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

D7.1.3
Applications Enabling for Tier-0

Final

Version: 1.0
Author(s): Alexander Schnurpfeil, FZJ
Date: 24.1.2015

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: D7.1.3	
	Deliverable Nature: Report	
	Deliverable Level: PU *	**Contractual Date of Delivery:** 31 / January / 2015
		Actual Date of Delivery: 31 / January / 2015
	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: Applications Enabling for Tier-0**	
	ID: D7.1.3	
	Version: 1.0	**Status: Final**
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D7.1.3.docx	
Authorship	**Written by:**	Alexander Schnurpfeil, FZJ
	Contributors:	Jörg Hertzer, HLRS Mads Ruben Burgdorff Kristensen, UCPH Petar Jovanovic, IPB Mohammad Jowkar, BSC Adrien Cassagne, CINES Thomas Ponweiser, JKU Alexandra Charalampidou, GRNET Gabriel Hautreux, CINES Bertrand Cirou, CINES Tristan Cabel, CINES Dimitris Dellis, IASA Volker Weinberg, LRZ Antoni, Artiguez Barcelo, BSC
	Reviewed by:	Dietmar Erwin, FZJ Jussi Enkovaara, CSC
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	09/December/2014	Draft	Set up structure of this document
0.2	11/December/2014	Draft	Section 2.1, 2.2, 2.3 included
0.3	12/December/2014	Draft	Application Evaluation Forms from Cut-off December 2013 included
0.4	16/December/2014	Draft	Application Evaluation Forms from Cut-off March 2014 included
0.5	17/December/2014	Draft	Summary added
0.6	07/January/2015	Draft	Before project-internal review
0.7	22/January/2015	Draft	After project-internal review
1.0	23/January/2015	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-ri.eu/IMG/pdf/d7.1.2_3ip.pdf

[3] Scalability Limitations of CESM Simulation on JUQUEEN, http://www.prace-ri.eu/IMG/pdf/WP200.pdf

[4] Improving the Scalability of the Overlapping Fragments Method Code, http://www.prace-ri.eu/IMG/pdf/WP201.pdf

[5] Parallel Mesh Partitioning in Alya, http://www.prace-ri.eu/IMG/pdf/WP202.pdf

[6] High-order Method for a New Generation of Large Eddy Simulation Solver, http://www.prace-ri.eu/IMG/pdf/WP203.pdf

[7] Optimizing the post-Wannier Berry-phase Code for Optical and Anomalous Hall Conductivities and Orbital Magnetization, http://www.prace-ri.eu/IMG/pdf/WP204.pdf

[8] Memory Optimization for the Octopus Scientific Code, http://www.prace-ri.eu/IMG/pdf/WP205.pdf

[9] http://www.prace-ri.eu/white-papers/

[10] J. M. Seminario, Calculation of Intramolecular Force Fields from Second-Derivative Tensors, International Journal of Quantum Chemistry, Quantum Chemistry Symposium 30, 1271 -1277 (1996).

[11] https://github.com/ponweist/Wannier90

[12] A. A. Mostofi, “WANNIER90: A tool for obtaining maximally-localised Wannier functions”, Computer Physics Communications, 178, 685 (2008).

[13] http://www.wannier.org

[14] P. Giannozzi, S. Baroni, N. Bonini, et al, “Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials”, J.Phys.:Condens.Matter 21, 395502, (2009)

[15] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, N. R. Tallent, “HPCToolkit: Tools for performance analysis of optimized parallel programs”, Concurrency and Computation: Practice and Experience 22 (2010) 685–701.

List of Acronyms and Abbreviations

AAA	Authorization, Authentication, Accounting.
ACF	Advanced Computing Facility
ADP	Average Dissipated Power
AISBL	Association International Sans But Lucratif (legal form of the PRACE-RI)
AISBL	Association sans but lucrative (legal form of the PRACE RI)
AMD	Advanced Micro Devices
APGAS	Asynchronous PGAS (language)
API	Application Programming Interface
APML	Advanced Platform Management Link (AMD)
ASIC	Application-Specific Integrated Circuit
ATI	Array Technologies Incorporated (AMD)
BAdW	Bayerischen Akademie der Wissenschaften (Germany)
BCO	Benchmark Code Owner
BLAS	Basic Linear Algebra Subprograms
BSC	Barcelona Supercomputing Center (Spain)
CAF	Co-Array Fortran
CAL	Compute Abstraction Layer
CCE	Cray Compiler Environment
ccNUMA	cache coherent NUMA
CEA	Commissariat à l'énergie atomique et aux énergies alternatives
CGS	Classical Gram-Schmidt
CGSr	Classical Gram-Schmidt with re-orthogonalisation
CINECA	Consorzio Interuniversitario, the largest Italian computing centre (Italy)
CINES	Centre Informatique National de l'Enseignement Supérieur (represented in PRACE by GENCI, France)
CLE	Cray Linux Environment
CPU	Central Processing Unit
CSC	Finnish IT Centre for Science (Finland)
CSCS	The Swiss National Supercomputing Centre (represented in PRACE by ETHZ, Switzerland)
CSR	Compressed Sparse Row (for a sparse matrix)
CUDA	Compute Unified Device Architecture (NVIDIA)
DARPA	Defense Advanced Research Projects Agency
DDN	DataDirect Networks
DDR	Double Data Rate
DEISA	Distributed European Infrastructure for Supercomputing Applications. EU project by leading national HPC centres.
DGEMM	Double precision General Matrix Multiply
DIMM	Dual Inline Memory Module
DMA	Direct Memory Access
DNA	DeoxyriboNucleic Acid
DP	Double Precision, usually 64-bit floating point numbers
DRAM	Dynamic Random Access memory
EC	European Community
EESI	European Exascale Software Initiative
EoI	Expression of Interest
EP	Efficient Performance, e.g., Nehalem-EP (Intel)

EPCC	Edinburg Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)
EPSRC	The Engineering and Physical Sciences Research Council (United Kingdom)
eQPACE	extended QPACE, name of the FZJ WP8 prototype
ETHZ	Eidgenössische Technische Hochschule Zuerich, ETH Zurich (Switzerland)
ESFRI	European Strategy Forum on Research Infrastructures; created roadmap for pan-European Research Infrastructure.
EX	Expandable, e.g., Nehalem-EX (Intel)
FC	Fiber Channel
FFT	Fast Fourier Transform
FHPCA	FPGA HPC Alliance
FP	Floating-Point
FPGA	Field Programmable Gate Array
FPU	Floating-Point Unit
FZJ	Forschungszentrum Jülich (Germany)
GASNet	Global Address Space Networking
GB	Giga (= 2^{30} ~ 10^9) Bytes (= 8 bits), also GByte
Gb/s	Giga (= 10^9) bits per second, also Gbit/s
GB/s	Giga (= 10^9) Bytes (= 8 bits) per second, also GByte/s
GCS	Gauss Centre for Supercomputing (Germany)
GDDR	Graphic Double Data Rate memory
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 Calcul Intensif (France)
GFlop/s	Giga (= 10^9) Floating point operations (usually in 64-bit, i.e. DP) per second, also GF/s
GHz	Giga (= 10^9) Hertz, frequency =10^9 periods or clock cycles per second
GigE	Gigabit Ethernet, also GbE
GLSL	OpenGL Shading Language
GNU	GNU's not Unix, a free OS
GPGPU	General Purpose GPU
GPU	Graphic Processing Unit
GS	Gram-Schmidt
GWU	George Washington University, Washington, D.C. (USA)
HBA	Host Bus Adapter
HCA	Host Channel Adapter
HCE	Harwest Compiling Environment (Ylichron)
HDD	Hard Disk Drive
HE	High Efficiency
HET	High Performance Computing in Europe Taskforce. Taskforce by representatives from European HPC community to shape the European HPC Research Infrastructure. Produced the scientific case and valuable groundwork for the PRACE project.
HMM	Hidden Markov Model
HMPP	Hybrid Multi-core Parallel Programming (CAPS enterprise)
HP	Hewlett-Packard
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing

HPCC	HPC Challenge benchmark, http://icl.cs.utk.edu/hpcc/
HPCS	High Productivity Computing System (a DARPA program)
HPL	High Performance LINPACK
HT	HyperTransport channel (AMD)
HWA	HardWare accelerator
IB	InfiniBand
IBA	IB Architecture
IBM	Formerly known as International Business Machines
ICE	(SGI)
IDRIS	Institut du Développement et des Ressources en Informatique Scientifique (represented in PRACE by GENCI, France)
IEEE	Institute of Electrical and Electronic Engineers
IESP	International Exascale Project
IL	Intermediate Language
IMB	Intel MPI Benchmark
I/O	Input/Output
IOR	Interleaved Or Random
IPMI	Intelligent Platform Management Interface
ISC	International Supercomputing Conference; European equivalent to the US based SC0x conference. Held annually in Germany.
IWC	Inbound Write Controller
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)
LBE	Lattice Boltzmann Equation
LINPACK	Software library for Linear Algebra
LLNL	Laurence Livermore National Laboratory, Livermore, California (USA)
LQCD	Lattice QCD
LRZ	Leibniz Supercomputing Centre (Garching, Germany)
LS	Local Store memory (in a Cell processor)
MB	Mega (= 2^{20} ~ 10^6) Bytes (= 8 bits), also MByte
MB/s	Mega (= 10^6) Bytes (= 8 bits) per second, also MByte/s
MDT	MetaData Target
MFC	Memory Flow Controller
MFlop/s	Mega (= 10^6) Floating point operations (usually in 64-bit, i.e. DP) per second, also MF/s
MGS	Modified Gram-Schmidt
MHz	Mega (= 10^6) Hertz, frequency =10^6 periods or clock cycles per second
MIPS	Originally Microprocessor without Interlocked Pipeline Stages; a RISC processor architecture developed by MIPS Technology
MKL	Math Kernel Library (Intel)
ML	Maximum Likelihood
Mop/s	Mega (= 10^6) operations per second (usually integer or logic operations)
MoU	Memorandum of Understanding.
MPI	Message Passing Interface
MPP	Massively Parallel Processing (or Processor)
MPT	Message Passing Toolkit
MRAM	Magnetoresistive RAM
MTAP	Multi-Threaded Array Processor (ClearSpead-Petapath)
mxm	DP matrix-by-matrix multiplication mod2am of the EuroBen kernels

NAS	Network-Attached Storage
NCF	Netherlands Computing Facilities (Netherlands)
NDA	Non-Disclosure Agreement. Typically signed between vendors and customers working together on products prior to their general availability or announcement.
NoC	Network-on-a-Chip
NFS	Network File System
NIC	Network Interface Controller
NUMA	Non-Uniform Memory Access or Architecture
OpenCL	Open Computing Language
OpenGL	Open Graphic Library
Open MP	Open Multi-Processing
OS	Operating System
OSS	Object Storage Server
OST	Object Storage Target
PCIe	Peripheral Component Interconnect express, also PCI-Express
PCI-X	Peripheral Component Interconnect eXtended
PGAS	Partitioned Global Address Space
PGI	Portland Group, Inc.
pNFS	Parallel Network File System
POSIX	Portable OS Interface for Unix
PPE	PowerPC Processor Element (in a Cell processor)
PRACE	Partnership for Advanced Computing in Europe; Project Acronym
PSNC	Poznan Supercomputing and Networking Centre (Poland)
QCD	Quantum Chromodynamics
QCDOC	Quantum Chromodynamics On a Chip
QDR	Quad Data Rate
QPACE	QCD Parallel Computing on the Cell
QR	QR method or algorithm: a procedure in linear algebra to compute the eigenvalues and eigenvectors of a matrix
RAM	Random Access Memory
RDMA	Remote Data Memory Access
RISC	Reduce Instruction Set Computer
RNG	Random Number Generator
RPM	Revolution per Minute
SAN	Storage Area Network
SARA	Stichting Academisch Rekencentrum Amsterdam (Netherlands)
SAS	Serial Attached SCSI
SATA	Serial Advanced Technology Attachment (bus)
SDK	Software Development Kit
SGEMM	Single precision General Matrix Multiply, subroutine in the BLAS
SGI	Silicon Graphics, Inc.
SHMEM	Share Memory access library (Cray)
SIMD	Single Instruction Multiple Data
SM	Streaming Multiprocessor, also Subnet Manager
SMP	Symmetric MultiProcessing
SNIC	Swedish National Infrastructure for Computing (Sweden)
SP	Single Precision, usually 32-bit floating point numbers
SPE	Synergistic Processing Element (core of Cell processor)
SPH	Smoothed Particle Hydrodynamics
SPU	Synergistic Processor Unit (in each SPE)

