploit integral-screening techniques which become increasingly important with molecular size, in combination with integral-acceleration techniques and highly efficient integral approximations. *LSDALTON* is fully atomic-orbital (AO) based, allowing for asymptotic linear-scaling treatment of large molecular systems in wave-function optimization, molecular gradients and various response-function calculations.
Programming language	Fortran
Supported compilers	Intel, GNU and PGI compilers

Parallel implementation	MPI and OpenMP
Accelerator support	N/A
Libraries	Linear algebra libraries (either ScaLAPACK or LAPACK/BLAS)
Building procedure	(Configuration in case of MPI+OpenMP, 64-bit Integer, ScaLAPACK) ./setup --fc=mpif90 --cc=mpicc --cxx=mpicxx \ --int64 -DCPP="-DHAVE_NO_LSEEK64" --type=release \ --mpi --omp --scalapack --blacs=openmpi \ build-dir (Compilation) cd build-dir; make (Imposed Flags in Intel Compiler) Fortran: -w -fpp -assume byterecl -DVAR_IFORT -openmp -parallel -i8 -O3 -xW –ip C: -g -wd981 -wd279 -wd383 -vec-report0 -wd1572 -wd1777 -restrict -DRESTRICT=restrict -openmp -O3 -ip
Web site	http://www.daltonprogram.org/
Licence	Free distribution to individuals upon the personal licence agreement; Site licences on the condition of keeping the source code confidential; Benchmark licences valid for a restricted period of time. All license agreement forms are applied from their homepage.

Table 36: Code general features for LSDALTON.

Main objectives:

In this work we aimed for examining the performance of linear-scaling DFT method in the *LSDALTON* code to enable the simulation of large biological molecules. We made an intensive investigation on the code performance at various MPI runs sizes and for different molecules. We primarily focused on finding obstacles associated with density-fitting (DF) and auxiliary-density-matrix method (ADMM), because these two approximations are essential for achieving highly efficient computations of large biological molecules in Tier-0 scale.

Accomplished work:

The main effort was towards enabling large molecular simulations with the latest *LSDALTON* implementation. Debugger (*Allinea DDT*) and memory monitoring tools (in-built implementation and system-provided statistics tool) reported that a significantly large amount of memory is allocated to internal tensor structure and Kohn-Sham matrices in case of applying DF scheme. The former is probably relieved by addressing a light-weight tensor structure at future release of the code, whilst the latter is hard to resolve without a new scheme. Therefore, we have focused on finding the right build environment (composition of compiler, MPI and linear algebra library) that entirely supports 64-bit integer formulation.

While compilers are highly matured to provide true 64-bit integer, MPI and linear algebra packages cause trouble. Intel MPI fails to perform the message passing over 2 GB array since the message size parameter is fixed as 32-bit integer. Bullxmpi on CURIE defines Fortran integers as 32-bit, thus naturally leads to the failure in handling large size arrays. The open-source OpenMPI library fully supports message passing of large datasets if it is installed with 64-bit integer representation. In view of linear algebra libraries, both open-source

ScaLAPACK and the encapsulated one in Intel MKL package fail to allocate large arrays in excess of 32-bit integer range. Intel MKL's LAPACK/BLAS interface was stable regardless of matrix size.

On top of the compiler chain, we evaluated the performances of DF scheme and ADMM approximation by solving three biological molecular structures of valinomycin, titin and insulin. On valinomycin simulation who consumes ~2.5 GB memory space per MPI rank, the performance gain by ScaLAPACK has been primarily investigated. On titin (16 GB memory allocation per MPI rank) and insulin (62 GB memory allocation per MPI rank) simulations, we focused on comparing performances by DF, ADMM with DF, and non-DF DFT calculations.

Main results:

Valinomycin results show that the scalability is improved by using ScaLAPACK library. In the example of DF calculation, speed-up at 32 (MPI rank) × 8 (OpenMP thread) cores is 10.41 (w. ScaLAPACK) and 8.76 (w. LAPACK/BLAS) in comparison to 2×8 cores. That was the same in other schemes and in pure MPI computation. The effect was stronger at larger numbers of CPU cores.

Notable performance improvements are observed with DF and ADMM, without noticeable degradation of the convergence criteria. In all three cases, the runtime for a single Kohm-Sham matrix construction iteration was shorter with DF and further accelerated with ADMM. In view of scalability, the tendency reverses: the best speed-up is achieved at non-DF calculation. As observed from Figure 21 and Figure 22 below, non-DF calculation shows the speed-up of 7.71 (comparing between 64×8 cores and 4×8 cores) in titin simulation and 6.38 (comparing between 128×16 and 8×16 cores) in insulin simulation. They decrease to 6.23 and 3.57, respectively, with the DF scheme. That is indeed natural because the DF scheme reduces large amount of computation by replacing a product operation between single particle basis set functions with the auxiliary basis set. Since the amount of computation has decreased, DF scheme is more dominated by communication overhead at large number of cores. While the calculation with DF still shows the similar scalability pattern to non-DF calculation at small number of CPU cores, DF with ADMM shows a highly non-scalable performance gain. It results that DF-ADMM provides the best performance at small number of CPU cores while DF run overtakes at large number of CPU cores (1024 in titin and 4096 in insulin). All three models are overwhelmed by communication overhead at the largest number of cores (2048 in titin and 4096 in insulin) simulations. Since insulin is considered as the largest biological application in view of memory requirement, 2K cores seem a limit of a reasonable parallel computation size.

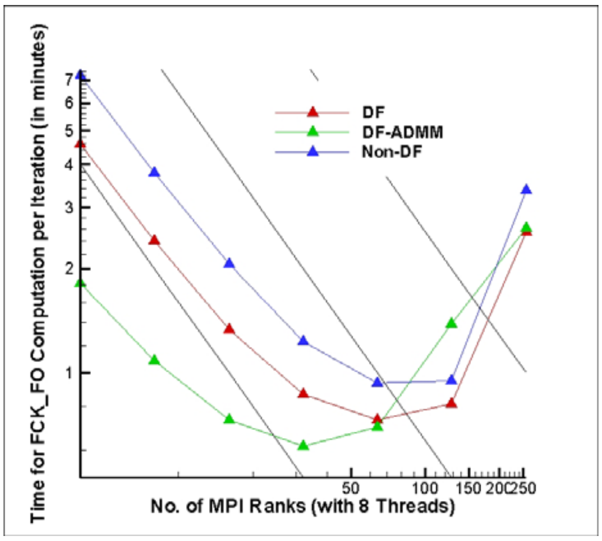


Figure 21: Elapsed Time in Kohn-Sham Matrix Construction for Hybrid Simulation of Titin molecule. Time measured at the 2nd Iteration.

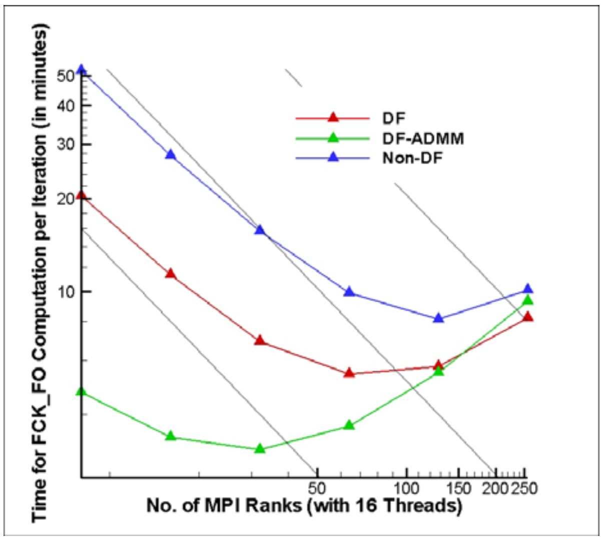


Figure 22: Elapsed Time in Kohn-Sham Matrix Construction for Hybrid Simulation of Insulin molecule. Time measured at the 2nd Iteration.

DF and ADMM introduce extra overhead at the non-iterative part of the code. It is natural since these schemes allocate extra matrix components for auxiliary basis set and resultant cost for broadcasting also increases. Thus, large gain on Kohn-Sham matrix construction with DF and DF-ADMM lessens in view of total simulation time. Still, DF-ADMM calculation with 32×8 cores was shortest and the gap is expected to increase in insulin calculation. Furthermore, this gain by using DF or DF-ADMM scheme will increase by redesigning internal tensor structure to a lighter fashion.

4.4.4 *PR6*

Code general features

Name	*TIMEDEL* in *UKRmol*
Scientific field	Electron-atom, electron-molecule scattering resonance finding and fitting program, DNA & RNA base molecule radiation damage.
Short code description	The specific part of *UKRmol* adapted is the module *TIMEDEL*, which explores and illuminates resonance features of electron molecule interactions. This module is called for a range of molecular geometries and sets of incident electron energies. The full R-matrix simulation takes place in several separate stages. Configuration space is divided into two regions by a sphere, which contains all the 'target' electrons. A full CI calculation involving continuum states and orbitals as well as bound states and orbitals takes place within the sphere. This is independent of scattering energy and is performed once for each geometry: it principally involves the setting-up and diagonalization of a large Hamiltonian matrix. The outer region calculations take place separately and quite possibly on another machine. The theory is of one electron moving in a multichannel potential arising from all of the CI states and channels included in the inner region calculation. Various options may be chosen to calculate collision parameters and other quantities. The *TIMEDEL* module can be run as a self-contained outer region

	package, particularly for the intense resonance fitting tasks of interest to our collaborators, and this is how we use it here. This code development facilitates a more detailed representation of electrons trapped in quasi-bound 'resonances' on the base components of DNA, which are implicated in cell damage.
Programming language	C, Fortran 77/95/2003
Supported compilers	Intel, IBM, GNU, Cray
Parallel implementation	Outer MPI-based parallel harness to control task scheduling. MPI in code.
Accelerator support	None at present – though calls to serial linear algebra operations could be replaced by calls to accelerator-enabled library routines.
Libraries	Lapack or vendor specific equivalent (IBM Essl, Cray LibSci, Intel MKL).
Building procedure	Compile outer, util libraries and harness wrapper from Fortran & C source code, then link together with the top level code (outermpi.f03) mpixlf2003_r -O3 -qfixed -o outermpi outermpi.f03 c_wrapper.o chdir_c.o -louter -lutil –llapack
Web site	http://ccpforge.cse.rl.ac.uk/gf/project/ukrmol-out/
Licence	GNU General Public License (GPL)

Table 37: Code general features for UKRmol.

Targets and accomplished work

This project has optimized and parallelised the resonance modules of the *UKRmol* [25] package, which will provide enhanced computational resources for collaborating research groups to study biological molecular resonance formation in much more detail than current initial (though impressive) calculations ([26],[27]) on DNA and RNA base molecules. In addition to the main thrust of this work on applications directly related to cell radiation damage and strand-breaking, the general application of the enhanced package will have impact in other socio-economic fields (for example, dissociative recombination studies relevant to atmospheric and aeronautical physics and chemistry). The parallelisation work has taken place in two stages, each of which is summarised in the following sections. Parallel performance tests have been undertaken on the JUQUEEN IBM Blue Gene/Q system and the Hartree Centre systems (IBM Blue Gene/Q and IBM iDataplex).

High-level parallelization

Main objectives:

For detailed resonance searches the large number of energies involved means that calculations for each geometry are quite extensive. Our collaborators were previously running in serial mode, which for the detailed resonance mapping required would have taken far longer than is practical (for example, for a PhD student, or indeed a fixed-term PDRA). The objective was to parallelize the code such that full calculations for large numbers of geometries could be performed in a short or at least manageable time. To this end the first stage objective was to speed-up significantly the process of finding and fitting of resonance from large-scale scattering calculations by distributing different molecular geometries amongst MPI tasks within a large-scale parallel job. The software development was designed to make this procedure as automatic and efficient as possible for users across a range of target high-end parallel platforms.

Accomplished work:

The first stage of the optimization work introduced an overall parallelism for these calculations with a parallel harness to run distinct geometries and ranges, invoking the serial code simultaneously for each dataset in separate folders. Several sets of scripts and codes were developed for the control programs, as our two alternative target hardware platforms (IBM Blue Gene/Q and IBM iDataplex) necessitated different approaches:

- MPI-based C control program incorporating the *exec()* family of Linux functions which enable the replacement of current process images with new process images.
- MPI-based Fortran 2003 control program with *iso_c_binding* calls to C functions for command line operations.

The first approach is suitable for most HPC systems with Linux kernels operating on the application nodes. However IBM utilize a 'Lightweight Compute Node Kernel' on the Blue Gene application nodes that does not support the required functions, hence the development of the second approach for this and other platforms.

Main Results:

The parallel code has been tested on a relatively simple and known case (e-N2+ dissociative recombination resonances) in conjunction with scientists from the AMOPP research group at University College London (UCL) [28]. The performance results for runs on up to 1024 cores are sampled from datasets across the full geometry range in order to be as representative as possible of the varying computational loads associated with each geometry.

The high-level parallelisation involves looping over data representing 1024 different geometries for dissociative e-N2+ scattering calculations. In each case the number of geometries increases with core count and therefore it is the weak scaling properties of this approach that are reported in Table 38 and represented graphically in Figure 23.

Number of Blue Gene/Q Cores Used	Number of Blue Gene/Q Nodes Used	Geometry Calculations (full energy range)	Time to Completion (secs)	Time taken relative to 16 core case
16	1	16	4781	1
64	4	64	6637	1.39
256	16	256	7106	1.49
1024	64	1024	7839	1.64

Table 38: Speed-up obtained from Stage 1 parallelisation of the 1024 geometry calculations.

For this large-scale dataset, the approach scales well up to 1024 cores on the IBM Blue Gene/Q, with the 1024 geometry calculation time around 64% slower than the 16-geometry calculation. Analysis has shown that it is the varying computational load across geometries that most impacts upon parallel efficiency at higher core counts. The parallelisation mechanism has been achieved with minimal reworking to the original code and is applicable generally to application codes that perform large numbers of (mostly) independent tasks to generate 'mapping' data. Studies have also been made to devise a 'production' job submission algorithm that can optimize the ratio of cores used for the 'external' harnessing and the 'internal' programmed parallelisation with final verification tests on a DNA/RNA base (adenine, guanine, uracil) to be set up following this project. The parallelised code will then be used to enable new more realistic and detailed DNA/RNA base resonance studies, which are planned but are not currently feasible.

