SEVENTH FRAMEWORK PROGRAMME
Research Infrastructures

INFRA-2011-2.3.5 – Second Implementation Phase of the European High Performance Computing (HPC) service PRACE

PRACE-2IP

PRACE Second Implementation Project

Grant Agreement Number: RI-283493

D8.2
Refactoring and Algorithm Re-engineering Guides and Reports

Version:	1.0
Author(s):	Claudio Gheller, CSCS
Date:	25.04.2013

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: RI-283493**	
	Project Title: PRACE Second Implementation Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: D8.2	
	Deliverable Nature: Report	
	Deliverable Level: PU *	**Contractual Date of Delivery:** 30 / 04 / 2013
		Actual Date of Delivery: 30 / 04 / 2013
	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: Prototype Codes Exploring Performance Improvements**	
	ID: D8.2	
	Version: 1.0	
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D8.2.docx	
Authorship	**Written by:**	Claudio Gheller (CSCS)
	Contributors:	Thomas Schulthess, CSCS; Fabio Affinito, CINECA; Alastair McKinstry; Laurent Crouzet, Marc Torrent, CEA; Andy Sunderland, STFC; Giannis Koutsou, Abdou Abdel-Rehim, CASTORC; Miguel Avillez, UC-LCA; Georg Huhs and Mohammad Jowkar, BSC; Guillaume Houzeaux, BSC; Charles Moulinec, Xiaohu Guo, STFC; Vít Vondrák, David Horák, Václav Hapla, VSB; Peter Raback, Juha Ruokolainen, Mikko Byckling, CSC; Bärbel Große-Wöhrmann, Florian Seybold, HLRS,
	Reviewed by:	Michael Browne, NUI Galway, Dietmar Erwin, FZJ
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	10/02/2013	First skeleton	
0.2	20/02/2013	Introduction written	
0.3	16/03/2013	Section 2 partially written	
0.4	22/03/2013	First projects added	
0.5	27/03/2013	More projects added	
0.6	04/04/0213	More projects added	
0.7	09/04/2013	More projects added	
0.8	10/04/2013	Summary written and remaining projects added	
0.9	11/04/2013	Final version ready for internal revision	
1.0	25/04/2013	Version for Approval	

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure, scientific applications, libraries, performance modelling.

Table of Contents

Project and Deliverable Information Sheet i
Document Control Sheet i
Document Status Sheet ii
Document Keywords iii
Table of Contents iv
List of Figures v
References and Applicable Documents v
List of Acronyms and Abbreviations vi
Executive Summary 1
1 Introduction 1
2 The web infrastructure 4
3 Astrophysics Section 8
3.1 RAMSES 8
3.2 EAF-PAMR 10
3.3 PFARM 11
4 Climate Section 13
4.1 Couplers: OASIS 13
4.2 Input/Output: CDI, XIOS, PIO 14
4.3 Dynamical Cores: ICON 16
4.4 Ocean Models: NEMO and Fluidity-ICOM 17
5 Material Science Section 19
5.1 ABINIT 19
5.2 Quantum ESPRESSO 21
5.3 Siesta 22
5.4 Exciting/ELK 23
6 Particle Physics Section 25
6.1 PLQCD 25
7 Engineering Section 29
7.1 Elmer 29
7.2 Code_Saturne 32
7.3 Alya 33
7.4 ZFS 35
7.5 Coupled FVM Solver 38
8 Summary 39

List of Figures

Figure 1: HPC Forge interface of one of the WP8 projects (ABINIT) 5
Figure 2: Basic structure of the WP8 wiki web site. 6
Figure 3: The wiki Main Page for one of the WP8 projects (RAMSES). 7
Figure 4: RAMSES hydro kernel scalability on different HPC architectures 9
Figure 5: RAMSES conjugate gradients gravitational solver scalability on different HPC architectures 9
Figure 6: Speedup of the EAF-PAMR code in a 3D Sedov Blast test with a 128^3 computational cells mesh, using 1 or 2 NVIDIA GeForce GTX 260 GPUs and 2 GPUs and two cores............................ 11
Figure 7: Repartition of time (%) for the MPI communication before the orthogonalisation of wave-functions. In blue, the waiting time; in red the actual communication time. The first graph shows the status before the present work. The second graph shows the status after the improvement of load balancing due to band distribution only. The third graph shows the status after the full improvement of load balancing. Test case: 107 gold atoms on TGCC-Curie Prace computer. 20
Figure 8 Profiling of QuantumESPRESSO CP module for a WAT64 system with (above) and without (below) ELPA. 21
Figure 9: Strong scaling results of the Hopping Matrix on Cray XE6 26
Figure 10: Linear solver time for Incremental Eig-CG as compared to CG for each Right-Hand Side.27
Figure 11: Scaling of the Fourier accelerated Landau gauge fixing.. 27
Figure 12: Acceptance rates for the Hybrid Monte-Carlo with optimized integrator. 27