SSD	Solid State Disk or Drive
STFC	Science and Technology Facilities Council (represented in PRACE by EPSRC, United Kingdom)
STRATOS	PRACE advisory group for STRAtegic TechnOlogieS
STT	Spin-Torque-Transfer
SURFsara	Dutch national High Performance Computing & e-Science Support Center
TARA	Traffic Aware Routing Algorithm
TB	Tera (= 240 ~ 1012) Bytes (= 8 bits), also TByte
TCO	Total Cost of Ownership. Includes the costs (personnel, power, cooling, maintenance, ...) in addition to the purchase cost of a system.
TDP	Thermal Design Power
TFlop/s	Tera (= 1012) Floating-point operations (usually in 64-bit, i.e. DP) per second, also TF/s
Tier-0	Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1
UFM	Unified Fabric Manager (Voltaire)
UNICORE	Uniform Interface to Computing Resources. Grid software for seamless access to distributed resources.
UPC	Unified Parallel C
UV	Ultra Violet (SGI)
VHDL	VHSIC (Very-High Speed Integrated Circuit) Hardware Description Language

Executive Summary

This deliverable covers the activities from WP7 Task 7.1.A "Petascaling & Optimisation Support for Preparatory Access Projects" which is part of the PRACE-3IP extension[1] phase. T7.1.A is a persistent service which provides code enabling and optimisation to European researchers as well as to commercial 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). Seven PA C projects have been carried out during the PRACE-3IP extension. Additionally, a total of five projects have started in the current project phase but will continue beyond its end. The outcome of these projects is expected to be reported in a future PRACE implementation phase project. This report focuses on the optimizations done and results achieved by the completed projects during the PRACE-3IP extension. The statistics about the PA C calls as well as a description of the call organization itself is also included. The results of the completed projects are documented in white papers which are published on the PRACE-RI website [1].

1 Introduction

Computational simulations have proved to be a promising way of finding answers to research problems from a wide range of scientific fields. However, such complex problems often have such high demands regarding the needed computation time that these cannot be met 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 commercial 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 code to make efficient use of the provided supercomputers. The complexity of supercomputers requires a high level of experience and advanced knowledge of different concepts regarding programming techniques, parallelization strategies, etc. Such demands often cannot be met by the applicants themselves and thus special assistance by supercomputing experts is essential. 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 as well as the assignment of PRACE experts to these proposals. Furthermore, the support itself is provided and monitored within this task. Section 2.1 gives a more detailed description and facts 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 are detailed in Section 2.2, Section 2.3 and Section 2.4 respectively. The contents of Sections 2.2-2.4 can already be found in deliverable D7.1.2 [2]. They are repeated here for completeness and the benefit of the reader. Section 2.5 gives an overview about the Preparatory Access type C projects covered in the 3IP extension. The announcement of the call is described briefly in Section 2.6. Finally, the work done within the projects along with the outcome of the optimization work is presented in Section 2.7 and Section 2.8. The deliverable closes with a summary in Section 3 and points out the outcome of Task 7.1.A.

[1] PRACE-3IP Extension denotes the period of M25-M31 extending the work of WP2 – WP7 by seven month in order to ensure a seamless and continuous support of the project for the PRACE RI prior to the planned start of the PRACE-4IP project in H2020

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 requirements, and runtime set up. There are many important scientific and commercial applications which do not meet these criteria today. 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 allows for submission of proposals at any time whereby the review of these proposals takes place takes place every three months. This procedure is also referred to as Cut-off. Therefore, new projects can be admitted for preparatory purposes to PRACE Tier-0 systems once every quarter. It is possible to choose between three different types of access:

- Type A is meant for code scalability tests the outcome of which is to be included in the proposal in 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 receive special assistance by PRACE experts to support the optimization requests. As well as access to the Tier-0 systems the applicants also apply for 1 to 6 PMs of supporting work to be performed by PRACE experts.

The following Tier-0 systems were available for PA:

- CURIE, BULL Bullx cluster at GENCI-CEA, France (thin, fat, and hybrid nodes are available)
- FERMI, IBM Blue Gene/Q at CINECA, Italy
- HORNET, Cray XC40 (GCS@HLRS, Germany), replacing HERMIT, CRAY XE6
- MARENOSTRUM, IBM System X iDataplex at BSC, Spain
- SUPERMUC, IBM System X iDataplex at GCS-LRZ, Germany
- JUQUEEN, IBM Blue Gene/Q at GCS-JSC, Germany

2.1 Cut-off specific facts and numbers

In the PRACE-3IP extension phase three Cut-offs for PA took place resulting in six projects. While the projects from Cut-off June 2014 / Cut-off September 2014 are currently active, the proposal from the Cut-off December 2014 – there is only one – is currently being evaluated. Any results from these projects will be reported in a possible future deliverable. Section 3 gives an overview about the status of these projects. Projects from Cut-off December 2013 and Cut-off March 2014 were initialized in PRACE-3IP and continued by the PRACE-3IP extension. Their results are presented in this deliverable.

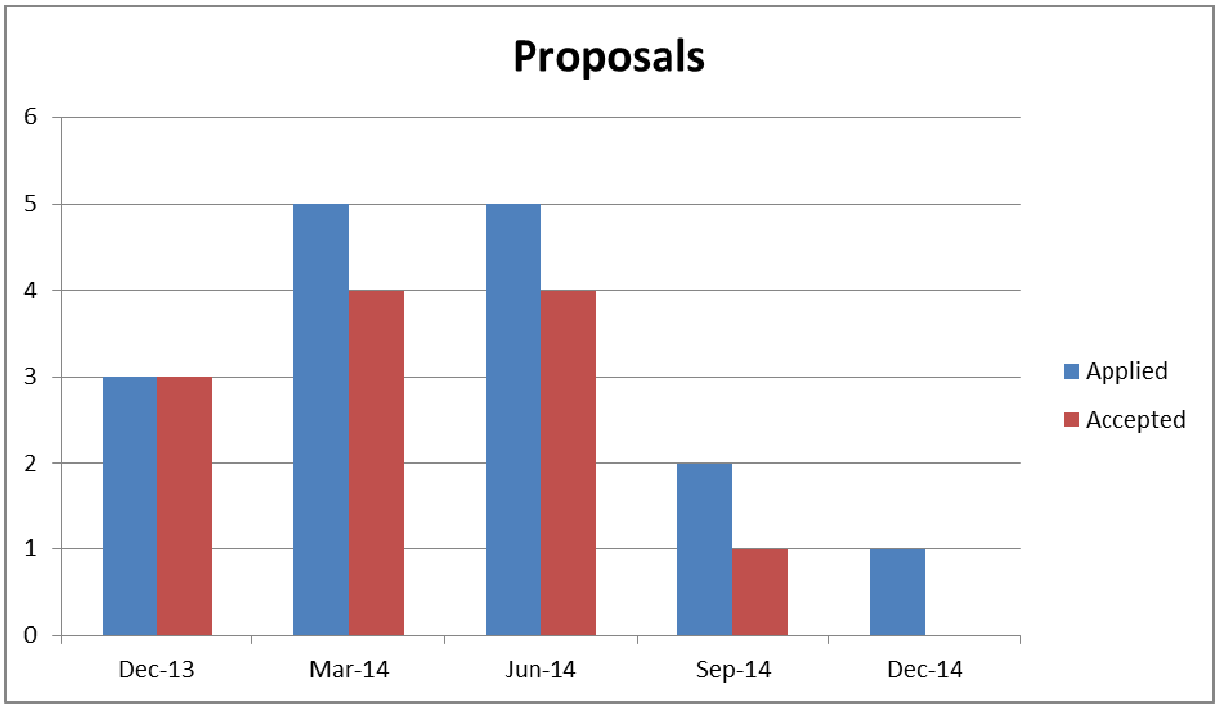

Figure 1: Number of submitted and accepted proposals for PA type C per Cut-off.

Figure 1 presents the number of proposals which have been accepted and rejected respectively for each Cut-off covered in this deliverable. In total 12 out of 16 proposals were accepted. Cut-off December 14 is currently in progress and therefore the final status is not yet available

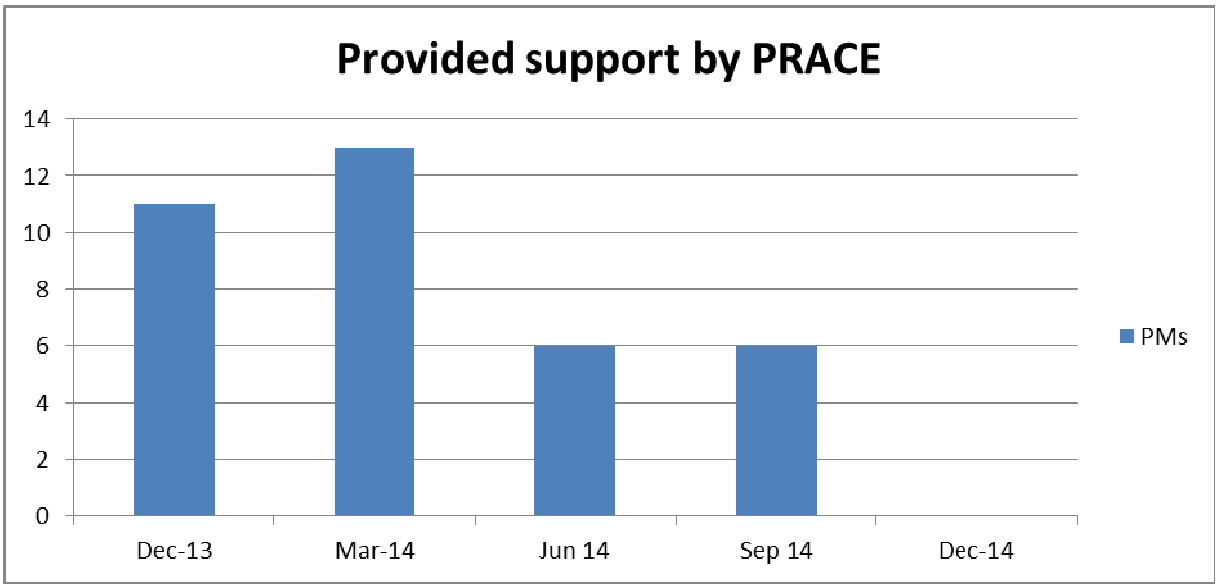

Figure 2: Amount of PMs assigned to PA type C projects per Cut-off.

for the report.

Figure 2 gives an overview of the number of PMs assigned to the projects per Cut-off. In total 36 PMs were made available to these projects. Dedicated PMs were partly utilized during PRACE-3IP as well as during the extension phase.

Finally, Figure 3 provides an overview of the scientific fields which are covered by the supported projects.