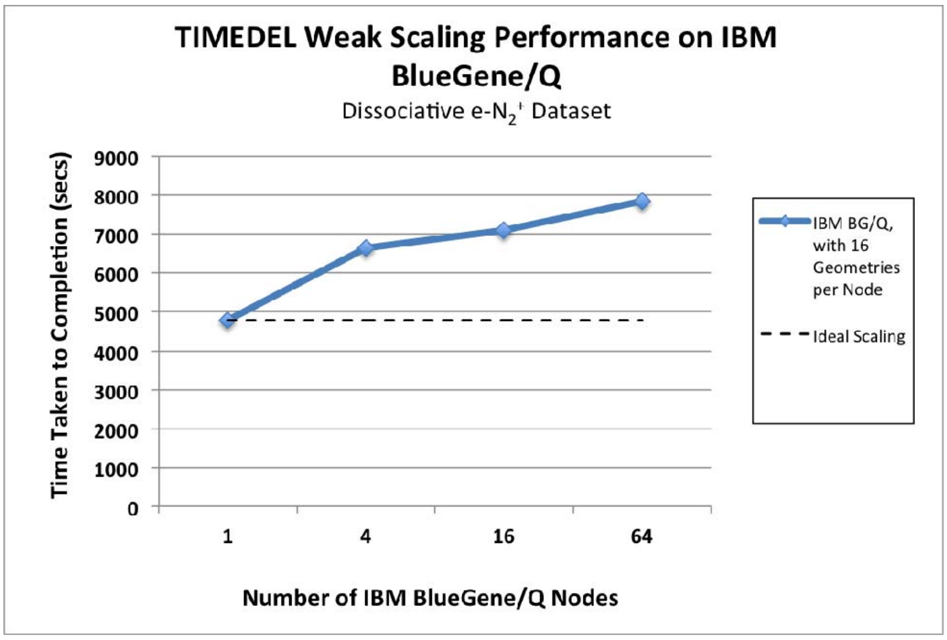


Figure 23: Speed-up obtained from Stage 1 parallelisation of the 1024 geometry calculations.

Low-level parallelization

Main objectives:

The second part of the parallelisation work aimed to introduce lower-level parallelisation to inner loops in the *TIMEDEL* module suitable for either MPI or OpenMP parallelisation. An analysis of the code identified the automated resonance search as having a suitable loop structure for MPI parallelisation.

Accomplished work:

MPI-based parallelisation work has been applied to the energy grid in *TIMEDEL*, which is sub-divided into clusters of energies in which resonances are expected, in particular around the thresholds for inelastic collisions into excited states of the molecule. The different clusters have been distributed across MPI tasks. It is also possible to input targeted grids of energies at which an earlier run has located evidence of resonances.

Main Results:

The dataset used is the dissociative e-N2+ scattering calculation that provides four clusters of scattering energies per geometry when run over the required energy range for this problem. The testing was undertaken on an IBM iDataplex machine at the Hartree Centre, UK, which comprises of 512 nodes each with two 8 core 2.6 GHz Intel SandyBridge processors making 8,192 cores in total. The nodes each have 32 GB of memory.

The parallel performance results demonstrate good scaling across the four clusters of energies for our dataset on a single node of the IBM iDataplex. The number of clusters remains the same across all geometries and they can be pre-determined with a short pre-processing step before embarking on a large-scale parallel run. However the number of energies per cluster does vary and this evidently has a significant impact on load-balancing the computations efficiently between the MPI tasks.

Number of MPI Tasks	Elapsed Time for Calculation (1 Geometry with 4 Energy Clusters) (secs)	Speed-up
1	3690.8	1
2	2171.4	1.7
4	1117.0	3.3

Table 39: Speed-up obtained from Stage 2 parallelisation of the scattering energy clusters.

The two developments for parallelisation described here will be combined, in order to extend parallel scaling further, as follow-up work given the relatively limited resources (3.5 PMs) available for this project. We have also identified further parts of the code that may be parallelized.

4.4.5 *PR7*

Code general features

Name	*DL_POLY_4*
Scientific field	Computational Chemistry, Physics, Materials
Short code description	*DL_POLY_4* is a classical molecular dynamics (MD) engine. It is written in modularised Fortran90 with parallelisation based on equi-spatial domain decomposition which implemented by the use of MPI. DL_POLY_4 has no library dependencies and also utilises parallel I/O strategies for writing and reading trajectory data. *DL_POLY* is a general purpose program targeting simulations of the microevolution of model systems with very large number of particles. Due to its generic definition of interactions and the wealth of available algorithmic forms for them *DL_POLY* possesses a great flexibility in acceptance of force-fields of any nature and can be used for simulation of all phases and materials; e.g. metals, ceramics, liquids, gasses, bio-minerals, polymers and bio-molecules; including mixtures of them should interaction parameters be available.
Programming language	Fortran 90
Supported compilers	GNU, Intel, PGI, Cray
Parallel implementation	MPI (initial OpenMP parallezation)
GPU support	CUDA port available but without maintenance
Libraries	None
Building procedure	Makefile (no auto tools)
Web site	http://www.ccp5.ac.uk/DL_POLY_4/
Licence	STFC Daresbury Laboratory – free of cost to academia

Table 40: Code general features for DL_POLY.

Main objectives:

So far when studying bimolecular systems, *DL_POLY* only supported explicit solvent simulations under periodic boundary conditions. The motivation to integrate the AGBNP2 implicit solvation model was not only to enable *DL_POLY_4* with an advanced implicit solvation methodology but also to provide a unique method for calculations of hydration energies. The PRACE-2IP library [29], implementing the AGBNP2 implicit solvent model, was integrated into the *DL_POLY_4* molecular dynamics package targeting to speed up the time to solution and/or reduce the computational cost for protein solvation processes. Generally, implicit solvent models lighten the computational loads by reducing the degrees of freedom of the model, removing those of the solvent and thus only concentrating on the protein dynamics that is facilitated by the absence of friction with solvent molecules. Furthermore, periodic boundary conditions are no longer formally required, since long-range electrostatic calculations cannot be applied to systems with variable dielectric permittivity. The AGBNP2 implicit solvation model improves the conformational sampling of the protein dynamics by including the influence of solvent accessible surface and water-protein hydrogen bonding effects as interactive force corrections to the atoms of protein surface. This requires the development of suitable bookkeeping data structures, in accordance with the domain decomposition framework of *DL_POLY*, with dynamically adjustable inter-connectivity to describe the protein surface. The work also required the use of advanced b-tree search libraries as part of the AGBNP library, in order to reduce the memory and compute requirements, and the automatic derivation of the van der Waals radii of atoms from the self-interaction potentials.

Accomplished work:

The previously developed Fortran90 code of the AGBNP2 library was incorporated in a local version of the *DL_POLY_4.05.1* source code. The integration complied with most of the data structures already available in the code. It was designed to stand alone as complementary and independently as possible. Configuration related input data, such as number of atoms, atomic positions, atom names and major Verlet neighbour listings, are imported from *DL_POLY*'s *config_module*. Interaction parameters had to be created on parsing short-range interaction parameters in order to define specific epsilon and sigma in a Lennard-Jones casting for all possible kinds of short-range interaction. These were incorporated in the *vdw_module* and then used as input from there. Working precision and some relevant constants needed for the calculations within the AGBNP2 library are used from the *setup_module*. The *agbnp2_module* contains the AGBNP2 library adapted in a manner commensurate with the MPI framework of *DL_POLY_4*'s domain decomposition so that it included its original OpenMP parallelism within each MPI domain task. The work on integrating the AGBNP2 library within *DL_POLY_4.05.1* included the following optimisation and adaptation tasks:

- Arrays referring to neighbour lists (doublets, triplets, quadruplets) were modified to match the look up style in *DL_POLY*. List ends were hardcoded in the 0th element of the list in order to optimise multiply nested *do* loops for look up and add up functions when calculating contributions;
- The *DL_POLY*'s Verlet neighbour list (VNL), which is unordered and single-sided, had to be split into two lists for the AGBNP2 library:
  - a very short range one for neighbours up to 3 Angstroms distance so that the search over possible doublets, triplets and quadruplets is minimised as based on cross-sectioning of spheres with Born radii (< 3 Angstroms) of the species involved in the multiplets; and
  - a complementary list for neighbours from 3 Angstroms to the full range of the non-bonded cut-off. This list is needed for applying corrections to the GB energy calculations.

- The split of this list involved changes to the original AGBNP2 library in terms of loops optimisations as otherwise the library would not have worked at all for architectures with small memory per core allowance and as we found worked very slowly due to the excessive search over the long VNL as supplied originally by *DL_POLY*.
- *DL_POLY* was enabled to provide the Lennard-Jones's characteristic length (sigma) and energy (epsilon) values as required by the OPLSAA force-field for calculations. As *DL_POLY* has no force-filed of its own a number of routines and modules had to be adapted so that this was possible for all possible short-range potentials forms (about 10 different potentials, available in *DL_POLY*), including numerical search for potentials supplied in a tabulated form.
- All force pre-calculations in the library had to be modified to include interactions of halo atoms with domain atoms according to the domain decomposition of *DL_POLY*. This involved selective extension of do loops over ranges of domain and halo.
- All full force calculations and their contribution had to be carefully filtered so that only qualifying atoms on the domain had the application and correction forces and energies added despite that domain atoms may interact with halo ones. This was necessary to ensure that energy contributions are not miscounted (the potential energy does not drift) and no total force is generated in the system (the kinetic energy does not lead the system to overheating).

For the purposes of demonstrating performance and scalability, a model system from our own research (the antimicrobial peptide magainine) was enlarged to a size of 46336 atoms and scaled up by a factor of two, to 92672 atoms, and by a factor of four, to 185344 atoms, using the NFOLD system enlarging routines of *DL_POLY_4*.

Main results:

The performance speed-up as function of number of threads for the system consisting of 46336 atoms is shown in Figure 24. As one can see, the performance scalability is close to ideal. We expect that this performance could be sustained up to about 1024 threads before the systems size to MPI tasks (domains) ratio puts *DL_POLY_4* in unfavorable regimes of performance with respect to the cut-off employed. It is worth noting that system sizes are small due to the absence of discrete water molecules and that plays a limiting factor on the size of the MPI tasks before time spent in communication prevails over compute time.

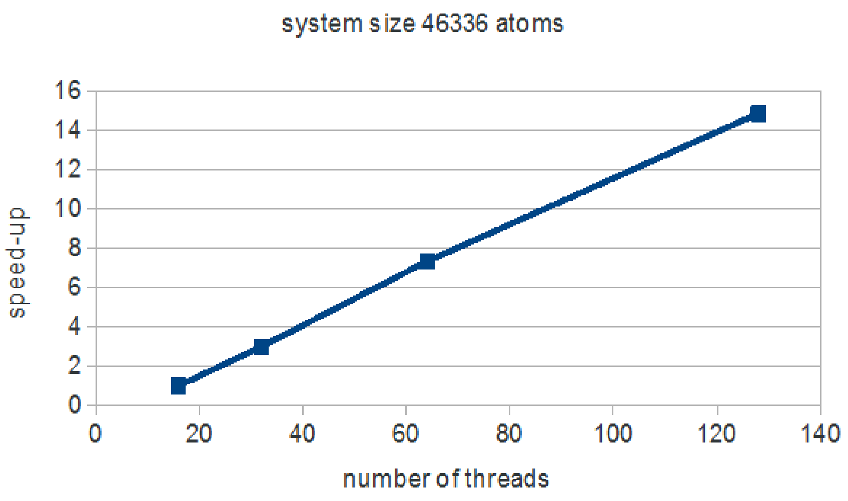


Figure 24: Speed-up upon number of threads for the system of 46336 atoms. The hybridized parallelization involved a load of 16 OpenMP threads per MPI task.

Figure 25 presents the performance scaling of the implementation within DL_POLY_4 by

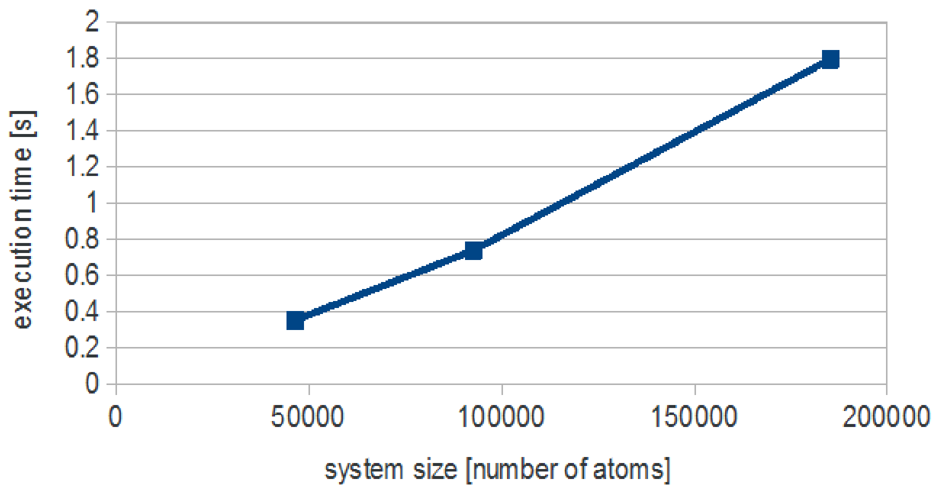


Figure 25: Execution time on 8 MPI tasks with 16 OpenMP threads each versus system size.

system size for a fixed compute resource. It plots the average execution time per time-step on 8 nodes (128 threads) for the three systems of sizes 46336, 92672 and 185344 atoms. It is worth mentioning that in this specific version of *DL_POLY_4* there is no OpenMP parallelism outside the AGBNP2 model library. Thus, the most compute intensive task within *DL_POLY_4*, the creation of the Verlet neighbour-list structures, is not OpenMP parallelized and hence the executions times are larger than those expected for this system sizes. It is the domination of this task that in fact leads to the super scaling observed in the figure. It is solely due the decreasing compute cost of the linked-cells construction pre-factor with respect to the cost for building up the Verlet neighbour list as the system size increases.

4.4.6 *PR8*

Code general features

Name	Socio-economic challenge on enabling bioinformatics workflows
Scientific field	Marine Genomics
Short code description	The tools perform different bioinformatics tasks on sequences and reference genomes such as indexing, alignment, trimming, quality checking.
Programming language	Bash, Perl, Python, Java, C
Supported compilers	Mostly script languages, various compilers for bioinformatics tools
Parallel implementation	Threading, simultaneous execution of workflow instances and jobs
Accelerator support	-
Libraries	None for the workflow execution, various for the bioinformatics tools
Building procedure	Ant, configure/make for bioinformatics tools
Web site	-

Licence	-

Table 41: Code general features for bioinformatics workflow.

Main objectives:

The main objective was to enable biological workflows on large HPC systems.