References and Applicable Documents

[1] http://www.prace-ri.eu
[2] Deliverable D8.1.1: “Community Codes Development Proposal”
[3] Deliverable D8.1.2: “Performance Model of Community Codes”
[4] Deliverable D8.1.4: “Plan for Community Code Refactoring”
[5] Bridging Performance Analysis Tools and Analytic Performance Modelling for HPC, T. Hoefler, Proceedings of Workshop on Productivity and Performance (PROPER 2010), Springer, Dec. 2010.
[6] A Framework for Performance Modelling and Prediction. Allan Snavely , Laura Carrington , Nicole Wolter , Jesus Labarta, Rosa Badia , Avi Purkayastha, Proceedings of the 2002 ACM/IEEE conference on Supercomputing.
[7] Performance Modelling: Understanding the Present and Predicting the Future. Bailey, David H.; Snavely, Allan. http://escholarship.org/uc/item/1jp3949m
[8] How Well Can Simple Metrics Represent the Performance of HPC Applications? Laura C. Carrington, Michael Laurenzano, Allan Snavely, Roy L. Campbell, Larry P. Davis; Proceedings of the 2005 ACM/IEEE conference on Supercomputing, 2005, IEEE Computer Society
[9] http://fusionforge.org/
[10] http://wiki.org/
[11] http://www.mediawiki.org/wiki/MediaWiki
[12] http://en.wikipedia.org/wiki/Main_Page
[13] http://wikimediafoundation.org/wiki/Home
[14] A coupled finite volume solver for the solution of incompressible flows on unstructured grids, M. Darwish et al., Journal of Computational Physics 228 (2009) p. 180-201
[15] A coupled finite volume solver for the solution of laminar/turbulent incompressible and compressible flows, L. Mangani, C. Bianchini, 5th OpenFOAM® Workshop, June 22-24, 2010, Gothenburg, Sweden
[16] A fully integrated Coupled Solver using Block-GAMG acceleration, L. Mangani et al., 7th OpenFOAM® Workshop, June 25-28, 2012, Darmstadt, Germany

[17] Block-Coupled Simulations Using OpenFOAM, I. Clifford, 6th OpenFOAM® Workshop, June 13-16, 2011
[18] http://www.openacc-standard.org/
[19] http://subversion.apache.org/
[20] Conservative Regridding When Grid Cell Edges Are Unknown -- Case of SCRIP, J. Chavas et al., Technical Report, CEA, 2013 . http://arxiv.org/abs/1302.1796
[21] http://www.hector.ac.uk/cse/distributedcse/reports/prmat/.
[22] http://elpa.rzg.mpg.de
[23] http://icl.cs.utk.edu/plasma/
[24] http://icl.cs.utk.edu/magma/
[25] http://www.khronos.org/opencl/
[26] http://aces.snu.ac.kr/Center_for_Manycore_Programming/SnuCL.html

List of Acronyms and Abbreviations

AMR	Adaptive Mesh Refinement
API	Application Programming Interface
BLAS	Basic Linear Algebra Subprograms
BSC	Barcelona Supercomputing Center (Spain)
CAF	Co-Array Fortran
CCLM	COSMO Climate Limited-area Model
ccNUMA	cache coherent NUMA
CEA	Commissariat à l'Energie Atomique (represented in PRACE by GENCI, France)
CERFACS	The European Centre for Research and Advanced Training in Scientific Computation
CESM	Community Earth System Model, developed at NCAR (USA)
CFD	Computational Fluid Dynamics
CG	Conjugate-Gradient
CINECA	Consorzio Interuniversitario per il Calcolo Parallello (Italy)
CINES	Centre Informatique National de l'Enseignement Supérieur (represented in PRACE by GENCI, France)
CNRS	Centre national de la recherche scientifique
COSMO	Consortium for Small-scale Modelling
CP	Car-Parrinello
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 (NVIDIA)
CUSP	CUda SParse linear algebra library
DFPT	Density-Functional Perturbation Theory
DFT	Discrete Fourier Transform
DGEMM	Double precision General Matrix Multiply
DKRZ	Deutsches Klimarechenzentum
DP	Double Precision, usually 64-bit floating-point numbers
DRAM	Dynamic Random Access memory
EC	European Community
ENES	European Network for Earth System Modelling
EPCC	Edinburgh Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)