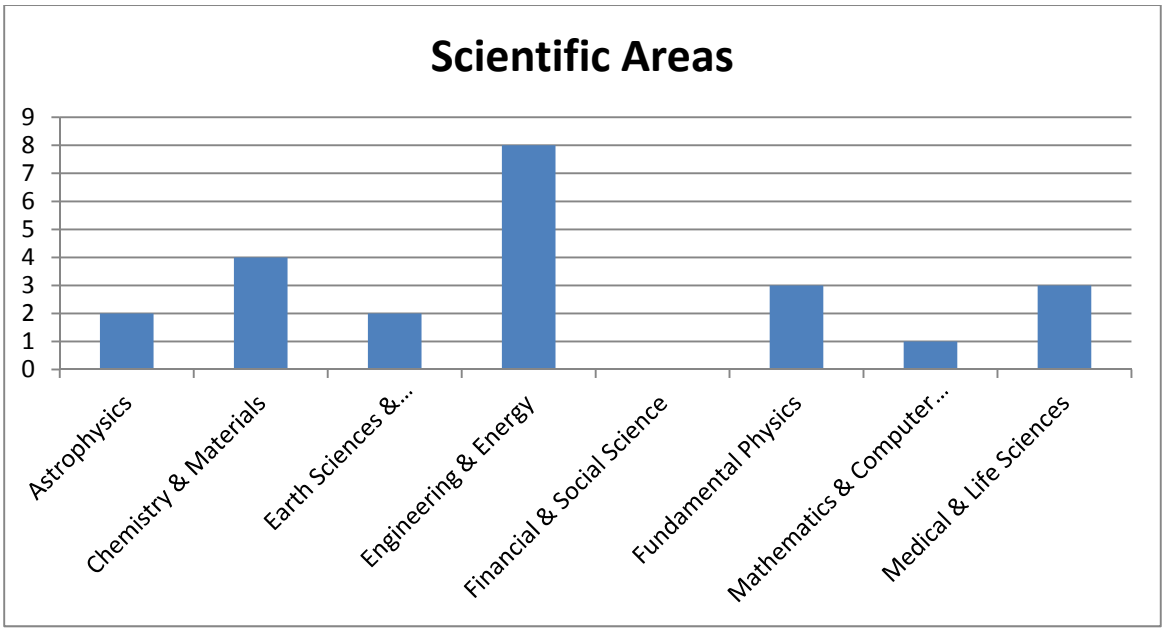

Figure 3: Number of projects per scientific field.

2.2 Review Process

The organization of the review procedure, the assignment of PRACE collaborators and the supervision of the PA C projects are managed 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 only four times a year it is necessary to keep the review process swift and efficient. A close collaboration between AISBL, T7.1.A and the hosting sites is important in this context. The process for both the technical and the 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 completed 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 on 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 to a broader community, and is it possible to integrate the achieved results 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 evaluates 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 shortcomings. In one case poor scaling potential of the underlying code was the main reason for rejection. In two other cases, suitable support could not be ensured.

2.3 Assigning of PRACE collaborators

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

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

Should the response not be sufficient to cover the support requirements 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 areas of expertise.

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

This procedure has proven to be extremely successful; only very few proposals had to be refused in the past due to a lack of available support.

The assignment of PRACE experts takes place concurrently to the review process so that the entire review can be completed within six weeks. This has proven itself to be a suitable approach, as the resulting overhead is negligible due to the low number of projects being rejected.

As soon as the review process 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:

- Formulating a detailed work plan together with the applicant,
- Participating in the optimization work,
- Reporting the status 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

Task 7.1.A includes the supervision of the Type C projects. This is challenging as the projects' durations (six months) and the intervals of the Cut-offs (3 months) do not cleanly align with each other. Due to this, projects do not necessarily start and end concurrently but overlap, i.e. at each point in time different projects might be in different phases. To solve this problem, a phone conference takes place in task 7.1.A every two months to discuss the status of running projects, to advise 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.

Twice a year, 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. Such a meeting was not scheduled in the frame of the PRACE-3IP extension phase.

2.5 PRACE Preparatory Access type C projects covered by the 3IP extension

Projects from Cut-off December 2013 / Cut-off March 2014 have their origin in PRACE-3IP but were finalized in the PRACE-3IP extension and their results are reported in this deliverable. Table 1 lists the corresponding projects.

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
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

Cut-offs December 2013/March 2014	
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	**High-order method for a new generation of large eddy simulation solver**
Project leader	Jean-François Boussuge
PRACE expert	Adrien Cassagne
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 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

Cut-offs December 2013/March 2014	
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 were actually finalised in the extension phase of PRACE-3IP.

Projects from Cut-off June 2014 and beyond were initiated in the PRACE-3IP extension phase but will only be finished by the end of January 2015. The final reports as well as the corresponding white papers are currently being produced. Therefore, results cannot be presented in this deliverable but will possibly be reported in a later deliverable.

Cut-offs June 2014/September 2014	
Title	**OpenFOAM capability for industrial large scale computation of the multiphase flow of future automotive component: step 2**
Project leader	Jerome Helie
PRACE expert	Gabriel Hautreux, Bertrand Cirou
PRACE facility	CURIE TN
PA number	2010PA2431
Project's start	01-Aug-14
Project's end	01-Feb-15
Title	**Numerical modelling of the interaction of light waves with nanostructures using a high order discontinuous finite element method**
Project leader	Stéphane Lanteri
PRACE expert	Gabriel Hautreux, Tristan Cabel
PRACE facility	CURIE TN, CURIE FN
PA number	2010 PA2452
Project's start	15-Jul-14
Project's end	15-Jan-15

Cut-offs June 2014/September 2014	
Title	**Large scale parallelized 3d mesoscopic simulations of the mechanical response to shear in disordered media**
Project leader	Kirsten Martens
PRACE expert	Dimitris Dellis
PRACE facility	CURIE TN
PA number	2010PA2457
Project's start	01-Aug-14
Project's end	01-Feb-15
Title	**PICCANTE: an open source particle-in-cell code for advanced simulations on tier-0 systems**
Project leader	Andrea Macchi
PRACE expert	Volker Weinberg
PRACE facility	FERMI, JUQUEEN
PA number	2010PA2458
Project's start	15-Jul-14
Project's end	15-Jan-15
Title	**Parallel subdomain coupling for non-matching mesh problems in ALYA**
Project leader	Guillaume Houzeaux
PRACE expert	Juan Carlos Caja
PRACE facility	MARENOSTRUM, FERMI
PA number	2010PA2486
Project's start	01-Nov-14
Project's end	30-Apr-15

Table 2: Projects which were established in the PRACE-3IP extension phase but will be finalised in a future PRACE implementation phase.

The evaluation of the December proposal is currently in progress and is therefore not listed here.

2.6 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 was again highlighted as a unique opportunity to improve code performance and as a means of getting code 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 page [9] and are also referenced by this deliverable.

2.7 Cut-off December 2013

This and the following two sections describe the optimizations performed on the Preparatory Projects type C. The projects are listed in accordance with the Cut-off dates in which they appeared. General information regarding the optimization work done as well as the gained results is presented here using the recommended evaluation form. The application evaluation form ensures a consistent and coherent presentation of all projects which were managed in the frame of PA C. Additionally the white papers created by these projects are referenced so that the interested reader is provided with further information.

2.7.1 *Development of a package for computer aided drug design, 2010PA2141*

Code general features

Name	GROMACS 4.6.4
Scientific field	Molecular Dynamics
Short code description	GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers.
Programming language	C
Supported compilers	Intel, GNU, PGI, Cray
Parallel implementation	MPI
Accelerator support	yes
Libraries	FFTW Libsci
Building procedure	CMake
Web site	http://www.gromacs.org/
Licence	GNU Lesser General Public License (LGPL), Version 2.1

Table 3: Code general features for GROMACS 4.6.4.

Name	CP2K 2.4
Scientific field	Molecular Dynamics
Short code description	CP2K is a program to perform atomistic and molecular simulations of solid state, liquid, molecular, and biological systems. It provides a general framework for different methods such as e.g., density functional theory (DFT) using a mixed Gaussian and plane waves approach (GPW) and classical pair and many-body potentials.
Programming language	Fortran
Supported compilers	GNU
Parallel implementation	MPI / OpenMP
Accelerator support	yes
Libraries	Libint libsci
Building procedure	Makefile
Web site	http://www.cp2k.org/
Licence	GPL

Table 4: Code general features for CP2K 2.4

Main objectives:

A software package for the computation of molecular mechanics force field parameters of drug candidates is currently under development. It uses CP2K for required quantum mechanics calculations and GROMACS for molecular dynamics simulations of drug candidates with a target.

The first program module reads an XYZ file containing the structure of a drug candidate molecule and generates a CP2K input file adding necessary keywords for a proper structure optimization. The second one reads the optimized CP2K geometry and creates an input file with necessary keywords for the frequency analysis and electron density map calculations. The next module prepares the input file for *g_x2top* program from GROMACS package using the optimized geometry of a drug like candidate and runs it.

Another module uses a new heuristic algorithm based on a set of rules to determine atom types for the GROMACS topology file.

The most important module reads required data such as the hessian matrix and the masses and calculates all necessary force field constants. For the parameter calculation we used and implemented the method published in [10]. The hessian matrix obtained by a CP2K frequency analysis run is used to determine intramolecular force constants. The module also writes a GROMACS topology file with calculated force constants. The last module generates input files for molecular dynamics simulations and runs GROMACS.

The goal of the project is to test the developed package with real models on supercomputing platforms.

Accomplished work:

Main goal was to test the scalability of molecular dynamics simulations with GROMACS 4.6.4(5) of neutralized and water solvated ribosome units.

The work of the investigators was described by the PI as follows:

Because of poor scalability of CP2K tasks due to the relatively small number of atoms (on the order of 100 for commonly used drug candidates) we developed a new heuristic approach for splitting bigger drug candidate molecules to a finite number of smaller fragments which were converted to chemically correct small tasks. These fragments can be calculated in many separate tasks which speeds up the whole calculation. This approach also circumvents the well known problem of non-linear growth of required resources (memory, cpu) when increasing the number of orbital functions. Smaller molecules also have a wider tolerance to a poor starting geometry.

During testing we found and fixed some bugs. Some internal coefficients were also tuned.

Main results:

The main outcome of the project is that CP2K and GROMACS software packages are suitable for performing the most time consuming stages in drug design development namely force field parameters assignment and molecular dynamics docking on a large target molecule such as ribosome.

The molecular dynamics setup was Force Field CHARMM27, md step 2 ps, NPT ensemble system size about 2 200 000 atoms (nucleic acides + protein + water + ions).

One MPI process per node with 32 OpenMP thread was used to run the jobs on HERMIT and two MPI processes per node with 16 OpenMP threads on JUQUEEN. The latter was consistent with the warnings that GROMACS is not fully operational when using more than 32 threads and the loss of scalability in very small domains (the bigger number of MPI processes the smaller domains) on the other.

The data presented in the following figures and tables shows that the molecular dynamics simulation scaled better on the HERMIT machine during testing.

Number of cores	Wall clock time [s]	Speed-up vs the first one	Number of Nodes	Number of MPI process
8192	14782	1.0	256	512
16384	9152	1.6	512	1024
32768	6046	2.4	1024	2048
65536	5958	2.5	2048	4096
131072	5692	2.6	4096	8192

Table 5: BG/Q – JUQUEEN

Number of cores	Wall clock time [s]	Speed-up vs the first one	Number of Nodes	Number of MPI process
512	2762	1	16	16
1024	1602	1.7	32	32
2048	919	3.0	64	64
4096	506	5.5	128	128
8192	360	7.7	256	256