Accomplished work:

We implemented two example workflows on the PRACE Tier-0 machine CURIE by writing job scripts for individual steps of a workflow. Each job is based on a template that integrates the actual workload and monitoring to capture various performance metrics. About a thousand jobs were ran and analyzed, which led to several improvements to lower the overall resource consumption (mainly disk space related). We also implemented the use cases on an easy-to-use portal, the Lifeportal (UiO, Norway), which is linked to the Tier-1 cluster ABEL (UiO, Norway).

Main results:

We successfully enabled example workflows to run fully automated on the PRACE Tier-0 system CURIE. With the given resource limits on the system (300 concurrent jobs) we could demonstrate a "speed-up" of about 10-100. The reference value for the speed-up is the number of concurrent jobs a scientist is able to manage in a traditional non-HPC setup (estimated at 10), i.e., where biological workflows are usually executed on a single large server. This reference value is usually not limited by the performance of a machine, but rather by the tedious, manual coordination work. Higher speed-ups are easily possible by lifting the resource consumption limits. Our approach to enable the use cases was unique in the sense, that we did not modify any of the application codes. In case we observed performance issues with an application, we contacted the developers to let them improve the code themselves. Most applications we used are single-threaded and ran quite short. The tools for compute intensive tasks, especially alignment, support multi-threading. We ran several tests with the maximum available number of processors in a single node, 128 on the fat nodes. Figure 26 shows the number of active threads and their aggregate memory consumption for an alignment job on CURIE. The graphs show the monitoring data of the first three hours of a job running three days in total. The main reason for using these high numbers of cores was to limit the runtime of a job below the runtime limit of three days. Since, the jobs did only require a fraction of the main memory of those nodes (32 GB of 512 GB, cf. second vertical axis in Figure 26), it would be a much better use of the resources to lift the runtime limit, and to use one of the thin nodes of CURIE (providing 64 GB of memory).

In order to reuse our results, we recommend a systematic procedure to scale the execution of a large number of workflow instances to a large HPC system.

- A description of the workflow is needed plus a comprehensive list of all needed software packages. This information should reveal possibilities for scaling at the level of individual steps and among independent steps of a workflow.
- Scalability goals need to be defined: how many workflows shall be executed (in total, simultaneously), what is the range for the size of input data (single workflow instance, all workflow instances).
- Information about the possible target systems' configuration and environment, and their resource management system as well as limits for resource consumption is needed.
- The software packages need to be tested on the target system to verify they function correctly and, more importantly, to determine their resource consumption. If a tool consumes too much resources alternatives may be considered or the developers of

them can improve them. Usually, the vast number of aspects to be considered in the scaling will limit ability of users to tweak specific tools themselves.

- Selecting or implementing a framework for executing a single workflow instance and orchestrating multiple workflow instances as well as the logistics for transferring data to and from the HPC system. The decision for a framework may limit the available target systems, and vice-versa.

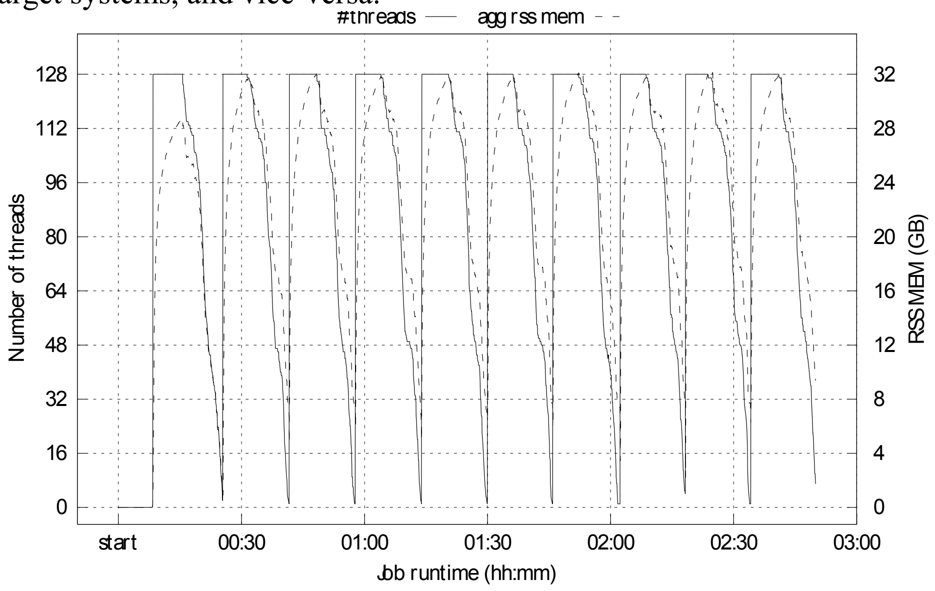


Figure 26: Monitoring of thread activity and memory consumption for an alignment job on CURIE.

- Incrementally implement and run a single workflow. Then, run multiple workflows to observe if any resource consumption limits (number of jobs, runtime, file system quota) are hit. Improve the workflow or the orchestration of several workflows if necessary to achieve the defined scalability goals.

All the details about the use cases being studied, the implementation challenges, the implementation itself, the findings and lessons learned are documented in white paper *"Scaling Biological Data Workflows to Large HPC Systems – A Case Study in Marine Genomics"* [17].

4.4.7 *PR9*

Code general features

Name	*OpenFOAM*
Scientific field	Computational fluid dynamics, solid dynamics, electromagnetics
Short code description	*OpenFOAM* (Open Field Operation and Manipulation) is a general purpose finite-volume simulation framework developed by OpenCFD Ltd at ESI Group and distributed by the *OpenFOAM* Foundation. It has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics. Instead of being a monolithic system *OpenFOAM* is designed as a highly customizable library where users can implement their own solution sequences using pre-defined program blocks.
Programming	C++

language	
Supported compilers	GNU (version 4.5.0), Intel (version 14.0.1), LLVM Clang (version 3.4)
Parallel implementation	MPI
Accelerator support	None
Libraries	SCOTCH
Building procedure	Wmake
Web site	http://www.openfoam.org
Licence	GNU general public licence (GPL)

Table 42: Code general features for OpenFOAM.

Task 1: Implementation and performance improvement of fluid-structure interaction solver

To evaluate and optimize new aircraft designs, e.g. with respect to fuel-efficiency and noise-reduction, high-fidelity simulations are needed that cover multiple disciplines and accurately model the interactions between them. The goal of this task is to use *OpenFOAM* for simulating fluid-structure interactions (FSI) and analyse its scaling characteristics.

Accomplished work:

On the basis of *OpenFOAM* a partitioned, strongly coupled solver for transient fluid-structure simulations with independent meshes for the fluid and solid domains has been implemented. For several variants of solvers overall scalability analyses have been performed using a solid 3D beam. The beam has a quadratic cross section, is fixed at one end and surrounded by fluid (see Figure 27). For the work described in this section we wanted to focus as much as possible on computation and communication performance and therefore we defined the test cases in such a way that I/O operations are minimized (i.e. only write final result data).

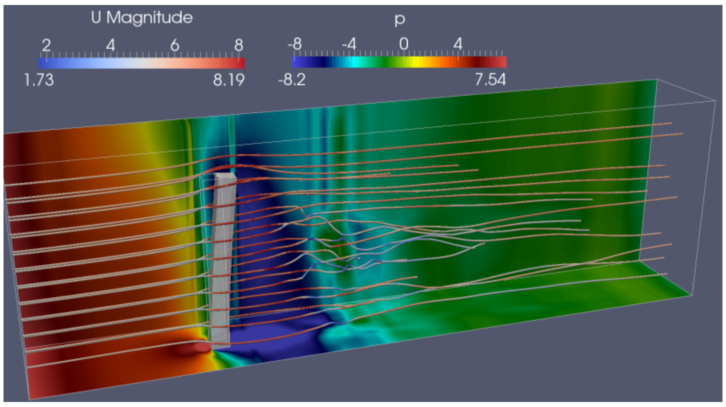


Figure 27: Solid beam consisting of homogeneous elastic material. The ratio between the number of cells in the solid and fluid mesh is approximately 1:50.

Profiling with *HPCToolkit* showed that the computation of mesh deformations of the fluid domain, as currently implemented in *OpenFOAM*, is a major limiting factor. By changing the implementation of *OpenFOAM*'s inter-process communication class *Pstream* from point-to-point MPI communication routines to collective MPI communication routines its strong scaling behaviour could be enhanced considerably (a 61 million cells example shows good scaling at least up to 4096 processes). The changes will be reported to the developers of the official *OpenFOAM* release in order to discuss how the implementation of inter-process communication can be improved in future releases.

Main results:

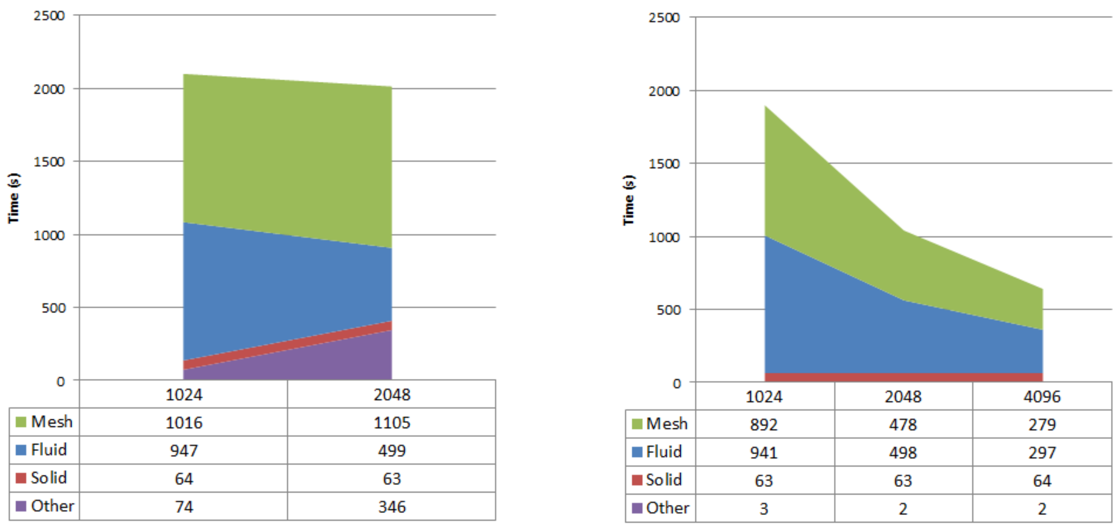
We implemented an FSI solver *icoFsiFoam*, which – in terms of *OpenFOAM* – defines two computational regions, one for the fluid domain and one for the solid domain. Each domain has its own independent mesh, where the fluid mesh is defined as a dynamic mesh allowing transient mesh deformations. The time step loop contains an inner loop for the strong-coupling formulation which has the following structure:

- Set interface displacement of fluid mesh (result of CSM computation). This step requires interpolation of displacement data from the solid domain to the fluid domain.
- Compute mesh deformation of fluid mesh using one of *OpenFOAM*'s mesh motion solvers.
- Solve fluid fields using a pressure implicit scheme with splitting of operators (PISO) scheme.
- Set interface pressure respectively forces for solid mesh (result of CFD computation). This step requires interpolation of pressure data from the fluid domain to the solid domain.
- Solve solid fields using a homogenous linear elastic material model.

One of our key findings is that performance and scalability of *icoFsiFoam* highly depend on the selected method for updating the fluid mesh. Mesh update is done by solving cell-centred Laplacian for the motion displacement (*displacementLaplacianFvMotionSolver* class). From the different variants that *OpenFOAM* offers, we used the uniform diffusivity model (as an example of a computationally simple method) and the quadratic inverse distance diffusivity model (as an example of a computationally complex method).

The interpolation and data transfer between meshes in *icoFsiFoam* either relies on patch to patch interpolation or on an arbitrary mesh interface (AMI). Patch to patch interpolation requires that all cells adjacent to the coupling interface are assigned to one process, whereas AMI allows distributing the coupling interface for both regions among multiple MPI processes, i.e. using an m to n communication pattern. For investigating strong scaling behaviour we used a mesh of the 3D beam case with 61 million cells and run 5 time steps.

For AMI and uniform diffusivity Figure 28 compares the scalability behaviour of the different FSI operations when using *OpenFOAM*'s original *Pstream* class or a modified variant which will be discussed in detail below. Using the original *Pstream* class, except for the CFD (Fluid) computation, we observe no scaling from 1024 to 2048 processes. For 4096 processes we do not have measurements available because the configured wall clock limit of 90 minutes was reached before the first of the 5 simulation time steps had finished. In contrast to that, on the right side of Figure 28 we see a good overall scalability up to 4096 processes when using our modified version of the *Pstream* class.



	1024	2048
🟩 Mesh	1016	1105
🟦 Fluid	947	499
🟥 Solid	64	63
🟪 Other	74	346

	1024	2048	4096
🟩 Mesh	892	478	279
🟦 Fluid	941	498	297
🟥 Solid	63	63	64
🟪 Other	3	2	2

Figure 28: Total runtime for FSI operations for uniform diffusivity. Scalability is considerably improved when using a modified version of OpenFOAM's Pstream library (right) instead of the original one (left).

Within *OpenFOAM*, inter-process communication is implemented in such a way that every MPI process writes output data into dedicated binary buffers (*PsteamBuffers* class). The actual data transfer is delayed to a collective synchronization point, when all MPI processes call the method *PstreamBuffers::finishedSends()*. After that, every MPI process reads back the input data from binary buffers.

The method *PstreamBuffers::finishedSends()* delegates the data transfer to the method *Pstream::exchange()*. Surprisingly, the hotspot within *Pstream::exchange()* is not the transfer of the actual messages, but the transfer of the message sizes, which is done by the function *combineReduce()*. Two things are worth mentioning:

- More data than necessary is transferred: *combineReduce()* replicates the complete matrix of message sizes on all processors, although in general only one row of this matrix is really relevant to a given processor.
- The implementation is not optimal: *combineReduce()* emulates a collective operation by using MPI point-to-point communication routines rather than using appropriate collective communication MPI routines directly.

In order to address these two observations, we changed the implementation of *Pstream::exchange()* in the following way. For the problem at hand, namely the transfer of message sizes, the most natural approach is to use the collective communication routine *MPI_Alltoall*. We added a wrapper method for *MPI_Alltoall* to the class Upstream and replaced the function call *combineReduce()* in the method *Pstream::exchange()* by a call to the new method *Upstream::alltoall()*. This way every MPI process receives just the data needed.