EPSRC	The Engineering and Physical Sciences Research Council (United Kingdom)
ESM	Earth System Model
ETHZ	Eidgenössische Technische Hochschule Zürich, ETH Zurich (Switzerland)
FFT	Fast Fourier Transform
FP	Floating-Point
FPGA	Field Programmable Gate Array
FPU	Floating-Point Unit
FT-MPI	Fault Tolerant Message Passing Interface
FZJ	Forschungszentrum Jülich (Germany)
GB	Giga (= 2^{30} ~ 10^9) Bytes (= 8 bits), also GByte
Gb/s	Giga (= 10^9) bits per second, also Gbit/s
GB/s	Giga (= 10^9) Bytes (= 8 bits) per second, also GByte/s
GCS	Gauss Centre for Supercomputing (Germany)
GENCI	Grand Equipement National de Calcul Intensif (France)
GFlop/s	Giga (= 10^9) Floating-point operations (usually in 64-bit, i.e., DP) per second, also GF/s
GGA	Generalised Gradient Approximations
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
GPL	GNU General Public Licence
GPU	Graphic Processing Unit
HDD	Hard Disk Drive
HLRS	High Performance Computing Center Stuttgart (Germany)
HMPP	Hybrid Multi-core Parallel Programming (CAPS enterprise)
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
HPL	High Performance LINPACK
ICHEC	Irish Centre for High-End Computing
ICOM	Imperial College Ocean Model
ICON	Icosahedral Non-hydrostatic model
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
I/O	Input/Output
IPSL	Institut Pierre Simon Laplace
JSC	Jülich Supercomputing Centre (FZJ, Germany)
KB	Kilo (= 2^{10} ~10^3) Bytes (= 8 bits), also KByte
LBE	Lattice Boltzmann Equation
LINPACK	Software library for Linear Algebra
LQCD	Lattice QCD
LRZ	Leibniz Supercomputing Centre (Garching, Germany)
MB	Mega (= 2^{20} ~ 10^6) Bytes (= 8 bits), also MByte
MB/s	Mega (= 10^6) Bytes (= 8 bits) per second, also MByte/s
MBPT	Many-Body Perturbation Theory
MCT	Model Coupling Toolkit, developed at Argonne National Lab. (USA)
MD	Molecular Dynamics

MFlop/s	Mega (= 10^6) Floating-point operations (usually in 64-bit, i.e., DP) per second, also MF/s
MHz	Mega (= 10^6) Hertz, frequency =10^6 periods or clock cycles per second
MIPS	Originally Microprocessor without Interlocked Pipeline Stages; a RISC processor architecture developed by MIPS Technology
MKL	Math Kernel Library (Intel)
MPI	Message Passing Interface
MPI-IO	Message Passing Interface – Input/Output
MPP	Massively Parallel Processing (or Processor)
MPT	Message Passing Toolkit
NCAR	National Center for Atmospheric Research
NCF	Netherlands Computing Facilities (Netherlands)
NEGF	non-equilibrium Green's functions,
NERC	Natural Environment Research Council
NEMO	Nucleus for European Modelling of the Ocean
NERC	Natural Environment Research Council (United Kingdom)
NWP	Numerical Weather Prediction
OpenCL	Open Computing Language
OpenMP	Open Multi-Processing
OS	Operating System
PAW	Projector Augmented-Wave
PGI	Portland Group, Inc.
PGAS	Partitioned Global Address Space
PIMD	Path-Integral Molecular Dynamics
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)
PWscf	Plane-Wave Self-Consistent Field
QCD	Quantum Chromodynamics
QR	QR method or algorithm: a procedure in linear algebra to factorise a matrix into a product of an orthogonal and an upper triangular matrix
RAM	Random Access Memory
RDMA	Remote Data Memory Access
RISC	Reduce Instruction Set Computer
RPM	Revolution per Minute
SGEMM	Single precision General Matrix Multiply, subroutine in the BLAS
SHMEM	Share Memory access library (Cray)
SIMD	Single Instruction Multiple Data
SM	Streaming Multiprocessor, also Subnet Manager
SMP	Symmetric MultiProcessing
SP	Single Precision, usually 32-bit floating-point numbers
STFC	Science and Technology Facilities Council (represented in PRACE by EPSRC, United Kingdom)
STRATOS	PRACE advisory group for STRAtegic TechnOlogieS
TB	Tera (=2^{40} ~ 10^{12}) Bytes (= 8 bits), also TByte
TDDFT	Time-dependent density functional theory
TFlop/s	Tera (=10^{12}) Floating-point operations (usually in 64-bit, i.e., DP) per second, also TF/s