Table 6: CRAY XE6 – HERMIT

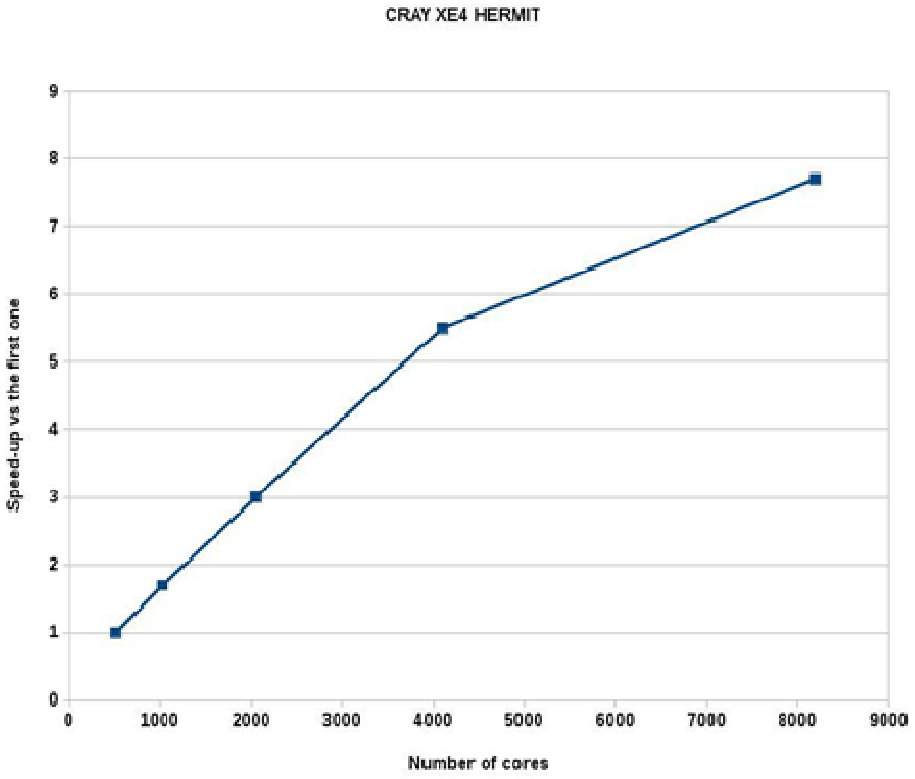

Figure 4: Scaling behaviour on JUQUEEN

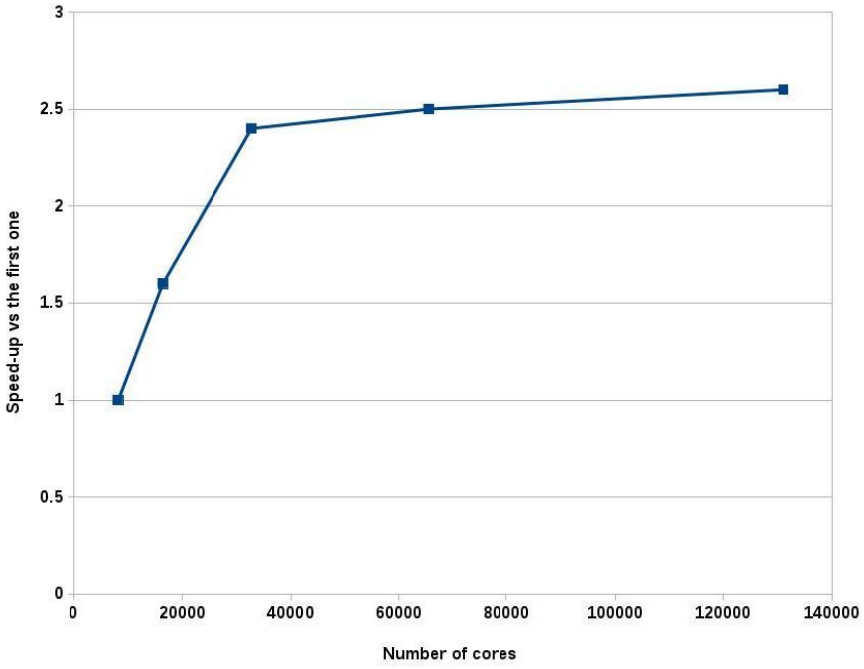

Figure 5: Scaling behaviour on HERMIT

2.7.2 *Very high resolution Earth System Model (CESM) energy flux and wind stress sensitivity experiments, 2010PA2066*

Code general features

Name	Community Earth System Model *(CESM)*
Scientific field	Climate research
Short code description	The Community Earth System Model (CESM) is a fully coupled, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states. CESM supports a broad range of architecture including the Blue Gene series B, P, and Q
Programming language	Fortran and C
Supported compilers	GCC
Parallel implementation	Yes
Accelerator support	No
Libraries	
Building procedure	
Web site	http://www2.cesm.ucar.edu/
Licence	Mixed Open Source licenses

Table 7: Code general features for CESM.

Main objectives:

The purpose of this project was the installation and optimization of the CESM model on JUQUEEN, an IBM Blue Gene/Q supercomputer. The focus of the project was the development and application of a heuristic load-balancing algorithm for fully coupled ultra-high-resolution model runs.

Accomplished work:

However, it turns out that running vanilla CESM v1.2.2 fails when scaling above 128 CPU-cores on JUQUEEN – the execution crashes with an out-of-memory error in the communication layer. Thus, rather than develop load-balancing algorithms, the focus of this project changed to the development of a set of workarounds that makes it possible to scale CESM v1.2.2 above 128 CPU-cores when running on JUQUEEN.

Main results:

In order to run CESM on JUQUEEN, we have to use a number of workarounds. Because the recent CESM v1.2.2 release is not supported on JUQUEEN, one has to back port the machine specific files from a CESM v1.3 development version. Our colleagues, J. Dennis and A. Baker, at National Center for Atmospheric Research (NCAR) did a great job and came up with the following modifications:

(1) We needed to obtain all of the CESM input data for the various resolutions from NCAR (since JUQUEEN is not yet supported, the input data has not been installed on the file system). The automated procedure for obtaining input data did not retrieve everything, so a number of files had to be obtained manually (e.g., *ice_ic*, *fsurdat*, all relevant/cpl/gridmaps files, and the pophdr file).

(2) We used the mpixlf95_r compiler (instead of mpixlf2003_r), adding "-qxlf2003=polymorphic" to the FFLAGS, "-DNO_C_SIZEOF" to the CPPDEFS, and "--enable-filesystem-hints=gpfs" to PIO_CONFIG_OPTS.

(3) Parallel NetCDF I/O (PIO) tuning: The high-resolution case (1/4 deg. atm and 1/10 ocean: ne120_t12) must use PNETCDF, and requires the following settings:

<entry id="PIO_NUMTASKS" value="256"/>

<entry id="PIO_TYPENAME" value="pnetcdf"/>

(4) In order to handle out-of-memory errors, a patch for the topology.c file was required (see appendix).

(5) We successfully ran a high-resolution case from default initial conditions for 4 days. Using 22324 MPI-tasks and 4 tasks per node (bg_size=5581), the simulation ran at a rate of 0.43 simulated years per day. This is quite slow, but subsequent attempts to incorporate threading have resulted in jobs crashing in different locations for unknown reasons.

Finally, for job submission on JUQUEEN the runjob command was used with the options that are presented here:

runjob --envs XLSMPOPTS=stack=64000000 --envs OMP_NUM_THREADS=${mthrds} --envs LOGNAME=${USER} --ranks-per-node ${MAX_TASKS_PER_NODE} --np ${ntasks} --exe \$ {EXEROOT}/cesm.exe >&! cesm.log.\$LID

With the presented workarounds, it is possible to achieve scalable performance of medium-resolution (1 deg. for atmosphere and ocean model) fully coupled CESM simulations on JUQUEEN. Figure 6 shows strong scale speedup with 128 CPU-cores as the baseline, i.e. the work size is fixed throughout all runs. At 256 and 512 CPU-cores, the speedup compared to the baseline is close to linear with a utilization of 95% and 90%, respectively. At 1024 and 3024 CPU-cores, the scalability is limited to a utilization of 76% and 62%, respectively.

Figure 7, shows the scalability of the individual CESM components. Most of the components scale fairly well but two components, River Transport Model and the Parallel Ocean Program, show no speedup when going from 1024 to 3024 CPU-cores. However, the scalability of ultra high-resolution (1/4 deg. for atmosphere and 1/10 for ocean model) fully coupled CESM simulations is limited - utilizing 22324 CPU-cores only achieves a rate of 0.43 simulated years per 24 hours of wall-time.

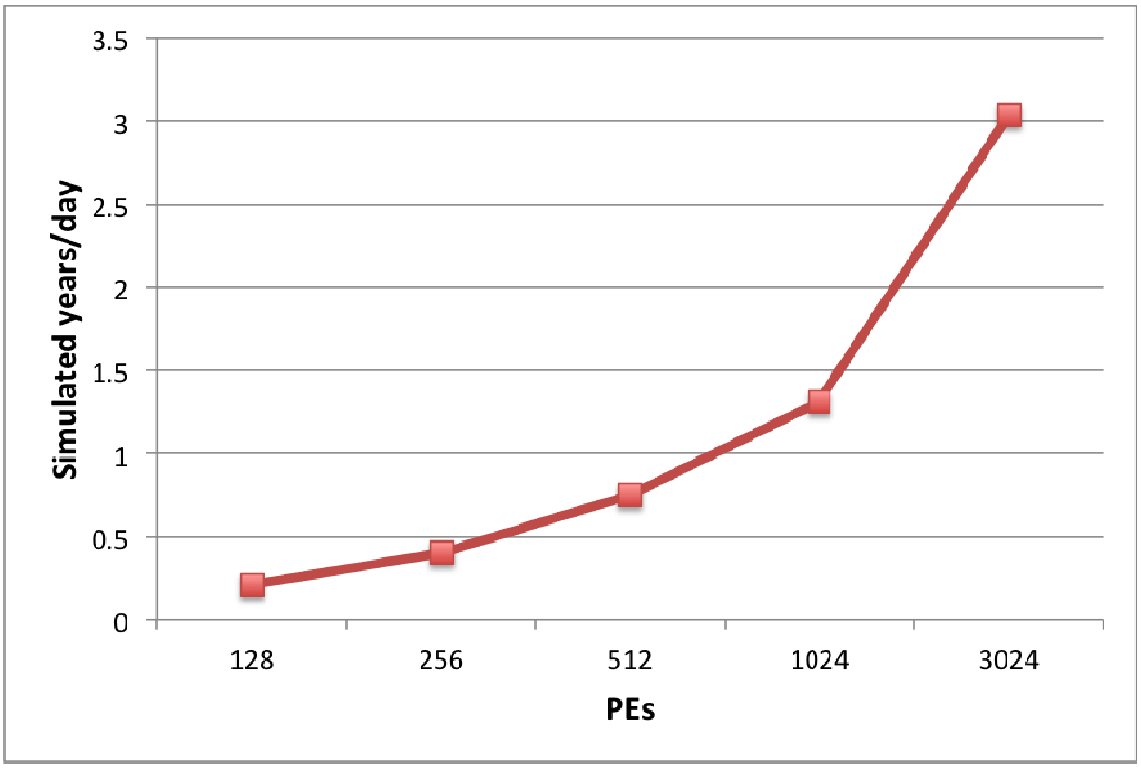

Figure 6: Fully coupled CESM comp-set with 1 x 1 deg. horizontal grid resolution for both atmosphere and ocean. Simulated years per day (24 hours) vs. number of processor elements.

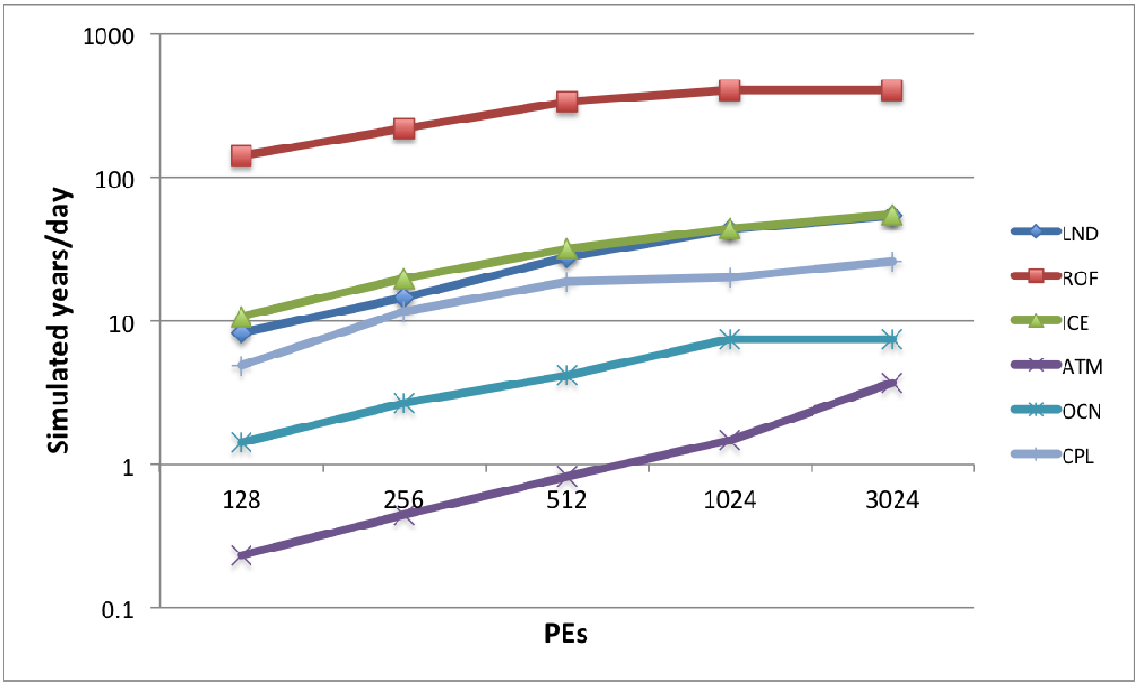

Figure 7: Fully coupled CESM comp-set with 1 x 1 deg. horizontal grid resolution for both atmosphere and ocean. Simulated years per day (24 hours) for each package vs. number of processor elements, where LND – Community Land Model (CLM), ROF – River Transport Model (RTM), ICE – sea-ice component (CICE), ATM – Community Atmosphere Model (CAM), OCN – Parallel Ocean Program (POP) and CPL coupler package.