Task 2: Scaling analyses of the individual sub-computations for full aircraft model

The main objective of this sub task was to analyse scaling capabilities of different *OpenFOAM* solvers that are used as building blocks for computing fluid-structure interactions (FSI). Analysis was done using a full aircraft model and original *OpenFOAM* sources, i.e. no modifications as described in the previous section were done. Scalability on full model is necessary for identification of bottlenecks of FSI simulation since mesh generation or re-meshing during simulation is an important part of whole simulation and should not be omitted.

Accomplished work:

Scalability tests for fluid and structural solvers were performed for several models with different mesh sizes. CFD tests with fixed meshes (*icoFoam*) and moving ones (*pimpleDMyFoam*) were performed on models consisting of 105, 235 and 845 million of cells respectively 185 million of cells. The structural solver (*solidDisplacementFoam*) was tested on a model with 5 million of cells. All tests were performed on system Anselm at VSB TU Ostrava, Czech Republic and PRACE Tier-0 system CURIE.

Main results:

Analysis of obtained results shows that a main bottleneck for CFD simulations is in mesh generation with *snappyHexMesh* (Figure 29) which will affect performance of a coupled simulation whenever re-meshing during simulation due to large mesh distortion is necessary.

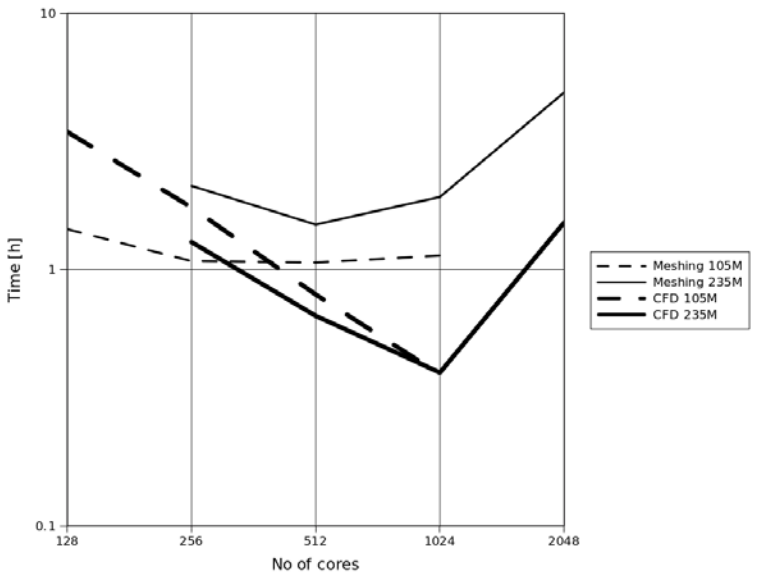


Figure 29: Scalability of mesh generation and CFD simulation.

Another conclusion made from performed scalability tests is that also the CFD simulation with a fixed mesh is not scaling that good for the selected example when more than 1024 cores are used. Although one of our main objectives was to use as many cores as possible it is clear that using more than 1024 cores is not very economical in terms of computational resources consumed.

To test the hypothesis that bad scalability of CFD with 235 million cells is due to the small number of cells per core we created a mesh consisting of 845 million of cells using 1024 and 2048 cores. Scalability tests showed bad scalability and the time needed to run 1000 time steps on 2048 cores was about 20% longer than the time on 1024 cores.

4.4.8 *PR11*

Code general features

Name	Petascaling Machine Learning Applications with *MR-MPI*
Scientific field	Computer Science, Numerical Algebra
Short code description	The codes are developed using *MR-MPI*. MR-MPI is a library that enables efficient utilization of MapReduce paradigm for scientific computing community. The developed codes contain three widely used algorithms in machine learning applications. Namely, these are all-pairs similarity search, all-pairs shortest path and decision

	trees. Contrary to the conventional MapReduce paradigm in which the underlying system consists of commodity machines and the nodes in the distributed environment communicate with remote procedure calls, the developed code is tuned to run on a high performance computing environment by using fast MPI primitives. Besides, efficient data structures that enable working on contiguous chunks of data are utilized. This feature is specific to *MR-MPI* and it allows fast processing of data as opposed to other MapReduce implementations tailored for distributed settings.
Programming language	C++
Supported compilers	GNU, Intel, IBM
Parallel implementation	MPI
Accelerator support	None
Libraries	MR-MPI
Building procedure	Makefile
Web site	http://mapreduce.sandia.gov/
Licence	GPL

Table 43: Code general features for MR-MPI library.

Main objectives:

- Assess and show applicability and usability of the MapReduce paradigm on high performance systems for basic and widely used machine learning algorithms and applications.
- Implementation of machine learning applications in parallel using MapReduce paradigm and message passing interface for communication. For this purpose, use and exploit an efficient tool that is specifically tailored for that purpose.
- Investigation and implementation of optimization techniques that are commonly adopted in scientific computing community to the selected machine learning applications.
- Scale the targeted machine learning applications to a few thousands of cores.

Accomplished work:

- Implementation of all-pairs shortest path (APSP) algorithm. There are two basic algorithms we have implemented using the Map/Reduce paradigm for solving this problem. The first one is the Repeated Squaring method and the second one is the Floyd-Warshall algorithm which uses a dynamic programming formulation.
- Implementation of the algorithm for solving all-pairs similarity search (APSS) problem. This problem is defined as finding all vector pairs the similarities of which are above a given similarity threshold.
- Implementation of building of the decision trees. The Map/Reduce-based construction of the decision tree from a given input set is performed in multiple Map/Reduce iterations, where in each iteration a single level of the tree is formed.
- Detailed large-scale experiments on the IBM Blue Gene/Q JUQUEEN Tier-0 system and the SGI Altix 8600 VILJE Tier-1 system up to 4K processors. Demonstration of the available potential of the MapReduce paradigm for high performance computing aimed at machine learning applications.

- Dissection of the obtained runtimes to investigate the bottlenecks and figure out the hindrances that prevent scalability. For this reason, the different stages of the implemented algorithms are grouped and traced to measure the time spent in each stage.

Main results:

We present the speedup results on an SGI Altix 8600 machine for building of the decision tree. The used dataset consist of a real-life trace with 500,000 records and 90 attributes. Our test is based on regression trees. The codes are developed from scratch with calls to the *MR-MPI* library. We have experimented from 64 to 2048 processors, doubling the processor count while fixing the dataset (strong scaling). We present the obtained running times in Figure 30.

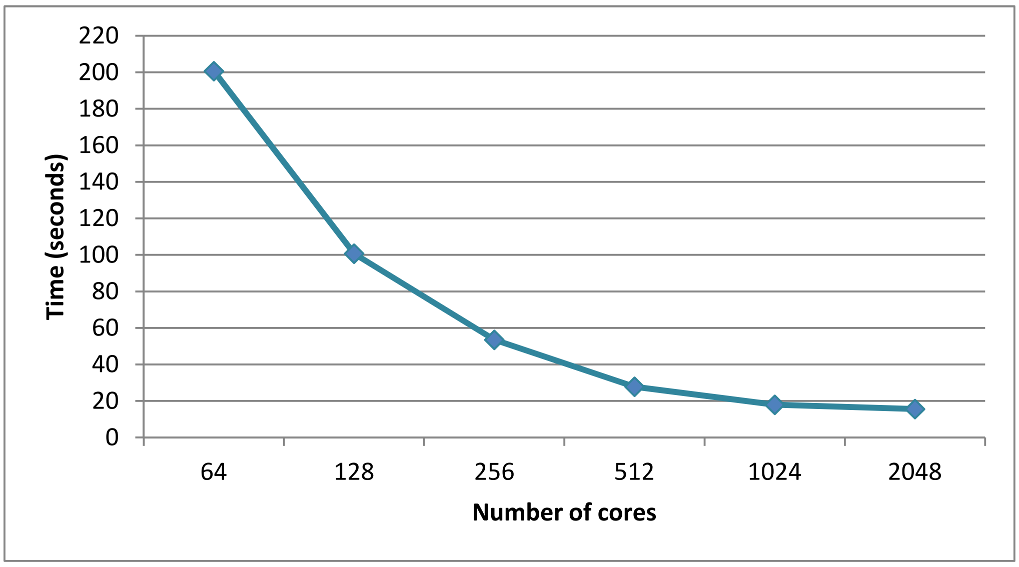


Figure 30: Running time of building decision tree with increasing number of cores.

Up to 1024 cores, we almost get linear speedup. The performance starts to deteriorate at 2048 processors, which is due to the increasing communication requirements. Even though decision tree building algorithm is highly iterative and sequential in nature due to dependencies between successive levels of the tree, the execution times show that a fast and optimized MapReduce library can attain quite good performance.

We also present the runtime dissection of the decision tree building in Figure 31 to reveal the hindrances to achieve a fine scalability performance. With increasing number of processors, the time spent in the collate stage increases while the time spent in the map stage decreases. At 2048 cores, they almost become equal. This indicates that communication requirements of the parallel decision tree building algorithm become bottleneck in obtaining a scalable performance.

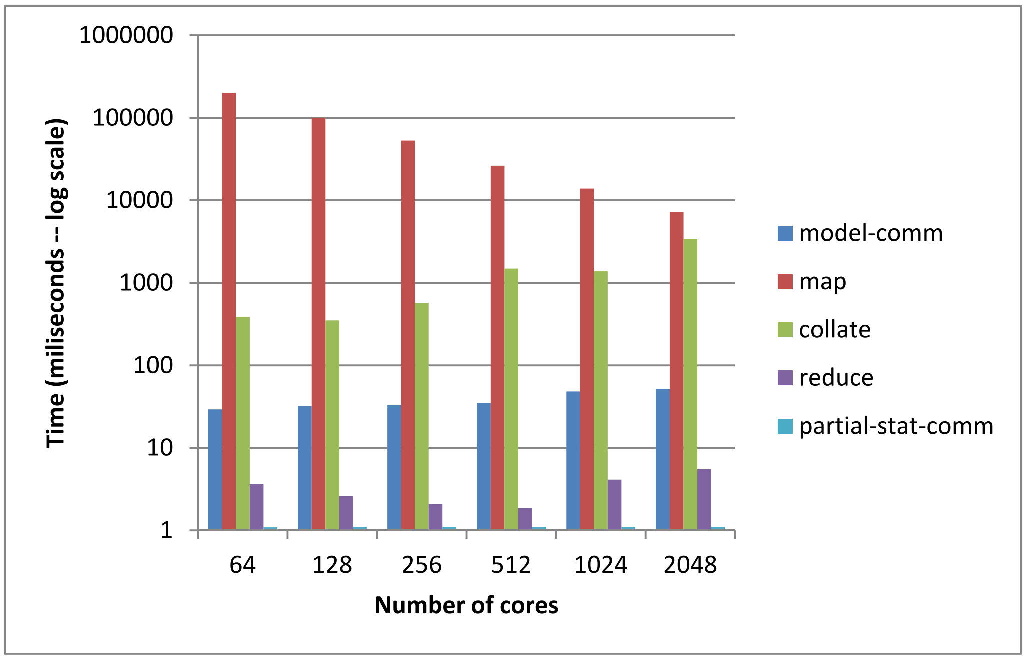


Figure 31: Decision tree build execution time dissection.

4.4.9 *PR12*

Code general features

Name	*Alya*
Scientific field	Multiphysics
Short code description	The *Alya* System, which was developed at the Barcelona Supercomputing Center (BSC), is a Computational Mechanics (CM) code with two main features. Firstly, it is specially designed to run with the highest efficiency standards in large-scale supercomputing facilities. Secondly, it is capable of solving different physics problems, each one with its own modelling characteristics, in a coupled way. These two main features are intimately related, which means that any complex coupled problems solved by *Alya* will still be solved efficiently.
Programming language	Fortran90
Supported compilers	ifort, gfortran, xlf
Parallel implementation	MPI, OpenMP
Accelerator support	None
Libraries	*METIS, HDF5*
Building procedure	Automatic makefile generator
Web site	http://www.bsc.es/computer-applications/alya-system
Licence	BSC proprietary

Table 44: Code general features for Alya.

The main features of *Alya* code are:

- Space discretization is based on unstructured meshes, with several types of elements (hexaedra, tetraedra, prisms, pyramids... linear, quadratic...) implemented.
- Both explicit and implicit time advance schemes are programmed.
- Depending on the case, staggered or monolithic schemes are programmed. However, staggered schemes with coupling iterations are preferred for large multi-physics problems.
- Parallelization is based on mesh partitioning (*METIS*) and MPI tasks, which is specially well-suited for distributed memory machines. On top of that, some heavy weight loops are parallelized using OpenMP threads. Both layers can be used at the same time in a hybrid scheme.
- *Alya* sparse linear algebra solvers are specifically developed, with a tight integration with the overall parallelization scheme. There are no third-parties solver libraries required.
- *Alya* includes some geometrical tools which operate on the meshes for smoothing, domain decomposition or mesh sub-division. In particularly, the latter is a key tool for large-scale simulations.

Main objectives:

- Solve multi-physics problems in a coupled way.
- Run with the highest efficiency standards in large-scale supercomputing facilities.

Accomplished work:

There are different physical modules within *Alya* that are contained in the PRACE benchmark suite. These modules solve incompressible and compressible flows, solid mechanics, chemical reactions and excitable media problems. This Work Package focuses on the incompressible, the solid mechanics and the electrical ones. This section summarizes the numerical models and strategies employed in the incompressible, the solid mechanics and the electrical modules.

Discretization method

The numerical model is a stabilized finite element method for the incompressible module, a standard Galerkin method for large deformation framework for the solid module and a standard Galerkin method for the electrical module (since it only solves a Poisson equation). The stabilization of the incompressible module is based on the Variational MultiScale (VMS) method. The formulation is obtained by splitting the unknowns into grid scale and subgrid scale components. In the present formulation of *Alya*, the subgrid scale is, in addition, tracked in time and in space, thereby giving more accuracy and more stability to the numerical model [30].

Solution strategy

The discretization of the system of equations obtained, either in fluid or solid, computational mechanical yields a coupled algebraic system to be solved at each linearization step within a time loop. Algebraic solvers to solve this coupled system are not robust enough; an iterative strategy should be applied. For incompressible flows the Orthomin method for the Schur complement of the pressure [31] is used, and for solid mechanics a classical Newton-Raphson strategy is used. At each linearization step it is necessary to solve the momentum equation twice and the continuity equation once. The *GMRES*, the *BICGSTAB* method (diagonal and Gauss-Seidel preconditioners) and the Deflated Conjugate Gradient [32] together with a linelet preconditioner are implemented, among others.