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

2.7.3 *Improving the scalability of the overlapping fragments method code for electronic structure of organic materials, 2010PA2132*

Code general features

Name	OFM (Overlapping Fragments Method) Improving the scalability of the overlapping fragments method code for electronic structure of organic materials
Scientific field	Materials
Short code description	OFM is the code for calculation of energies and wave functions of electronic states in the spectral region near the band gap in semiconducting materials and nanostructures. It is based on the division of the system into mutually overlapping fragments (groups of atoms), the calculation of eigenstates of these fragments, the calculation of matrix elements between eigenstates of different fragments and final diagonalization of the obtained representation of the Hamiltonian. The input to the code is the single particle potential obtained from some other method. The code can be applied to rather large systems containing even tens of thousands atoms, such as disordered organic polymers, grain boundaries between organic crystals, etc.
Programming language	Fortran
Supported compilers	Intel Fortran, PGI,…
Parallel implementation	MPI
Accelerator support	None
Libraries	ACML or MKL
Building procedure	Gnu make
Web site	None
Licence	BSD 2-Clause license

Table 8: Code general features for OFM.

Main objectives:

The objective of this project was to enhance the scalability of the OFM code by improving the communication in the parts of the algorithm where orbitals of the fragments are redistributed among different nodes.

Accomplished work:

In the course of this project, we have initially performed the profiling of the original OFM code, including the scalability tests. From the profiling results, we have identified the weak points in the original code, where a significant amount of time is spent on pure communication. Therefore, we have reorganized the way in which the input potential is stored – instead of having a single copy distributed over all nodes, we keep several local copies distributed over small groups of nodes. In this way, the communication which is needed when a certain fragment needs to access the local potential is significantly speeded up. Next, we have also reorganized the implementation of the redistribution of wave functions among nodes which is required when the overlaps of wave functions from different fragments are

calculated. Finally, we have performed the profiling of the modified code, which shows improvements in comparison to the original code.

Main results:

The original OFM code was initially benchmarked by performing the weak scaling test. The runs were performed for the poly(3-hexylthiophene) (P3HT) polymer system. The smallest system consists of four P3HT chains, each 10 thiophene rings long and contains 1008 atoms altogether. The run for that system was performed with 256 CPU cores. The runs for larger systems were performed up to the system with 64512 atoms, while the number of CPU cores used was increased proportionally to the number of atoms. Figure 8 shows the dependence of CPU time (defined as the wall time multiplied by the number of CPU cores used) on the number of atoms in the system.

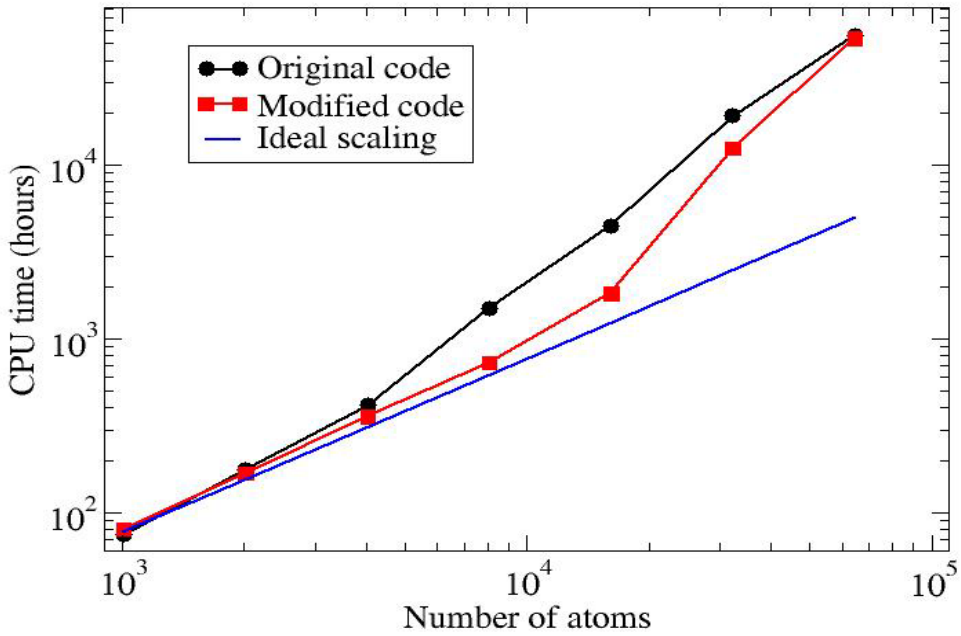

Figure 8: The dependence of CPU time on the number of atoms in the system for the simulations performed on CURIE. The number of CPU cores used for the smallest system was 256 and was increased proportionally to the number of atoms for larger systems.

As seen from Figure 8 (black circles) the original code exhibited good scalability up to 4000 atoms (1000 cores). We have identified three weak points in the code where communication significantly slows down the code and we have refactored these parts of the code to improve its performance and scalability. In particular:

1) In the original code, the input potential was read from an input file and it was then stored in memories of all nodes in such a way that each node contains only a part of the potential. The procedure for communicating the parts of the potential to their corresponding nodes had a significant impact on code performance. We have therefore reorganized the way in which the potential is stored in memory. Instead of having a single copy of the potential distributed among all nodes, we use several copies of the potential distributed among a certain group of nodes. The input potential is therefore read and subsequently broadcasted to each group of nodes. The procedure for doing this was introduced into the new code.

2) Each fragment that is stored on a small number of nodes needs only a part of the input potential. In the original code, the potential was distributed among all nodes and significant

time was needed to communicate it to a given fragment since the fragment needed to receive the data from all nodes. We have reduced this time by keeping several local copies of the potential and therefore each fragment needs to receive the data from a small group of nodes only. The procedure for communicating a part of the local copy of the potential to a given fragment was introduced into the new code.

3) In the main part of the code, one needs to calculate the overlap between wave functions of different fragments. To achieve this task one needs to bring these wave functions to the same group of nodes. In the original code, this communication was performed using a combination of send and receive commands and appeared to be inefficient. We have changed this part of the code by implementing the communication using mpi_alltoallv command which significantly reduced the time for this communication.

The performance of the code with the introduced improvements is shown in Figure 8 (red squares). These improvements extended the range of system sizes where scaling of the code is rather good up to 16000 atoms (4000 cores).

The above mentioned runs were performed on CURIE system. To test the performance of the old and the new code on a different architecture, we have benchmarked the old and the new code on the Hermit system. The obtained CPU times are presented in Figure 9. Interestingly, one obtains a rather different code scaling behaviour on HERMIT than on CURIE. We find that the original code has rather good scaling performance on HERMIT. Consequently, the improved code has only a slightly better performance.

We believe that the origin of better scaling behaviour on HERMIT in comparison to CURIE is a different node interconnect with corresponding MPI libraries. HERMIT is a CRAY XE6 system with CRAY Gemini interconnect, while CURIE has an Infiniband QDR Full Fat Tree network.

In conclusion, in the course of the project, we have refactored the OFM code in such a way that its scaling on the machine with Infiniband QDR network is improved and we have established that the original code already has good scaling properties on a machine with CRAY Gemini interconnect.

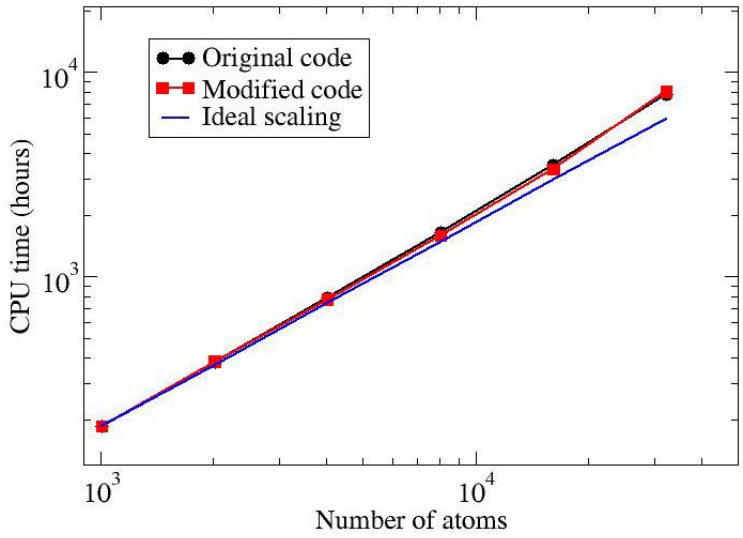

Figure 9: The dependence of CPU time on the number of atoms in the system for the simulations performed on HERMIT. The number of CPU cores used for the smallest system was 256 and was increased proportionally to the number of atoms for larger systems.

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

2.8 Cut-off March 2014

2.8.1 *Parallel mesh partitioning in Alya, 2010PA2171*

Code general features

Name	ALYA
Scientific field	Computational physics simulations
Short code description	Alya solves time-dependent Partial Differential Equations (PDEs) using the Finite Element Method (FEM). The meshes are unstructured and can be hybrid with different types of elements. Alya is specifically designed to run efficiently in supercomputers, Among the problems Alya can simulate are: • Incompressible Flows • Compressible Flows • Non-linear Solid Mechanics • Species transport equations • Excitable Media • Thermal Flows • N-body collisions • ...
Programming language	Fortran
Supported compilers	gfortran, ifort
Parallel implementation	MPI and OpenMP
Accelerator support	No
Libraries	metis, parmetis
Building procedure	Makefiles
Web site	http://www.bsc.es/computer-applications/alya-system
Licence	Open Source. http://www.prace-ri.eu/ueabs/?lang=en#ALYA

Table 9: Code general features for ALYA.

Main objectives:

Alya is divided into two types of processes: The master and 'Nw' workers. The master is mainly in charge of pre-processing and some of the post-processing tasks. The workers are in charge of running the simulation iterations concurrently, by assembling matrices and RHS (Right-Hand Side), and solving the corresponding algebraic system by way of iterative solvers.

In the initial version of Alya, the mesh partitioning is done sequentially by the master while the workers wait for receiving their parts of the mesh.

The goal of this project is to parallelize the sequential mesh partitioning steps. The approach consists of performing these steps by a subset of the 'Nw' workers in parallel, and to let the

master only be in charge of calculating the communication scheduling at the end of the process. Let us denote a partitioning worker as a worker belonging to the subset of the Nw workers, and participating in the parallel mesh partitioning. Thus we have Np<=Nw workers involved in the partitioning.

Accomplished work:

In this project we have implemented and validated the most complex part of the parallel partitioning. The workflow which has been implemented is the following:

1) (Master) Mesh reading from the hard drive.
2) (Master) Initialise mesh properties variables.
3) (Master) Distributes a consecutive part of the mesh to a partitioning workers subset.
4) (Workers) Each partitioning worker receives a part of the mesh and computes its own elements adjacency graph.
5) (Workers) Each partitioning worker computes the partition of it's part calling the parmetis library in parallel (LEPAR_PAR).
6) (Workers) Each partitioning worker sends to the master his LEPAR_PAR.
7) (Master) Joins all the LEPAR_PAR received from the partitioning workers.
8) (Master) Compute the communication arrays based on the LEPAR_PAR
9) (Master) Compute the communication scheduling.
10) (Master) Distribute the mesh part to its corresponding worker.

Main results:

Partitioning Workers Scalability

In this test we ran a well known Navier-Stokes simulation, with a 30 million elements mesh. We partitioned the mesh in 511 domains using the parallel partition algorithm. We used 2, 5, 10, 15 and 20 partitioning workers in different runs to achieve this task.