Mesh Multiplication

In Peta-scale applications, the pre- and post-process tasks are becoming a bottleneck in the complete simulation cycle. Techniques like parallel I/O have been introduced to mitigate these effects in post-processing, but these are only effective within a limited range. Mesh multiplication (MM) was introduced as an alternative. This technique consists of refining the mesh uniformly, recursively, on-the-fly and in parallel. For tetrahedra, hexahedra and prisms, each level multiplies the number of elements by eight, while a pyramid is divided into ten new elements. This technique is also very useful for studying mesh convergence as well as weak and strong scalability. Figure 32 sketches the recursive MM algorithm. As an example of the efficiency of the algorithm, a mesh of 3 billion elements was obtained in 1 second on 16384 CPUs, starting from a mesh of 3 million elements.

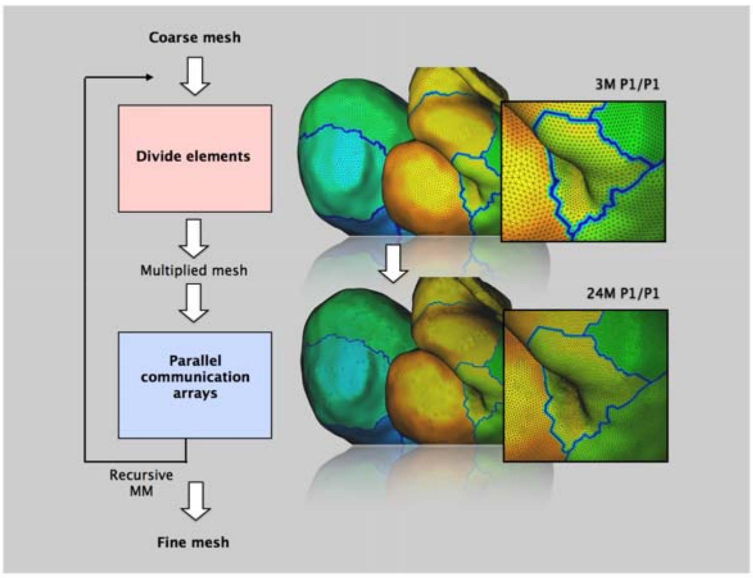


Figure 32: Mesh Multiplication scheme in Alya.

Parallelization

The parallelization is based on a master-slave strategy for distributed memory supercomputers, mainly using MPI as the message-passing library. The master reads the mesh and performs the partition of the mesh into sub-meshes, or sub-domains, using *METIS* (an automatic graph partitioning library). Each process will then be in charge of a sub-domain. These sub-domains are the slaves. The slaves build the local element matrices and the local right-hand sides, and are in charge of solving the resulting system solution in parallel.

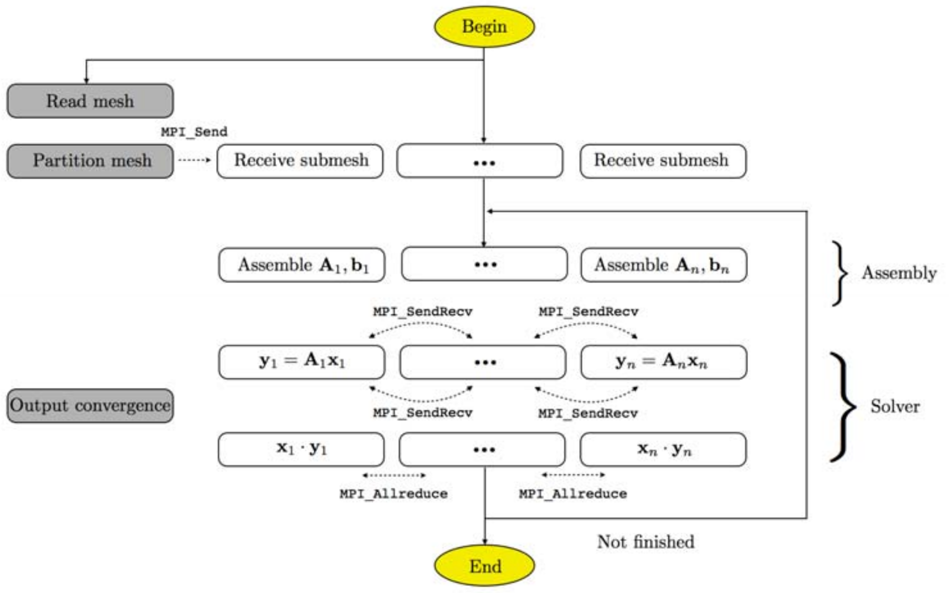
In the assembling tasks, no communication is needed between the slaves, and the scalability depends only on the load balancing. In the iterative solvers, the scalability depends on the size of the interfaces and on the communication scheduling.

As mentioned previously, the momentum and continuity equations are solved with unsymmetric and symmetric iterative solvers respectively. During the execution of the iterative solvers, two main types of communications are required:

- global communications via *MPI_AllReduce*, which are used to compute residual norms and scalar products, and
- point-to-point communications via *MPI_SendRecv*, which are used when sparse matrix-vector products are calculated.

All solvers need both these types of communications, but, when using complex solvers like the Deflated Conjugate Gradients, additional operations may be required, such as the *MPI_AllGather* functions. In the current implementation of *Alya*, the solution obtained in parallel is, up to round-off errors, the same as the sequential one all the way through the computation. This is because the mesh partition is only used for distributing work without in any way altering the actual sequential algorithm. This would not be the case if one considered more complex solvers, like the primal/dual Schur complement solvers, or more complex preconditioners, like linelet or block LU.

Figure 33 is a schematic flowchart for the execution of a simulation using *Alya*. The tasks that the master process is responsible for are shown on the left side of the figure with a grey background. The master process performs the first steps of the execution, namely reading the file and partitioning the mesh. Afterwards, the master sends the corresponding sub-domain information to each slave process; then the master and the slaves enter the time and linearization loops, represented as one single loop.



Begin
Read mesh
MPI_Send
Partition mesh
Receive submesh
•••
Receive submesh
Assemble $\mathbf{A}_1, \mathbf{b}_1$
•••
Assemble $\mathbf{A}_n, \mathbf{b}_n$
Assembly
MPI_SendRecv
MPI_SendRecv
$\mathbf{y}_1 = \mathbf{A}_1\mathbf{x}_1$
•••
$\mathbf{y}_n = \mathbf{A}_n\mathbf{x}_n$
Output convergence
MPI_SendRecv
MPI_SendRecv
Solver
$\mathbf{x}_1 \cdot \mathbf{y}_1$
•••
$\mathbf{x}_n \cdot \mathbf{y}_n$
MPI_Allreduce
MPI_Allreduce
Not finished
End

Figure 33: Master-Slave strategy.

Hybrid parallelization exploits the thread-level parallelism of multi-core architectures, combining MPI tasks with OpenMP threads. The space mesh is partitioned, distributing each partition among the MPI tasks. Then, loops on elements (i.e. assembly loops) and nodes (i.e. solver loops) are threaded using OpenMP. Hybrid parallelization is especially well suited for the current trend of supercomputers, which are large clusters of multi-core processors. The strategy is assessed through transient non-linear solid mechanics problems, both for explicit and implicit schemes, running on thousands of cores.

Scalability

Figure 34 shows the strong scalability and efficiency for the kiln example [33]. The plots show the total scalability, measured summing up the CPU times for all the Physical problems solved, namely low Mach, temperature and chemical reactions. In this example we show the results for two meshes: 528M (called DIV2) and 4.22B elements (called DIV3). For DIV2 (labelled “DIV2 Ref 1K”) the scalability is measured all the way from 1024 up to 100K cores, with the sweet spot around 16K mean elements per core. Beyond that point, efficiency falls below 0.80. For DIV3, i.e. the largest mesh, we run the last three points of the plot, 32768, 65536 and 100000 cores, using as the scalability normalizing value the CPU time obtained for 32768 (labelled “DIV3 Ref 32K”). In order to be fair with the comparison, we have added the scalability and efficiency plots for DIV2, but now normalizing with 32768 instead of 1024 (labelled “DIV2 Ref 32K”). As expected, “DIV2 Ref 32K” is very close to a translation upwards of “DIV2 Ref 1K”. On the other hand, “DIV3 Ref 32K” presents a much better scalability and efficiency, with a sustained large efficiency up to 100.000 cores.

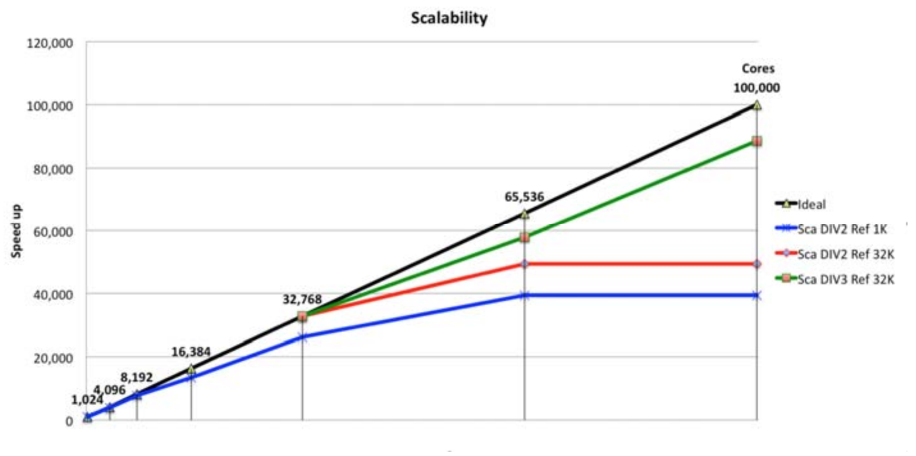


Figure 34: Scalabilty of Alya code.

Coupling

A general coupling strategy for multi-physics problems is implemented in *Alya*. The basic idea is to have independent codes, one for each physical problem, and communicate the coupling variables using MPI. Usually, the multi-physics problems involve different space and time scales, which can lead to situations in which optimised algorithms for the individual problems are useless in the coupled one. Thus, different coupling algorithms and relaxation schemes are considered and tested. In this approach, and in order to have a general scheme, it is of prime importance to exchange the information in the proper section of the code at the right time. Different kind of couplings leads to different kind of information exchange. In some cases the information needs to be exchanged only in a contact surface, while in others it is necessary to exchange information in the whole domain. Also, it has to be taken into account if the information is going to be used as a source term, as a boundary condition or as part of the matrix in the target code.

A possible coupling scheme for a Fluid-Structure Interaction (FSI) problem is shown in Figure 35.

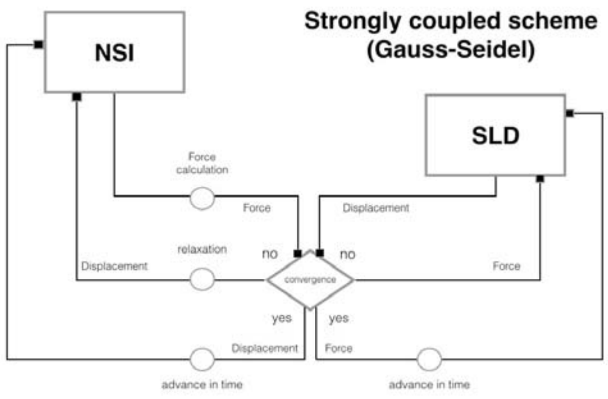


Figure 35: Coupling between incompressible and solid modules in Alya.

4.4.10 *PR14*

Code general features

Name	*EMAC*, *KPP*
Scientific field	Climate science
Short code description	The atmospheric chemistry model *EMAC* combines the global circulation model ECHAM and the *Modular Earth Submodel System* (MESSy). It is used and developed at many European institutions to study a wide range of problems in climate and environmental sciences. Presently, *EMAC* is a classic distributed memory application exclusively relying on the Message Passing Interface (MPI) for parallelisation. It runs on large clusters of nodes with a lot of memory and powerful processors such as IBM POWER and Intel Xeon. Each node processes one or more vertical columns of the three dimensional matrix representing the atmosphere, so that the parallelisation is limited by the horizontal resolution.
Programming language	Fortran, C
Supported compilers	Intel, IBM, Lahey, GCC
Parallel implementation	MPI
Accelerator support	None
Libraries	MPI, NetCDF
Building procedure	Autotools
Web site	http://www.messy-interface.org http://people.cs.vt.edu/~asandu/Software/Kpp/
Licence	MPI-M Software License Agreement, MESSy Software Licence Agreement, GPL

Table 45: Code general features for EMAC model.

Main objectives:
The main goal of this project is to allow *EMAC* to take advantage of recent and forthcoming Peta-scale machines to significantly increase the detail and precision of chemistry-climate simulations. More specifically, the objective is to exploit GPU accelerators on accelerated clusters. A large number of such hybrid systems is already accessible for production runs, e.g., the PRACE Tier-0 system CURIE and the CY-TERA machine at the Cyprus Institute, but also in future this architecture will likely play a leading role in high performance computing, paving the way to Exa-scale computing.

Accomplished work:
We have evaluated different approaches to enable GPU acceleration in the *EMAC* chemistry-climate model. As method of choice we have identified extending the Kinetic PreProcessor (KPP) to generate CUDA code for solving the chemical kinetic equations. The implementation of this most important milestone towards a GPU accelerated *EMAC* is one of the main accomplishments of our project, which is also to the benefit of other applications relying on *KPP*. To allow performance improvements beyond accelerated chemistry, we have additionally ported the *EMAC* model to the PGI compiler which enables an easy acceleration of further sub-models using OpenACC directives.