We measured the time needed to partition the mesh between the following two points (both included):

- The master distributes a consecutive part of the mesh to a workers subset.
- The master receives and joins all the partitioned mesh.

In this test, we measured the speed-up of the algorithm to partition the mesh in parallel as we added more workers to partition the mesh. The obtained response times are presented in Table 10.

Number of partitioning workers	Response time in seconds
2	252,2
5	118,8
10	76,7
15	62,2
20	57,3

Table 10: 30 million mesh partitioning time in 511 domains.

As can be seen from Figure 10 the code scales when we use more workers to partition the mesh. Running the same partition with the serial version and *metis*, we have obtained a response time of 218,7 seconds. With 5 partitioning workers the response time is 118,8 seconds, that doubles the performance of the serial version.

Parmetis strong scalability

We have measured the Alya scalability running a Navier-Stokes simulation with these inputs:

- 30 million elements cube mesh.
- From 127 to 4095 domains.
- Partitioned in parallel with 8 partitioning workers.
- 2 iteration time steps are measured.

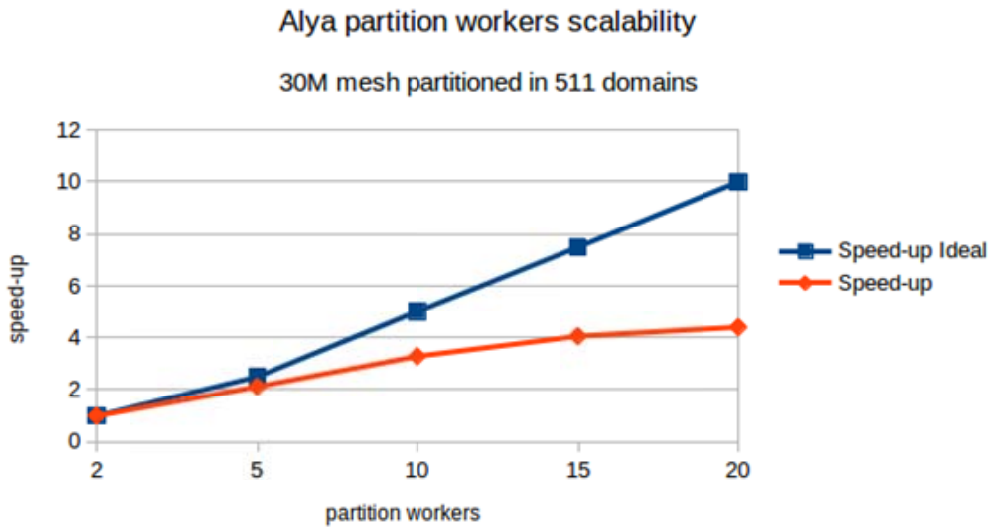

Figure 10: Partition algorithm speed-up when more partitioning workers are used, with a 30 million mesh in 511 domains.

Here we want to measure how well parmetis is doing a balanced partitioning between the domains and what is the scalability obtained with the parallel partition. Figure 11 presents the results.

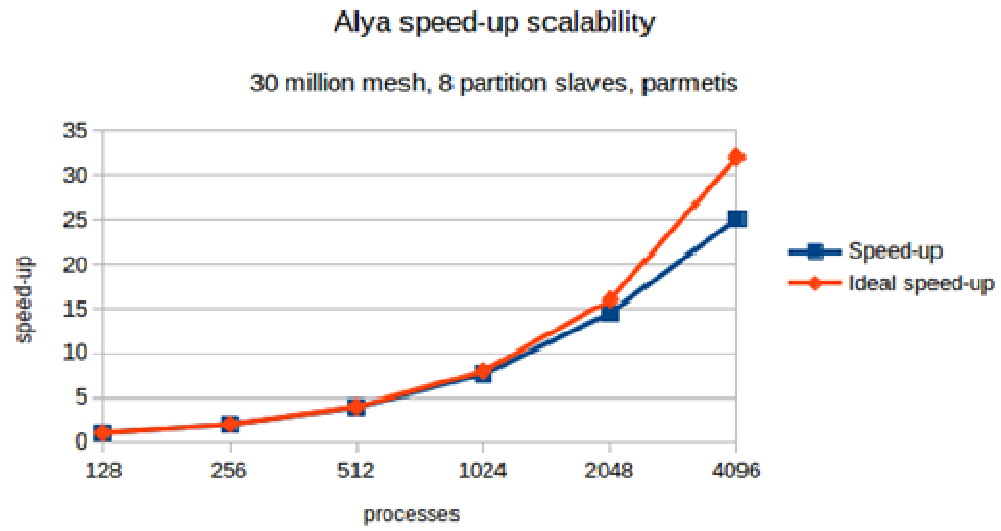

Figure 11: Alya scalability with 8 partitioning workers and parmetis.

As is shown in Figure 11 the Alya scalability with Parmetis is close to ideal. Taking into account that with 4096 domains we have only 8000 elements per domain, the scalability is quite good.

Parmetis and metis elements distribution comparison

We have partitioned the same 30 million mesh in 511 domains, using metis in serial and parmetis in parallel with 20 and 40 partitioning workers. If we plot the histogram of the elements distribution between the domains we can see that metis is doing a better job (Figure 12).

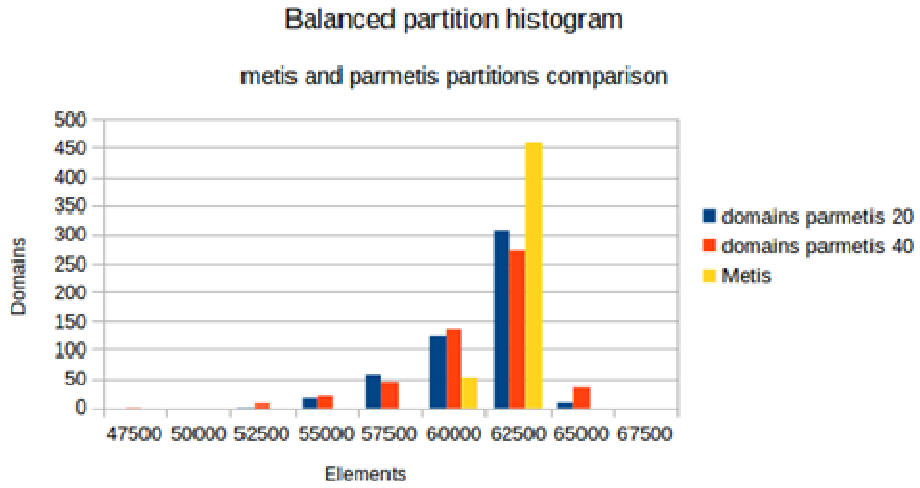

Figure 12: Parmetis and metis distribution between domains histogram. It shows how elements are distributed among the domains.

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

2.8.2 *High-order method for a new generation of large eddy simulation solver, 2010PA2194*

Code general features

Name	JAGUAR
Scientific field	HPC, CFD
Short code description	JAGUAR is a new CFD solver. It uses high-order method (Spectral Differences) and it is dedicated to LES computations. The idea is to have a polynomial per cell, so this way there is a lot of degrees of freedom in each cell (depending on the polynomial order). The code manages both structured and unstructured meshes.
Programming language	Fortran 90
Supported compilers	GNU, Intel, PGI
Parallel implementation	MPI, OpenMP, CUDA
Accelerator support	Yes, with GPUs
Libraries	CGNS, GMSH
Building procedure	Make
Web site	http://www.cerfacs.fr/~puigt/jaguar.html
Licence	Private

Table 11: Code general features for JAGUAR.

Targets and accomplished work

We have enabled hybrid OpenMP/MPI computations on a parallel high-order method (Spectral Differences) applied in a Computational Fluid Dynamic code. The code is written in Fortran 90 with MPI library and OpenMP directives for the parallelization. This work focuses on the performance achieved with the OpenMP shared memory model in a well spread environment (bi-socket nodes and multi-core x86 processors). This was done in order to reduce the number of MPI communications with a large number of cores. We compared the different approaches: full MPI versus full OpenMP computations, full MPI versus hybrid OpenMP/MPI computations, etc. We observed that hybrid and full MPI computations took nearly the same time for a small number of cores.

Main objectives:

The main objective was to enable hybrid computations with OpenMP and MPI because the initial implementation was not satisfying. After that the second objective was to run the code on a large number of cores in order to detect when hybrid computations are more efficient than full MPI computations.

Accomplished work:

We succeeded in enabling hybrid computations. We modified the solver part of the code in order to reduce the number of threads synchronisations and to increase threads independence.

We also worked on threads binding and processes pinning on NUMA environment because we ran the code on CURIE thin nodes (2 sockets per node).

Main results:

Figure 13 shows the improvements made in the new version of the code compared to the initial one with full OpenMP (no MPI).

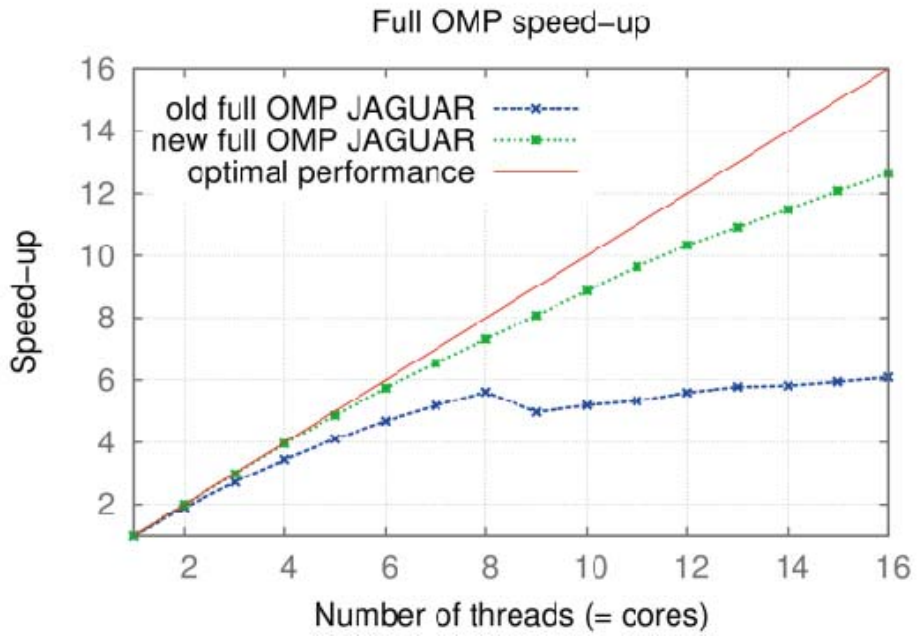

Figure 13: Scaling curve for the OMP version of JAGUAR.

We can clearly see that the achieved speed-up is significantly better with the new version of the code: for 16 cores we achieved a 6-fold speed-up with the old version of the code and a nearly 13-fold speed-up with the new version.

Now if we take a look at the hybrid version of the code (2 processes and 8 threads per processes) we can see that the achieved speed-up is slightly improved with the hybrid version (14-fold speed-up). Figure 14 shows the results.

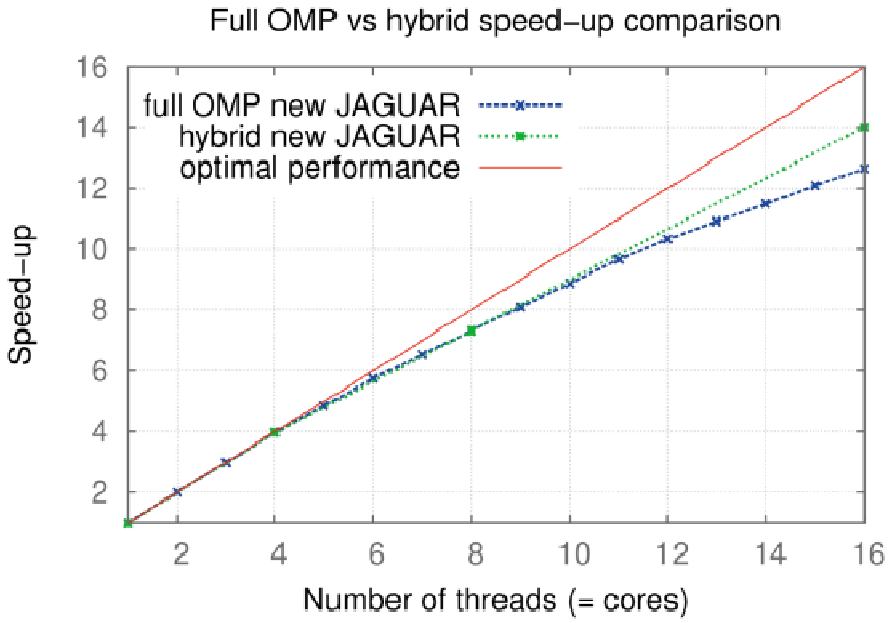

Figure 14: Performance comparison between OMP and hybrid version of JAGUAR.