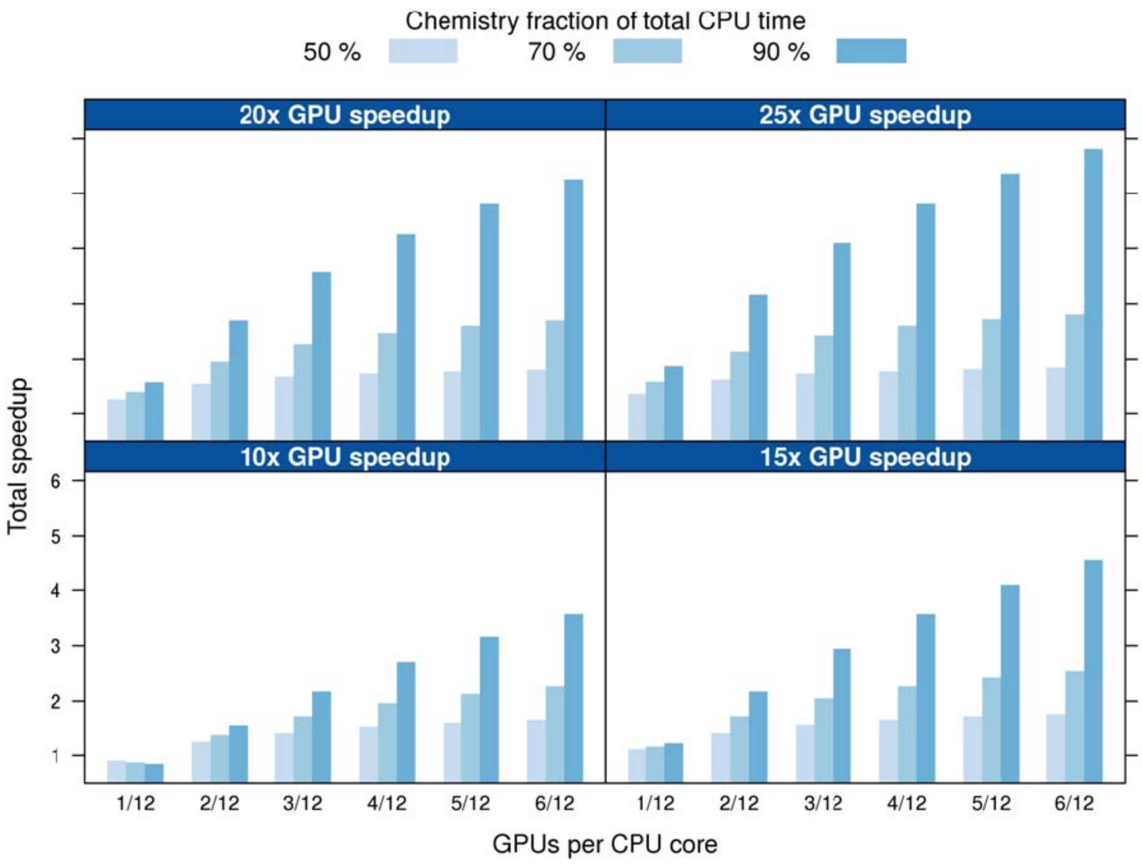


Figure 36: Projected total speedup of EMAC assuming different GPU speedups.

Main results:

Application architecture development

In a typical *EMAC* simulation, computing the atmospheric chemistry processes is one of the computationally most intensive tasks, sometimes requiring up to 90 % of the total computing time. For this reason, our efforts focus on accelerating the chemistry, which promises the largest overall performance gains. Compared to the evaluation on the CPU, offloading the chemical kinetics to GPUs yields large speedup factors. Figure 36 shows the projected total speedup of *EMAC* resulting from typical speedups of GPU accelerated chemical kinetics (GPU performance / CPU core performance = 10, 15, 20 and 25) depending on the fraction of CPU time spent on the chemical kinetic system (50 %, 70 % and 90 %) and the number of GPUs available per CPU core. Further GPU acceleration of other parts of the model might be achieved in future using OpenACC directives.

The components involved in the process illustrated in Figure 37. The system is defined in the domain specific *KPP* language which is translated to Fortran or C code by the Kinetic PreProcessor *KPP*. The code is then optimised and integrated into *MECCA* and *EMAC* by the KPP Post-Processor *KP4*.

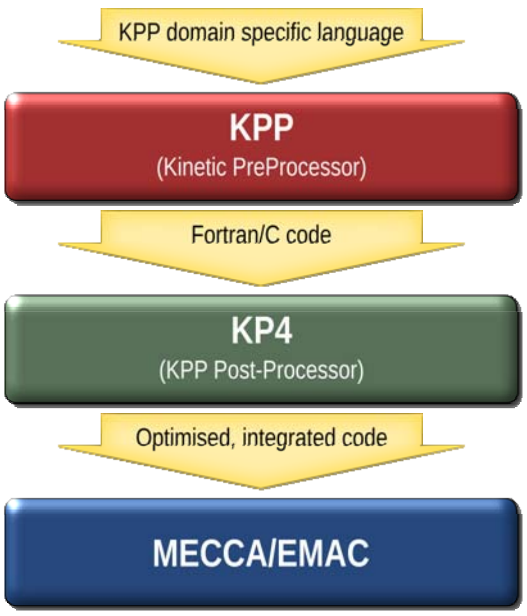


Figure 37: Processing of the chemical kinetic system in MECCA/EMAC.

In this project, both approaches have been pursued and evaluated. Refining *KP4* to introduce CUDA calls into the *KPP* output promises quick success. On the downside, the result of this effort is exclusively to the benefit of *EMAC* and depends heavily on both, the *KPP* and the *EMAC* version. As on top of that the already complex post-processing by *KP4* would be supplemented by a further post-processing layer, our study concludes that a tolerably clean and foresighted implementation of this approach is not feasible. Hence, our focus shifted to modifying *KPP*. This approach is much more ambitious, however the result is not only a cleaner, easier maintainable implementation of GPU acceleration, but moreover it is of more general interest.

GPU enabled KPP

KPP is used in many atmospheric chemistry related applications other than *EMAC*. All these applications could benefit from enabling *KPP* to produce GPU accelerated code.

We have developed a GPU enabled *KPP* replacement based on the open source code of the original *KPP* (version 2.2.3). As GPU programming model we use Nvidia's *Compute Unified Device Architecture* (CUDA). The source code of our *KPP* replacement is available under GPL from a public Bitbucket repository. The basic functionality is implemented and the repository will track further additions and improvements. So far, the development has been focused on profiling the *KPP* generated code and identifying computational bottlenecks which are overcome by offloading to GPUs. The development of future versions will focus on performance optimisations, aiming at offloading as much of the workload as possible to GPUs.

Even though a still young project, our GPU enabled *KPP* version has already sparked the interest of other projects relying on *KPP*. Their inquiries underline the general interest in an open source GPU enabled *KPP* alternative and further justify switching from the post-processor (*KP4*) oriented to the more ambitious *KPP* oriented approach to enabling GPU acceleration in *EMAC*.

OpenACC development enabling

On the long term, when the optimal speedup of the chemical kinetics computation is approached, the overall speedup of *EMAC* is limited by the remaining non-accelerated sub-models. To achieve further speedup, additional computationally expensive sub-models like the aerosol sub-model GMXe could be addressed similarly.

As most convenient way to do so, we have identified the use of OpenACC compiler directives. They are well supported by the PGI compiler from Nvidia, however, so far, this compiler has not been used for *EMAC* development. Introducing a new compiler for compiling EMAC with its more than one million lines of code is highly non-trivial but has been accomplished as part of our project to enable future OpenACC programming. Indeed, several problems were encountered, most of them related to the PGI compiler itself. The main problem is the compiler's naming convention for functions in object files. We have discussed possible solutions with PGI but for downward compatibility the problems will not be resolved on compiler level. However, a workaround is the renaming of the affected subroutines and modules in the source code, which enables future use of OpenACC directives to accelerate *EMAC*.

4.4.11 *PR15*

Code general features

Name	*Delft3D-FLOW* & *SWAN*, coupled with *Delft3D-WAVE*
Scientific field	Environmental sciences
Short code description	*Delft3D* is a world leading 3D modelling suite used to investigate hydrodynamics, sediment transport and morphology and water quality for fluvial, estuarine and coastal environments. As of 1 January 2011, the *Delft3D* flow (FLOW), morphology (MOR) and waves (WAVE) modules are available as open source. *Delft3D* has over 350k lines of code and is developed by Deltares. *Delft3D-FLOW* computes circulation with to the shallow water equations; *SWAN* computes wave heights and frequencies.
Programming language	Fortran 90, C, and C++
Supported compilers	GNU, INTEL, IBM
Parallel implementation	*Delft3D-FLOW* uses MPI, *SWAN* uses either MPI or OpenMP.
GPU support	No
Libraries	MPI, netcdf, expat, pthreads, Autotools, Libtool
Building procedure	configure & make
Web site	Delft3D: http://oss.deltares.nl/web/delft3d, SWAN: http://www.swan.tudelft.nl/
Licence	Delft3D: GPL, SWAN: GPL

Table 46: Code general features for DELF3D and associated coupled codes.

Task 1: Coupled simulation with Delft3D-FLOW & SWAN for informed decision making

The design process of a lake and its environment asks for an interactive approach in which different aspects (economical, engineering, recreational, safety for flooding, ecology) from different stakeholders can be combined. For this purpose, for Lake Marken in the

Netherlands, a multidisciplinary coupled model exists. However, the current runtime for a scenario with the model is four days, so interactive sessions (that combines drawing measures with calculations effects with the model with stakeholders) are not feasible yet.

The two main objectives of the project are:

- Reduction runtime from 4 days to less than 1 hour.
- Interactive design sessions in visualization room with high bandwidth connection to HPC facilities.

Accomplished work:

The applications *Delft3D-FLOW* and *SWAN* are used to simulate respectively water flow and water waves. These two applications have been coupled with *Delft3D-WAVE* and the combination of these 3 executables has been optimized on the Tier-1 Bull cluster CARTESIUS. The runtime could be decreased with a factor 4 with hardly any additional hardware. Over 80% of the total runtime consists of unnecessary I/O operations for the coupling, of which 70% could be removed. Both I/O optimizations and replacement with MPI were used. As a result, we must conclude that it is not possible to use this coupled model as a tool in an interactive design session that cannot last more than 1 working day. However, the interactive design session with a more simplified model, learned us that the simplified model is of value for the design process. We think we can further improve this by making the transition for modelling with the simplified model and the more detailed model more flexible.

Main results:

The initial simulations were run on 1 node: *Delft3D-FLOW* is run with 6 MPI processes, *SWAN* is run with 24 OpenMP threads. Both models run concurrently, not side-by-side. Several optimizations were made and overall results are summarized in Table 47.

Test	**Estimated remaining runtime**	**Remarks**
0. Initial performance	4 days 18 hours	
1. *Delft3D-FLOW* with 12 MPI processes (disc.)	4 days 8 hours	Failure after 880 time steps
2. Use of ramdisk (/dev/shm) for I/O	3 days 2 hours	
3. Use of *FORT_BUFFERED=true*	3 days 0 hours	
4.Use of *KMP_AFFINITY=compact,0,0*	2 days 19 hours	
5. *SWAN* with NetCDF for hotstart data (disc.)	2 days 16 hours	
6. *SWAN* with MPI for hotstart data	2 days 3 hours	
7. *Delft3D-FLOW* with 11 MPI processes	2 days 0 hours	
8. *SWAN* with 32 threads on fat node	1 day 21 hours	
9. *WAVE-FLOW* coupling through MPI	1 day 6 hours	

Table 47: Successive results of optimization strategies.

Over 80% of the runtime is spent in unnecessary I/O that is used to couple the different applications. The optimization of the I/O has resulted in more than 69% performance improvement, without increasing the number of cores used for the simulation. Another 5% performance improvement was reached with scaling up the application to use as many cores as possible on the one node of Tier-1 system CARTESIUS. It is expected that the runtime can be reduced to about 12 hours with further removal and optimization of the I/O. *SWAN* can scale to about 100 MPI processes for this particular case, but it was not yet ready to be used in this coupled simulation. Refactoring of the code was the introduction of a few well-placed MPI calls, to show the optimization potential.

Task 2: Porting Delft3D/Flow on FERMI Blue Gene/Q system

Port the *Delft3D* code on massively parallel architectures such as the Blue Gene/Q architecture, and test it, in order to have its scalability and found its performance issue. This work will be preparatory for an extensive work aimed to overcome the parallel bottlenecks of the *Delft3D* code.

Accomplished work:

- Some improvement and modification on the code were made. For a detailed list please refer to the related white paper [22]. Some of the more important changes are listed below:
  - Conversion of all auxiliary files (e.g. headers or include files) from DOS format to UNIX.
  - Installation of the last version of autotools currently available (Libtool 2.4.2 Autoconf 2.69, Automake 1.13.2) on FERMI system in order to avoid all possible compatibility problems.
  - Installation of the EXPAT library to enable some I/O procedures of the code (in particular those used for reading XML files).
  - The code was compiled and tested using both IBM XL and GNU compilers. To ensure a sufficient compatibility with some platform specific options it was necessary to write a more complex configure procedure.
  - Creation of a Static linking procedure in order to overcome the limitation due to the incompatibility of Blue Gene/Q compilers and dynamic linking. This change was particularly complicated because the use of libtool creates a series of bugs in linking that we eventually resolved using the *"--all-static*" flag (only during the linking procedure).
  - Declaration of some new environment variables in order to prevent crashing of the compilation or execution procedure (e.g. *ac_cv_func_malloc_0_nonnull=yes* ; *ac_cv_func_realloc_0_nonnull=yes* ;).
  - Remove all the C-style comments in the *config.h* files included in pre-processable FORTRAN files, since they are not recognized by IBM FORTRAN compilers.
  - Separate compilation for the version_number.exe utility. Indeed, although it is necessary to compile the code for the execution on the back-end nodes (the nodes on which the code will run), the version_number.exe executable must be compiled using the front-end tool chain.
  - The *common.am* file was modified removing all reference to *DelftOnLine*.
  - Some modifications were necessary also on the configure.ac file in order to resolve a compilation problem related to pre-processing.
  - A new *rdtsc* function was created to provide some architecture-dependent information.

  - Some files (*iniid.f90, rdgrid.f90, strgrd.F90 inicut.f90* and *inigrd.f90*) was modified to allow the creation of more than 1000 temporary files that are necessary to increase the maximum number of useable cores up to 10,000.
  - The *–DPTR8* flag was added to the compilation procedure to enforce the use of 8 byte file pointers. This flag solves a problem with the NEFIS library that on Blue Gene/Q systems creates files larger than 4 exabytes (presumably due to corrupt i-node data).
- In order to port the code on FERMI Blue Gene/Q the code was first ported and tested on two Tier-1 systems: the PLX and the EURORA hybrid clusters.
- The code was tested on FERMI using a small size problem of sediment transport. The complete results of this benchmark phase were reported in the related white paper.

Main results:

After a significant porting effort, the *Delft3D-FLOW* application has been benchmarked on the IBM Blue Gene/Q Tier-0 system FERMI - to our knowledge the first time that this has been achieved. Thanks to our benchmarks we found that some of the major bottlenecks of the flow module of the *Delft3D* code are related to the sediment transport routines as we can see from Figure 38.