We also ran the hybrid version of the code on 64 cores and we achieved a 54-fold speed-up. This is not perfect but we used a small mesh (low computations/communications ratio) and we hope that with bigger meshes the scalability would have been better.

Our aim was to run the code on more nodes but we only succeeded too late in enabling efficient OpenMP version of the code and we encountered some difficulties to generate big meshes.

At the end we estimate that we have been reworking 30% of the original code.

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

2.8.3 *Performance of the post-Wannier Berry-phase code for the anomalous Hall conductivity calculations, 2010PA2231*

Code general features

Name	Wannier90
Scientific field	theoretical physics, materials science
Short code description	WANNIER90 is a quantum-mechanics code for the computation of maximally localized Wannier functions, ballistic transport, thermoelectrics and Berry-phase derived properties – such as optical conductivity, orbital magnetization and anomalous Hall conductivity.
Programming language	Fortran
Supported compilers	Intel, PGI
Parallel implementation	MPI only
Accelerator support	No
Libraries	BLAS, Lapack
Building procedure	make
Web site	http://www.wannier.org
Licence	GPL

Table 12: Code general features for Wannier90.

Main objectives:

WANNIER90 [12], [13] consists of two main executables: 1) a serial tool named wannier90.x and 2) a purely MPI-parallelized application, called postw90.x. Based on an initial electronic structure calculation using for example Quantum ESPRESSO [14], *wannier90.x* computes MLWFs which are needed by *postw90.x* for computing the thermoelectric (the BoltzWann code) and Berry-phase derived properties of interest.

Within this workflow, the calculations of postw90.x for Berry-phase derived properties were most problematic regarding wall-clock runtime and total resource consumption. For this reason, the main focus of this project has been the optimization of postw90.x with respect to performance and scalability.

Accomplished work:

All optimizations to postw90.x implemented in the course of this project are based on the integrated performance analysis tool suite HPCToolkit [14].

The performance of core computations, dominated by dense matrix operations, could be increased by a factor of 5 and higher by using BLAS instead of the Fortran built-in *matmul* function or explicit loop constructs for performing runtime-critical matrix products. A further speedup could be achieved by algebraically rearranging matrix multiplications (exploiting associativity) and reusing intermediate results.

Another important improvement was the elimination of a severe bottleneck in the initialization phase. This now makes previously unfeasible computations with more than 64

atoms possible. Besides that, the scalability of postw90.x benefited significantly from this optimization. An additional improvement in scalability has been achieved through the parallelization of two previously serial program modules, *kpath* and *kslice*.

Main results:

We were able to demonstrate near- perfect strong scalability of postw90.x up to 2048 processes for sufficiently large problem settings. When restricting the computations to the program module berry_main, we were able to demonstrate optimal weak-scalability up to 16-thousand processes.

In Figure 15, we give a detailed comparison of the strong scalability behaviour of the original and new version of postw90.x for different test cases with 8 up to 128 atoms. All program modules, berry_main, kpath and kslice have been enabled and representative sampling resolutions have been chosen for each of them. Improvements in performance are mainly due to the usage of BLAS for all performance-critical matrix multiplications. Increased scalability – which is almost optimal up to 2048 processes for large problems – has been achieved mainly through parallelization of the previously serial computation modules kpath and kslice. A different view on the achieved scalability improvements is provided by Figure 16 in terms of relative scaling overhead.

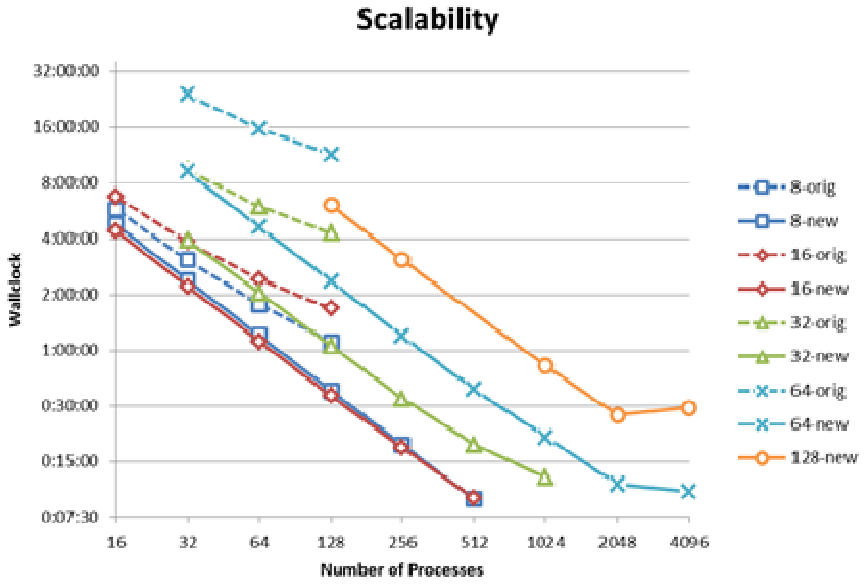

Figure 15: Strong scalability comparison for the original (dashed lines) and the new (solid lines) code version for computations with 8 up to 128 atoms. Performance has been increased significantly; scalability is now optimal up to 2048 processes for large problems.

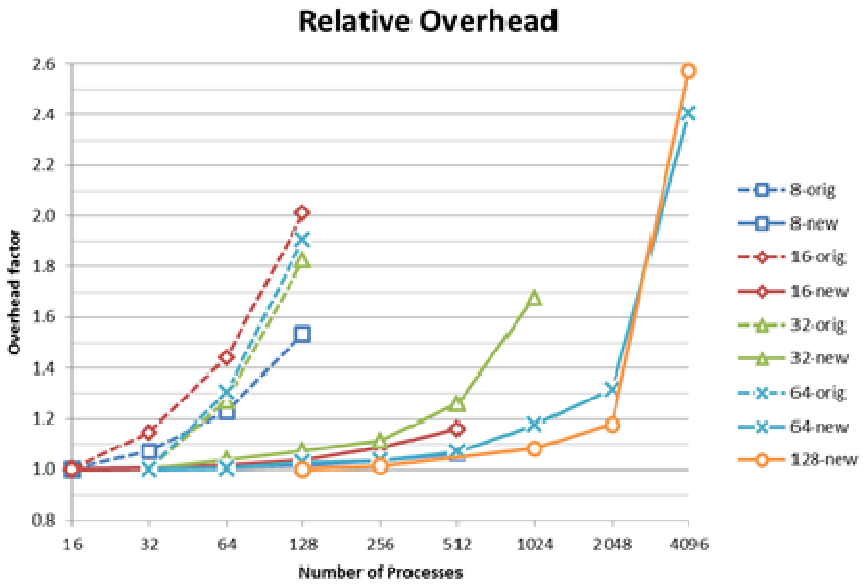

Figure 16: Increase of computational cost relative to the run with smallest number of processes in the test series. The scalability improvement from the original (dashed lines) to the new (solid lines) code version can clearly be seen.

Additional information on all changes to the code-base of WANNIER90 (approximately 6000 out of 43000 lines of code have been touched) can be found online in the GitHub repository of this project [11].

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

2.8.4 *Memory optimization for the Octopus scientific code, 2010PA2216*

Code general features

Name	Octopus (development version)
Scientific field	Chemistry and Materials
Short code description	Octopus is a very efficient code used to study by first principles the properties of the excited states of large biological molecules, complex nanostructures, and solids. Electrons are described quantum-mechanically within density-functional theory (DFT), in its time-dependent form (TDDFT) when doing simulations in time. Nuclei are described classically as point particles. Electron-nucleus interaction is described within the pseudopotential approximation.
Programming language	Fortran
Supported compilers	GNU, INTEL
Parallel implementation	MPI / OpenMP (MPI implementation has been employed for LCAO section of code)
Accelerator support	supports GPU architectures (not used in this project)
Libraries	GSL, LIBXC, LAPACK, ScaLAPACK, METIS, ParMETIS, FFTW, PFFT
Building procedure	Configure script and Makefile
Web site	http://www.tddft.org/programs/octopus/wiki
Licence	GPL 2

Table 13: Code general features for Octopus

Main objectives:

Linear Combination of the Atomic Orbitals (LCAO) is performed as one of the first steps of the ground-state calculation to construct an initial guess for the wave function and to build the Hamiltonian matrix previous to the SCF iterations for a given system. This project focuses on optimization of the LCAO implementation to reduce memory cost and execution time.

Accomplished work:

Within this project matrices that cause of high memory consumption were identified and the behavior of octopus code with different implementations of the LCAO step was investigated.

The memory allocation issue has been solved with a parallel implementation of the LCAO using the external library ScaLAPACK. With this parallel library, large matrices have been distributed among all the MPI processes. Besides, alternative implementations have decreased the execution time of the LCAO step. It has been shown that the memory allocation obstacle can be overcome when using the parallel alternative implementation of LCAO, enabling Octopus to run for larger systems. A performance optimization has also been achieved. Developments in this project will allow bigger simulations to run and, therefore, more interesting systems to be investigated.

Main results:

Extensive profiling of Octopus memory allocation has been performed using the Octopus ad-hoc profiler, which proved to be a very useful tool, due to its capability to provide very detailed information on memory usage and performance. Valgrind Massif heap profiler has been also used.

Two alternative LCAO approaches were tested on MARENOSTRUM III and were compared to native LCAO implementation:

- LCAO is performed by the root MPI process only. After the computation is finished, the eigenvalues and eigenvectors are distributed to the other MPI processes with MPI_Bcast
- LCAO is performed using ScaLAPACK parallel library

Scaling of the application

Results presented in Table 14 have been acquired from benchmark tests performed on 5121 orbitals use case. Octopus initialization and LCAO steps do not demonstrate good scaling behavior.

Number of cores	Hamiltonian / Overlap matrix size (MB)	dpsi matrix size (MB)	Wall clock time (sec)	Speed-up vs the first one	Number of MARENOSTRUM III Nodes
256 LAPACK (native)	200.078	854.85	5216.277	1.000	16
512 LAPACK (native)	200.078	482.655	4938.483	1.056	32
1024 LAPACK (native)	200.078	285.983	4098.483	1.272	64
2048 LAPACK (native)	200.078	166.169	4351.960	1.198	128
alternative implementations					
128 ScaLAPACK	1.875	1472.730	467.293	11.162	8
256 ScaLAPACK	< 1.8	854.85	553.208	9.429	16
256 LAPACK only root MPI process	200.078 (allocated only by root process)	854.85	564.02	9.248	16

Table 14: Walltime and speed-up from benchmark tests performed on 5121 orbitals use case.

Significant performance improvement is observed when applying the alternative LCAO in comparison to the native implementation.

Diagram in Figure 17 demonstrates scaling behaviour of the native LCAO implementation in comparison to the parallel implementation using ScaLAPACK library when the number of orbitals is increased.