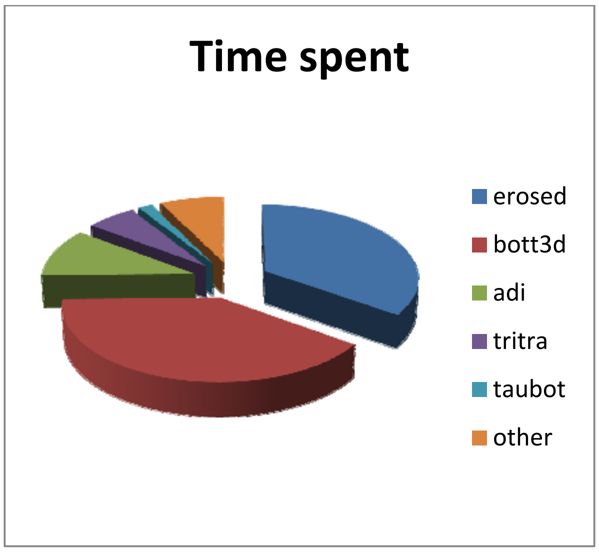


Figure 38: Time spent by Delft3D subroutines in the sediment transport model as obtained by Scalasca.

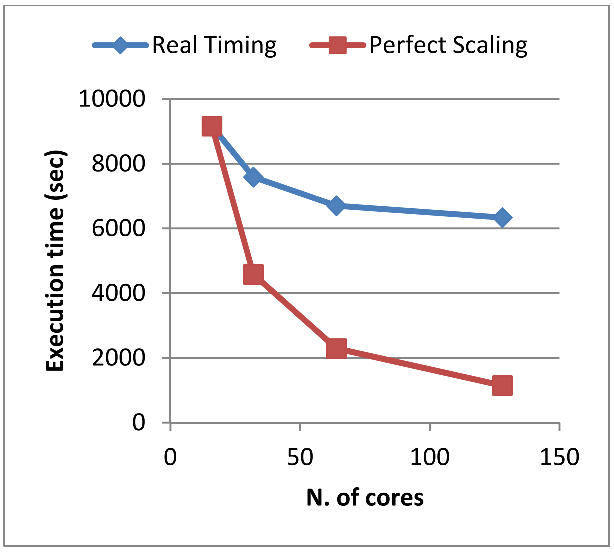


Figure 39: Performance of Sediment transport benchmark on FERMI. We plot the Execution time vs the number of cores.

In Figure 39 we plot the total execution time for a 2000 time step integration. It can be seen that the benchmark does not scale well, even if the execution time decreases up to 128 cores. The overhead for performing sediment transport is significant and sensitive to the number of processes, since the solution of the ADI-solver depends on the process count which cascades into changes in sediment transport and resulting height changes that feed back to the circulation, even for short simulations like these. This affects the reliability of the results, although we do not know to what extent.

Given the current limited parallel scalability of the code with MPI this porting is only really relevant for models which are large enough to utilize at least 1024 cores but with further improvements in the OpenMP parallelisation it may be possible to extend the utility of the application to smaller datasets.

5 Summary

Three parallel sub-tasks on application enabling in Work Package 7 of PRACE-3IP have been described including final reports on the supported applications. These three activities have been organized into support projects formed on a basis of either scaling and optimisation support for Preparatory Access, application support for DECI or for applications addressing major socio-economic challenges.

5.1 Preparatory Access Type C

Task 7.1.A successfully performed four cut-offs for preparatory access including the associated review process and the support for the approved projects.

In total 17 Type C Preparatory Access projects have been supported by T7.1.A. The timeline of these projects is shown in the Gantt-chart in Figure 40. The chart shows the run time of each project with start date and end date in PRACE-3IP. PRACE-3IP took over responsibility for running PA C projects from PRACE-2IP on August 31, 2013. Projects which started in the frame of PRACE 3IP but will finalize in the extension phase of 3IP are not reported in the current deliverable but will be covered by deliverable D7.1.3 which is due in the beginning of next year.

Usually the projects run for six months. Some projects received a prolongation; this is possible on special request, mainly in case of technical system problems. The reason for the slightly different start dates within one cut-off is that each hosting member finally decides on the exact start date of the projects at their local site. Also the PI can choose a slightly later start date.

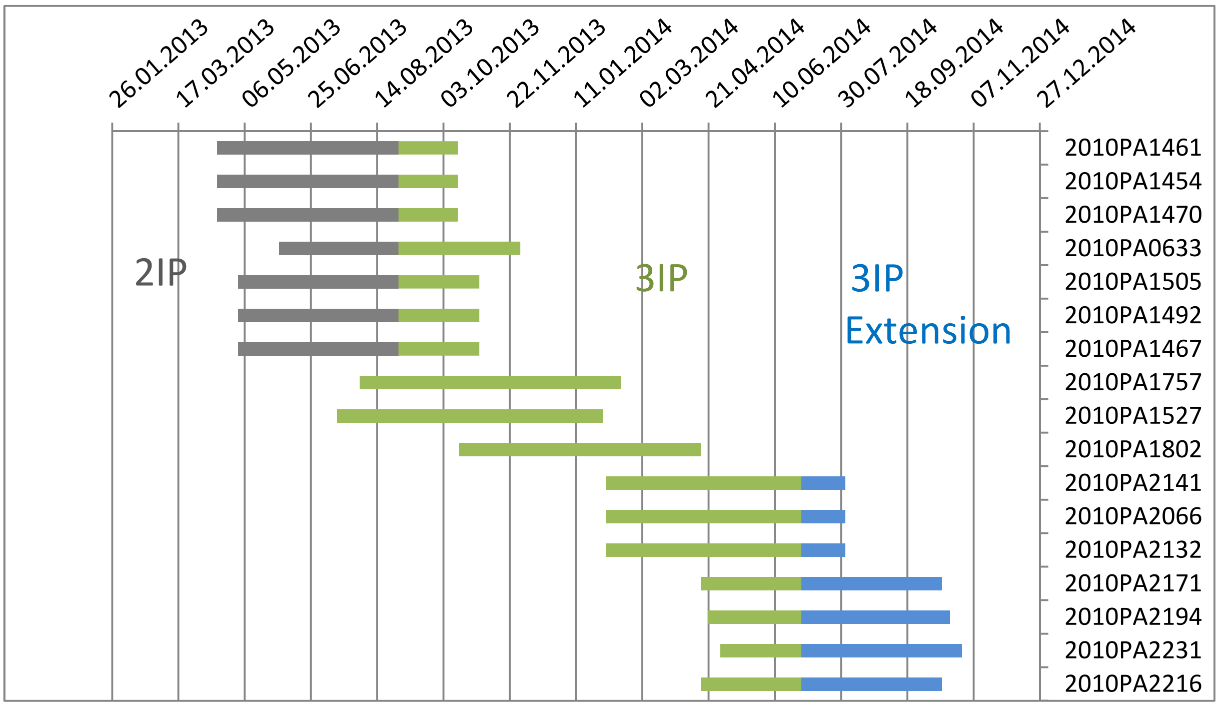


Figure 40: Timeline of the PA C projects.

For the finalized PA C projects ten white papers were created and published. The detailed optimization work has been described in section 2.8, section 2.9 and section 2.10.

The main goal of T7.1.A is to enable codes for the provided Tier-0 systems. Table 48 reflects the success of T7.1.A in this regard. Two out of ten projects already successfully applied for the PRACE Regular call and therefore continue their work on Tier-0 systems. In addition, half of the finalized projects plan to carry on their work on Tier-0 systems and corresponding applications are currently in progress. For the remaining three projects no further information regarding future plans are available yet.

The Preparatory Access to Tier-0 services and especially the support by PRACE experts are invaluable for the scientists to use and effectively exploit the resources of PRACE.

Project number	Title	PRACE Tier-0 regular access
2010PA1461	Enabling Xnavis (URANS solver for fluid-dynamics) for massively parallel simulations of wind farms.	Application in preparation
2010PA1454	Scalability analysis, OpenMP hybridization and I/O optimization of a code for Direct Numerical Simulation of a real wing	N/A
2010PA1470	PA1470 Next generation pan-European coupled Climate-Ocean Model - Phase 1 (ECOM-I)	Application in preparation
2010PA0633	Increasing the QUANTUM ESPRESSO capabilities II: towards the TDDFT simulation of metallic nanoparticles	Awarded regular project access (6th call)
2010PA1505	Scalability of gyrofluid components within a multi-scale framework	Application in preparation
2010PA1492	Direct numerical simulation of a high-Reynolds-number homogeneous shear turbulence	Awarded regular project access (8th call)
2010PA1467	Massively Parallel Multiple Sequence Alignment Method Based on Artificial Bee Colony	Application in preparation
2010PA1757	Optimization of PIERNIK for the multiscale simulations of high-redshift disk galaxies.	N/A
2010PA1527	URANIE	N/A
2010PA1802	Parsek2D-MLMD	Application in preparation

Table 48: Future plans of finalized PA C projects.

5.2 Technical support for DECI

In the 2nd year of PRACE-3IP, T7.1.B continued providing technical support for DECI. The technical evaluations (TEs) were provided for 119 DECI-11 proposals and 61 DECI-12 proposals by the DECI partner sites. All of the180 TEs were completed using the online PRACE PPR tool for DECI.

T7.1.B provided the technical support for all ongoing DECI projects in the 2nd year of PRACE-3IP, including the continued support for 31 DECI-9 projects and 37 DECI-10 projects since August after the PRACE-2IP ended, the full support for 52 DECI-11 projects, and the starting assistance for the 34 accepted DECI-12 projects. The technical support included providing technical assistance for DECI users to get access to their assigned Tier-1 systems, providing closing instructions for their projects, providing solutions whenever any technical issues were raised by the DECI users, and most importantly, providing applications enabling support as required by the DECI projects. There were 4 DECI enabling projects supported during this period, including the DECI-9 project Planck-LFI2, the DECI-10 project DNSTF, the DECI-10 project HYDRAD, and the DECI-11 project Planck-LFI3. A total of 4-12 PMs enabling efforts were contributed by DECI partner sites for the enabling projects listed above. One white paper for the enabling work done on the DECI-10 project DNSTF will be available in end June 2014. Due to the timescale of DECI-12 and PRACE-3IP, no enabling support is available for the DECI-12 projects.

T7.1.B involves 15 partner sites providing 73 PMs in total for the DECI technical support. Monthly telcons and face-to-face sessions were arranged for the subtask progress reporting and discussions. T7.1.B also worked together with WP2 in close collaboration. WP2 was responsible for the DECI process management, while T7.1.B focused on the technical support and applications enabling provided to the DECI projects. The joint videoconference on every 3rd Friday of each month has been started since March 2014, for T7.1.B and WP2 to discuss DECI projects and DECI support in a more efficient way.

5.3 Socio-economic application support

Support for application codes addressing key socio-economic challenges resulted in 11 successful projects. Each project aimed at supporting the given challenge with enabling identified application for efficient Tier-0 HPC systems usage. More than 20 representative application codes received technical support from selected PRACE experts.

Wide range of socio-economic areas has been addressed including BLA. BLA resulting with application codes contributing to solution of these problems. Key results of the enabling work include:

- application suites enabled for a multi-discipline modelling with coupling separate codes together on the selected HPC platforms in cardiac mechanics and environmental studies,
- improved computational kernels for fluid-structure interactions in aeronautics design process and large scale - free of periodic boundary conditions - molecular dynamics,
- scalable implementation of the HPC workflow for genomic data analysis, improved linear-scaling of the widely used DFT code,
- improved runtime of the coupled software application to simulate the different aspects of a lake, ranging from the optical properties of the water to safety concerns allowing interactive decision making,
- Big Data processing approach using HPC implementation of MapReduce programming paradigm.

Applications used for solving problems classified as socio-economically relevant have been enabled for selected PRACE Tier-0 systems including IBM Blue Gene/Q JUQUEEN and FERMI systems, IBM iDataPlex MARENOSTRUM and Bullx CURIE. Nature of the problem studied and associated technical case have been considered for careful target system choice. Each project has addressed appropriate Tier-0 system and also a number of smaller Tier-1 systems have been used for preliminary enabling work.

Technical support and preceding selection process gathered 13 PRACE partners with 80 PMs in total for applications enabling. PRACE SSC members were also involved in selection evaluation phase.

Each of the reported projects has been completed giving clear indications for a future improvements and possible benefits for application community. For a number of projects natural continuation perspectives have been identified being an impulse for a further improvements and innovations for HPC contribution to socio-economic challenges needs.

6 Annex

6.1 DECI Projects

A complete list of DECI projects for each DECI calls in PRACE-3IP is provided below. Lists include the projects in DECI-9, DECI-10, DECI-11 and DECI-12. Technical support for all listed the DECI projects were provided in T7.1.B in PRACE -3IP since August 2013 after PRACE-2IP ended.

6.1.1 *DECI-9 Project List*

PRACE Site (home)	**Project name**	**PRACE Site (exec)**	**Machine(s)**	**Enabling work (PMs)**
BSC	COIMBRALATT	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1[4]
	ICREIMUTANTS	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 1
	SpEcBNS	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
CINECA	DOPE	CINECA	PLX (Intel Westmere EX@2.4)	0 ~ 1
	GPCR4D	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 1
	iMIG	BSC	MinoTauro (XEON E5649@2.53 + GPU (NVIDIA Tesla))	0 ~ 1
CINES	AuPd-Seg	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
	FORSQUALL	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 1
	IONGATE	UIO	Abel (Sandy Bridge@2.6)	0 ~ 1
	NPR-LQCD	CINECA	PLX (Intel Westmere EX@2.4)	0 ~ 1
CSC	NMRCONF	CSCS	Rosa (XT5 CSCS)	0 ~ 1
	CompSym	CSCS	Rosa (XT5 CSCS)	0 ~ 1
	Planck-LFI2	CSC	Sisu (Sandy Bridge@2.6) Louhi XT (XT5 DC@2.3)	1 ~ 3
CSCS	LCRR	EPCC	ICE-Advance (BGQ)	0 ~ 1
		PDC	Lindgren (XE6	

[4] 0~1 PMs indicates that the DECI project requested no technical support or very small amount of efforts for regular technical support only.