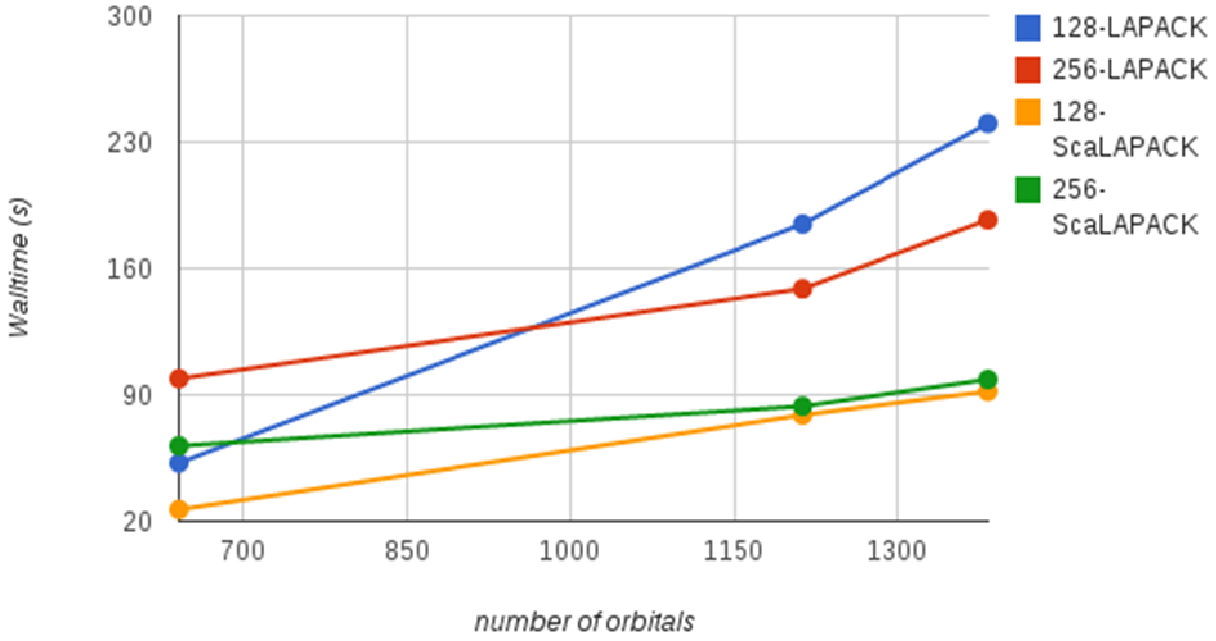

Figure 17: Performance improvement of LCAO ScaLAPACK in comparison to native LCAO implementation.

Memory usage optimization

The total memory consumption per process has been decreased in comparison to native LCAO implementation. LCAO implementations are compared in terms of memory consumption in Figure 18.

As this project aims to optimize the memory usage in order to make running Octopus for many states systems on HPC infrastructures possible, benchmark tests that were performed for this purpose, used a few nodes to extract conclusions from the behaviour of the code when the memory provided is insufficient. This practice is intended to simulate the memory allocation effect on bigger use cases in terms of the number of orbitals.

In Figure 18, in the cases of LCAO native and LCAO ScaLAPACK implementations, memory consumption is measured per process. However, it should be noted that when applying the LCAO LAPACK alternative implementation only the root MPI process stores Hamiltonian and Overlap matrices. Therefore, considering this specific implementation maximum memory allocation displayed in Figure 18, refers only to the root process.

MARENOSTRUM III provides 2896 nodes with 1800 MB of RAM per task. Due to this limitation we were not able to run the native LCAO implementation for 5121 orbitals using 128 processes, as a larger amount of memory would have been required. It can be observed that, although the LCAO LAPACK alternative differs from native implementation and allocates less memory when applied to larger systems, this approach is also severely limited by the maximum memory per task, due to Hamiltonian and Overlap matrices allocation by the root process only. Therefore, the LCAO LAPACK alternative implementation was also not able to run for 5121 orbitals using 128 processes. On the contrary, the LCAO ScaLAPACK implementation achieved to decrease memory allocation per process and as a result this approach allows the study of larger atomic systems.

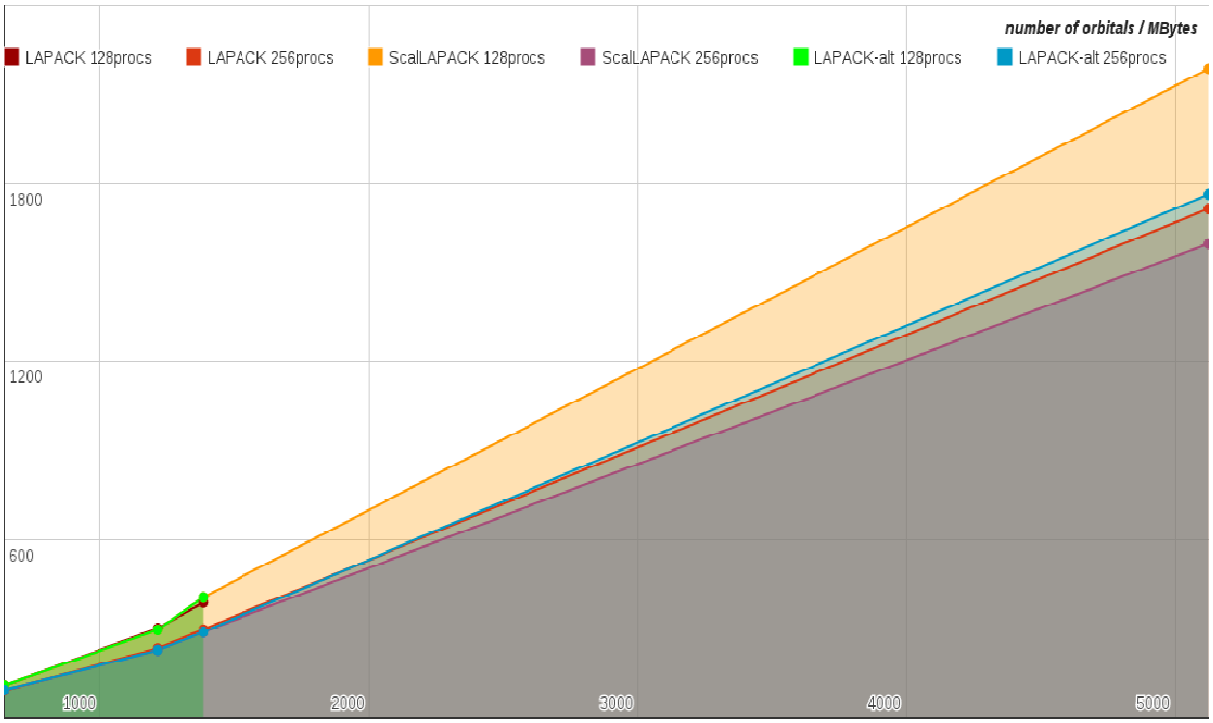

Figure 18: Memory consumption per process for LCAO implementations with varying number of processes and number of orbitals.

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

3 Summary

During the PRACE-3IP extension phase Task 7.1.A successfully performed three Cut-offs for preparatory access including the associated review process and support for the approved projects.

In total 12 Preparatory Access type C projects have been supported by T7.1.A. The timeline of these projects is shown in the Gantt chart in Figure 19. The chart shows the time span of each project. The blue background demarcates the PRACE-3IP extension phase and shows which projects were initiated in PRACE 3IP but concluded within the extension phase. These projects are shown in orange in the Gantt chart. All but one of these projects plan to or have already produced a white paper. Approved white papers are published online on the PRACE RI web page [9]. Table 15 gives an overview of the status of the white papers for all projects.

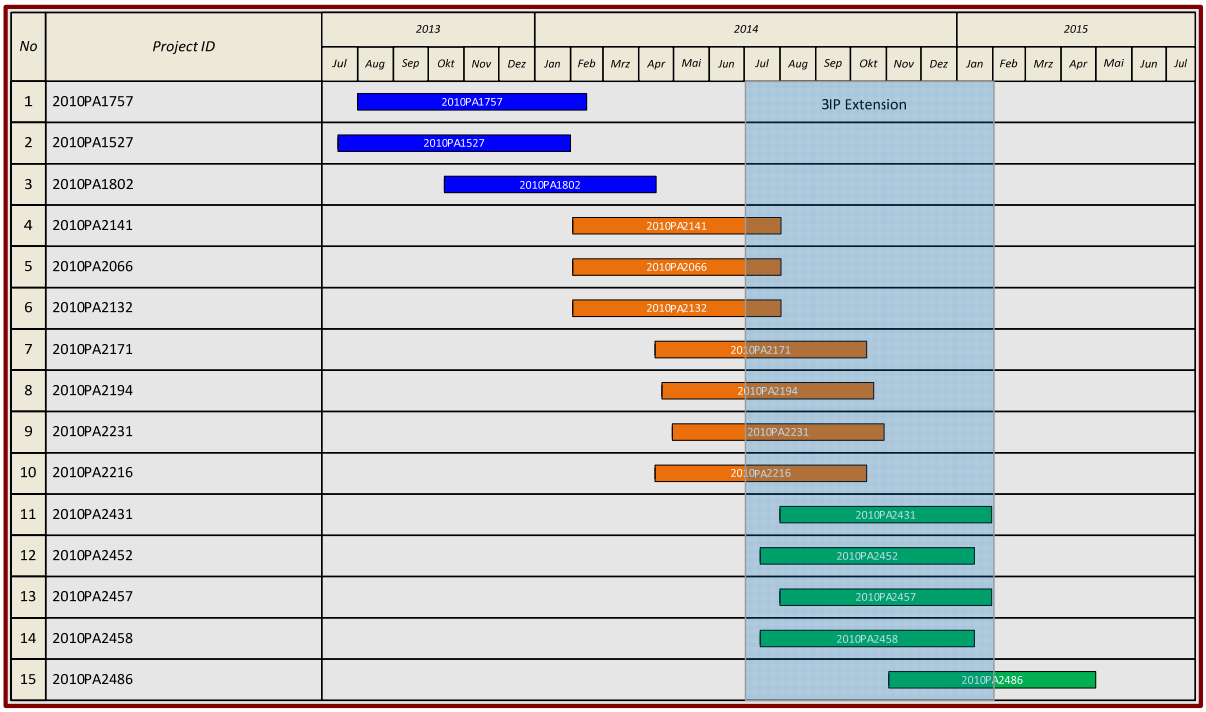

Figure 19: Timeline of the PA C projects.

The green projects in Figure 19 were initialized in the 3IP extension phase. Apart from project 2010PA2486 all of these projects will be finalized during the current extension phase and the creation of reports and white papers is currently in progress. The project from the Cut-off December 2014 is currently being reviewed and therefore does not appear in this deliverable. The projects displayed in blue were initiated and finalized during PRACE-3IP. The outcome of these projects was already reported in deliverable D7.1.2.

The slightly different starting dates of the projects per Cut-off is the result of the decisions made by the hosting members which determine the exact start of the projects at their local site. Additionally, PIs can set the starting date of their projects within a limited time frame.

Project ID	White paper	White paper status
2010PA2141		No white paper produced
2010PA2066	**WP200**: Scalability Limitations of CESM Simulation on JUQUEEN	Published online [3]
2010PA2132	**WP201**: Improving the Scalability	Published online [4]

Project ID	White paper	White paper status
	of the Overlapping Fragments Method Code	
2010PA2171	**WP202**: Parallel Mesh Partitioning in Alya	Published online [4]
2010PA2194	**WP203**: High-order method for a New Generation of Large Eddy Simulation Solver	Published online [6]
2010PA2231	**WP204**: Optimizing the post-Wannier Berry-phase Code for Optical and Anomalous Hall Conductivities and Orbital Magnetization	Published online [7]
2010PA2216	**WP205**: Memory Optimization for the Octopus Scientific Code	Published online [8]
2010PA2431	**WP206**: OpenFOAM capability for industrial large scale computation of the multiphase flow of future automotive component: step 2	In progress
2010PA2452	**WP207**: Numerical modelling of the interaction of light waves with nanostructures using a high order discontinuous finite element method	In progress
2010PA2457	**WP208**: Large scale parallelized 3d mesoscopic simulations of the mechanical response to shear in disordered media	In progress
2010PA2458	**WP209**: PICCANTE: an open source particle-in-cell code for advanced simulations on tier-0 systems	In progress
2010PA2486		Project finishes by the end of April 2015. White paper will subsequently be produced

Table 15: White paper status of the current PA C projects. The colours in the table correspond to the colours chosen for the projects in the Gantt chart given above.

Table 15 shows the success of task 7.1.A as almost all finalized projects published their results or plan to publish it in the near future. The remaining projects coloured green in Table 15 are also expected to produce white papers to make their outcome available to a wider audience.