			12C@2.1)	
EPCC	LBSCOM	CSC	Louhi XT (XT5 DC@2.3) Sisu (Sandy Bridge@2.6)	0 ~ 1
		EPCC	HeCToR XE6 (XE6 16C@2.3)	
	ESM4OED	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 1
FZJ	TB-Drugs-In_silico	CINECA	PLX (Intel Westmere EX@2.4)	0 ~ 1
	MoMoGal	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 1
		UIO	Abel (Sandy Bridge@2.6)	
ICHEC	Reactive_Ceria	FZJ	JuRoPA (Intel Nehalem@2.93)	0 ~ 0
	Si-Interfaces	CINES	JADE-Harpertown (Intel Harpertown@3)	0 ~ 0
	SPH-WEC	CSCS	Rosa (XT5 CSCS)	0 ~ 1
		NCSA	EA ECNIS (BGP@0.85)	
NCSA	AIMD-PAF	EPCC	ICE-Advance (BGQ)	0 ~ 1
PDC	CoStAFuM	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
		UIO	Abel (Sandy Bridge@2.6)	
	DifVib	EPCC	ICE-Advance (BGQ)	0 ~ 1
		PDC	Lindgren (XE6 12C@2.1)	
	HydFoEn	UHEM	Karadeniz (Intel Nehalem@2.93)	0 ~ 1
RZG	GanDaLF	UIO	Abel (Sandy Bridge@2.6)	0 ~ 1
	PTACRB	ICHEC	Stokes (Intel Westmere EP@2.67) Fionn-thin (Fionn-thin)	0 ~ 1
SURFsara	SPSC	FZJ	JuRoPA (Intel Nehalem@2.93)	0 ~ 1
UHEM	HiSSor	EPCC	ICE-Advance (BGQ)	0 ~ 1
		RZG	Hydra (Sandy Bridge@2.6)	
VSB-TUO	MPI-FETI	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 0

	NPT_MC	UHEM	Karadeniz (Intel Nehalem@2.93)	0 ~ 1
		UIO	Abel (Sandy Bridge@2.6)	

Table 49: DECI-9 project list.

6.1.2 *DECI-10 Project List*

PRACE Site (home)	Project name	PRACE Site (exec)	Machine(s)	Enabling work (PMs)
BSC	AIDMP	FZJ	JuRoPA (Intel Nehalem@2.93)	0 ~ 1
	fplb	WCSS	Supernova (XEON X5650@2.67)	0 ~ 1
	Novel_Anticoagulants	CINECA	PLX (Intel Westmere EX@2.4)	0 ~ 1
	SPAITAC	EPCC	HeCToR XE6 (XE6 16C@2.3) Archer (Sandy Bridge@2.6)	0 ~ 1
CINECA	ERPP	CYFRONET	Zeus BigMem (AMD 6276@2.3)	0 ~ 1
	MOTUS	ICM	Boreasz (P7@3.83)	0 ~ 1
CSC	CONVDYN13	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
	HIV1-GSL	EPCC	HeCToR XE6 (XE6 16C@2.3) Archer (Sandy Bridge@2.6)	0 ~ 1
	HyVaMPI	UIO	Abel (Sandy Bridge@2.6)	0 ~ 1
	NANODROPS	EPCC	HeCToR XE6 (XE6 16C@2.3) Archer (Sandy Bridge@2.6)	0 ~ 1
EPCC	Dissipative_Phenomena	EPCC	Blue Joule (BGQ) Archer (Sandy Bridge@2.6)	0 ~ 1
	GalChem	SURFsara	Cartesius (Cartesius)	0 ~ 1
	Galsim	CSC	Sisu (Sandy Bridge@2.6)	0 ~ 1
	HIGHERFLY	EPCC	HeCToR XE6	0 ~ 1

			(XE6 16C@2.3) Archer (Sandy Bridge@2.6)	
	InterDef	SURFsara	Cartesius (Cartesius)	0 ~ 1
	JOSEFINA	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1
	WISER	EPCC	HeCToR XE6 (XE6 16C@2.3) Archer (Sandy Bridge@2.6)	0 ~ 1
FZJ	INPHARMA	EPCC	HeCToR XE6 (XE6 16C@2.3)	0 ~ 0
	LargeRB2013	EPCC	Blue Joule (BGQ)	0 ~ 1
	MoDSS	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
ICHEC	APOP20X3	PSNC	Cane (Intel Harpertown@2.5)	0 ~ 0
	RODCS	CSCS	Rosa (XT5 CSCS)	0 ~ 1
	waveclim	CSCS	Rosa (XT5 CSCS)	0 ~ 1
ICM	CELESTE	CSCS	Rosa (XT5 CSCS)	0 ~ 1
PDC	DNSTF	EPCC	HeCToR XE6 (XE6 16C@2.3) Archer (Sandy Bridge@2.6)	1 ~ 3
	LipoSim	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1
	MEGAREACT	UIO	Abel (Sandy Bridge@2.6)	0 ~ 1
	PLANETESIM-2	FZJ	JuRoPA (Intel Nehalem@2.93)	0 ~ 1
RZG	HYDRAD	RZG	Hydra (Sandy Bridge@2.6)	1 ~ 3
		VSB-TUO	Anselm (E5-2665-8@2.4)	
	PTACRB-2	ICHEC	Fionn-thin (Fionn-thin)	0 ~ 1
		CYFRONET	Zeus BigMem (AMD 6276@2.3)	
SURFsara	DIVI	SURFsara	Cartesius (Cartesius)	0 ~ 1
		ICHEC	Stokes (Intel Westmere EP@2.67) Fionn-thin (Fionn-thin)	
	SCosPtS	CSC	Sisu (Sandy	0 ~ 1

			Bridge@2.6)	
	TheoMoMuLaM	UIO	Abel (Sandy Bridge@2.6)	0 ~ 1
UHEM	GREENLIGNITE	not assigned	not assigned	0 ~ 1
	WIND-FORECAST	CYFRONET	Zeus (XEON E5645@2.40)	0 ~ 1
VSB-TUO	EXC-XMCD	PSNC	Chimera (Intel Westmere EX@2.67)	0 ~ 1
	TransMem	SURFsara	Cartesius (Cartesius)	0 ~ 1

Table 50: DECI-10 project list.

6.1.3 *DECI-11 Project List*

PRACE Site (home)	Project name	PRACE Site (exec)	Machine(s)	Enabling work (PMs)
BSC	ceriahydro	FZJ	JuRoPA (Intel Nehalem@2.93)	0 ~ 1
	ConfTransHSP90	BSC	MinoTauro (XEON E5649@2.53 + GPU (NVIDIA Tesla))	0 ~ 1
	WHALE	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
CINECA	ATPSYNS	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	HyDiG	NIIF	NIIFI SC (NIIF-NIIFI SC)	0 ~ 1
	SCENE	EPCC	Blue Joule (BGQ)	0 ~ 1
CSC	DyNet	PSNC	Chimera (Intel Westmere EX@2.67)	0 ~ 1
	EERSC	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1
	gklocsoc	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1
	Planck-LFI3	CSC	Sisu (Sandy Bridge@2.6)	1 ~ 3
	Syndecan	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
CSCS	ClChannels	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1
	CROWDING	CSC	Sisu (Sandy Bridge@2.6)	0 ~ 1
EPCC	ASTROGKS	ICHEC	Fionn-thin (Fionn-thin)	0 ~ 1
	BRAFKIN	CSC	Sisu (Sandy Bridge@2.6)	0 ~ 1
	CBCAGE	EPCC	Blue Joule (BGQ)	0 ~ 1

	ECG-MD	EPCC	Blue Joule (BGQ)	0 ~ 1
	EMMA	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	FRAPLAWI	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1
	GGOA	EPCC	Blue Joule (BGQ)	0 ~ 1
	TLRSimSys	CSCS	Rosa (XT5 CSCS)	0 ~ 1
	UltraFOx	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
FZJ	FFF2	CINECA	PLX (Intel Westmere EX@2.4)	0 ~ 0
	GraFI	PDC	Lindgren (XE6 12C@2.1)	0 ~ 0
ICHEC	ELECNANO	CINECA	PLX (Intel Westmere EX@2.4)	0 ~ 1
	IIPDRS	PDC	Lindgren (XE6 12C@2.1)	0 ~ 0
	Photocatalyst	WCSS	Supernova (XEON X5650@2.67)	0 ~ 1
ICM	GPCR-SWITCH	IPB	PARADOX (Sandy-bridge@2.6GHz + Nvidia M2090)	0 ~ 0
IDRIS	CompClay	VSB-TUO	Anselm (E5-2665-8@2.4)	0 ~ 1
	HTMTCC	PSNC	Chimera (Intel Westmere EX@2.67)	0 ~ 1
		UIO	Abel (Sandy Bridge@2.6)	
	Meso-NH-4-DRIHM	CYFRONET	#NAME?	0 ~ 1
	SPOC-MULOR	PSNC	Chimera (Intel Westmere EX@2.67)	0 ~ 1
	UnMAD	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
IPB	CMBE	ICHEC	Fionn-hybrrid (Fionn-hybrid)	0 ~ 1
NIIF	ProPep	CSCS	Rosa (XT5 CSCS)	0 ~ 1
		NIIF	NIIFI SC (NIIF-NIIFI SC)	
PDC	FLOCS	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	GSTP	FZJ	JuRoPA (Intel Nehalem@2.93)	0 ~ 1
PSNC	abinitio-nanocarbon	FZJ	0 (no results) - (no results)	0 ~ 1
	MAPLER	CASTORC	0 (no results) - (no results)	0 ~ 1
RZG	GraSiC-1	RZG	Hydra (Sandy Bridge@2.6)	0 ~ 1

		VSB-TUO	Anselm (E5-2665-8@2.4)	
	NaUSIKAS	SURFsara	Cartesius (Cartesius)	0 ~ 1
SURFsara	FSTRAP	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	LESPVC	SURFsara	Cartesius (Cartesius)	0 ~ 1
	OXYN-LED	CYFRONET	Zeus BigMem (AMD 6276@2.3)	0 ~ 1
	TECHAR	ICM	Boreasz (P7@3.83)	0 ~ 1
	thermospin	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
UIO	N-MILiB	UIO	Abel (Sandy Bridge@2.6)	0 ~ 1
	WeSearch	UIO	0 (no results)	0 ~ 1
VSB-TUO	ELTUNBIO	PSNC	Cane (Intel Harpertown@2.5)	0 ~ 1
	PROS-HIFU	WCSS	Supernova (XEON X5650@2.67)	0 ~ 1
WCSS	Mechanic-Kepler	CYFRONET	Zeus BigMem (AMD 6276@2.3)	0 ~ 1
	PIERNIK-SI	SURFsara	Cartesius (Cartesius)	0 ~ 1
		ICHEC	Fionn-thin (Fionn-thin)	

Table 51: DECI-11 project list.

6.1.4 *DECI-12 Project List*

PRACE Site (home)	Projectname	PRACE Site (exec)	Machine(s)	Enabling work (PMs)
BSC	COIMBRALATT 2	CSC	Sisu (Sandy Bridge@2.6)	0 ~ 1
	PRIMO	ICHEC	Fionn-thin (Fionn-thin)	0 ~ 1
		WCSS	Supernova (XEON X5650@2.67)	
CINECA	MoTOFET	CYFRONET	0 (no results)	0 ~ 1
	MUPPIBOX	NIIF	0 (no results)	0 ~ 1
	TCSF	CYFRONET	0 (no results)	0 ~ 1
CSC	ALLOTRANS	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	complexIdyn	PDC	Lindgren (XE6 12C@2.1)	0 ~ 1
	PlanckLFI4	CSC	Sisu (Sandy Bridge@2.6)	1 ~ 3
EPCC	DynaMITE	not assigned	not assigned	0 ~ 1
	fusionFEM	not assigned	not assigned	0 ~ 1

	GalChem2	not assigned	not assigned	0 ~ 1
	LocalUniverse	not assigned	not assigned	0 ~ 1
	NANOMOF	not assigned	not assigned	0 ~ 1
	nanostrain	not assigned	not assigned	0 ~ 1
	NAV	not assigned	not assigned	0 ~ 1
	OSFMO	not assigned	not assigned	0 ~ 1
	QMC-H2	not assigned	not assigned	0 ~ 1
	TOACD	not assigned	not assigned	0 ~ 1
GRNET	CepFlow	not assigned	not assigned	0 ~ 1
	ELSOC	not assigned	not assigned	0 ~ 1
	Multi2D	not assigned	not assigned	0 ~ 1
	NANOGRAPHENE	not assigned	not assigned	0 ~ 1
HLRS	MoMoGal2	SURFsara	Cartesius (Cartesius)	0 ~ 1
		CYFRONET	Zeus BigMem (AMD 6276@2.3)	
PDC	DNSTF2	CSC	Sisu (Sandy Bridge@2.6)	3 ~ 6
	EXODUS	CASTORC	Cy-Tera (Cy-Tera)	0 ~ 1
		CYFRONET	Zeus BigMem (AMD 6276@2.3)	
	FENICS	PDC	Lindgren (XE6 12C@2.1)	3 ~ 6
	MicroMagNum	ICHEC	Fionn-thin (Fionn-thin)	0 ~ 1
	ParaWEM	EPCC	Archer (Sandy Bridge@2.6)	1 ~ 3
		PDC	Lindgren (XE6 12C@2.1)	
	VFEH	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
SURFsara	ACEID	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	GLAM	EPCC	Archer (Sandy Bridge@2.6)	0 ~ 1
	scHeaTrans	SURFsara	Cartesius (Cartesius)	0 ~ 1
		VSB-TUO	Anselm (E5-2665-8@2.4)	
	TheDeNoMO	SURFsara	Cartesius (Cartesius)	0 ~ 1
UHEM	APGTBL	NIIF	0 (no results)	0 ~ 1

Table 52: DECI-12 project list.

[i] R. Muscari, M. Felli, and A. Di Mascio. Analysis of the flow past a fully appended hull with propellers by computational and experimental fluid dynamics. Journal of Fluids Engineering, 133(6), 2011.
D. Durante, R. Broglia, R. Muscari, and A. Di Mascio. Numerical Simulations of a turning circle manoeuvre for a fully appended hull. In Proc. of 28th Symposium on Naval Hydrodynamics, Pasadena, California, 2010.

S. Zaghi, R. Broglia, and A. Di Mascio. Analysis of the interference effects for highspeed catamarans by model tests and numerical simulations. Ocean Engineering, 38(1718):2110--2122, 2011.
R. Muscari, A. Di Mascio, and R. Verzicco. Modeling of vortex dynamics in the wake of a marine propeller. caf, 73(0):65--79, 2013.
A. Di Mascio, R. Broglia, and R. Muscari. On the Application of the One-Phase Level Set Method for Naval Hydrodynamic Flows. Computer and Fluids, 36(5):868--886, 2007.

[ii] A. Di Mascio, S. Zaghi, R. Muscari, R. Broglia, B. Favini, and A. Scaccia. On the Aerodynamic Heating of VEGA Launcher: Compressible Chimera Navier-Stokes Simulation with Complex Surfaces. In 7th European Aerothermodynamics Symposium, Brugge, Belgium, 9-12 Maggio, 2011.

[iii] S. Zaghi, R. Muscari, and A. Di Mascio. Cnr-insean: Numerical simulations of wind turbines by means of dynamic overset grids. In "Blind test 2" Workshop Calculations for two wind turbines in line, Trondheim, Norway, 2012.