Tier-0	Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1
UML	Unified Modelling Language
UPC	Unified Parallel C
VSB	Technical University of Ostrava (Czech Republic)

Executive Summary

The present document provides a snapshot of the contents of the PRACE-2IP work package 8 (hereafter WP8) web site (http://prace2ip-wp8.hpcforge.org) at the time of submission. The WP8 web site has been implemented to keep track of and publish the work accomplished in the work package in terms of re-design and refactoring of numerical codes, documentation, user guides, and publications. This was needed in order to deploy an effective instrument both to support and optimise collaborative work among the different groups involved in the development of the various applications, and to give to the scientific community an effective way to gain insight into the details of the on-going work, to the achieved results and to access the released software.

The first part of the document provides technical details on the web infrastructure, introducing the basic adopted technologies, and describes its design and organisation. The FusionForge and the MediaWiki software represent the main tools exploited for creating the web site, proving to be specifically suitable to develop the intended web platform.

In the second part, the status of the code refactoring work carried out in WP8 is presented. In the work package a number of applications, designed to perform large-scale numerical simulations in various scientific areas, namely Astrophysics, Material Science, Climate, Particle Physics and Engineering, are being re-designed and re-implemented in order to efficiently and effectively exploit the coming generation of supercomputing architectures. The work accomplished on each code is summarised and a few preliminary results shown, providing guidelines to the contents of the web site, where all the details are available.

Finally, an outlook toward the next (and final) steps of the work package, is given, summarising the remaining activities. It is worth stressing how the adopted working methodology (based on a close collaboration between researchers and computational scientists) has allowed an early initiation of the process of validation and reintroduction of the codes in the users' community, originally expected for the last months of WP8. A few of these codes (e.g. ABINIT, Quantum ESPRESSO or RAMSES) are even already adopted for current scientific production.

1 Introduction

The overall objective of WP8 is to initiate a sustainable programme in application development for supercomputing applications targeted at problems of high scientific impact that require HPC for their solution.

This is being accomplished by involving a number of scientific communities in the process of enabling some of their most relevant simulation codes to innovative supercomputing architectures. Such process relies on a close synergy between scientists, code developers and HPC experts, the first two contributing with their deep comprehension of the research subjects and of the algorithms, the latter providing the necessary skills and competencies on novel hardware architectures, programming models and software solutions and driving the refactoring programme in order to optimally map applications to coming supercomputing architectures.

The final step of this process consists of the reintegration of the re-implemented codes into the existing applications communities, so that the cycle can close with an effective deployment of the achieved results (in terms of software, but also of documentation, tests and benchmarks and expertise) to the end-users.

In order to successfully develop this programme, suitable instruments have to be used, supporting algorithmic analysis, code evaluation and development, sharing of knowledge, experiences, successes and failures, publication and deployment of the results.

During the first six months of WP8 a number of codes have been analysed, their performance evaluated, their suitability to the refactoring programme investigated, adopting the performance modelling approach ([3], [5], [6], [7], [8]). This methodology relies on the analytic modelling of the main algorithms (characterising the dependency from the critical model parameters) and on the performance analysis, using state of the art performance tools. The performance modelling approach allows studying the current behaviour of a code, emphasising performance and bottlenecks. But it is also a predictive tool, allowing the estimation of the code's behaviour on different computing architectures, and identifying the most promising areas for performance improvement. This methodology is cross-domain, codes independent and provides an objective and quantitative way of evaluating applications.

As a result of the performance modelling, the following codes were identified for re-design and refactoring ([2], [4]):

Code name	**Scientific Domain**	**Responsible partner**
RAMSES	Astrophysics	ETH
PFARM	Astrophysics	STFC
EAF-PAMR	Astrophysics	UC-LCA
OASIS	Climate	CEA
I/O Services	Climate	ICHEC
ICON	Climate	ETH
NEMO	Climate	STFC
Fluidity/ICOM	Climate	STFC
ABINIT	Material Science	CEA
QuantumESPRESSO	Material Science	CINECA
SIESTA	Material Science	BSC
EXCITING/ELK	Material Science	ETH
PLQCD	Particle Physics	CASTORC
ELMER	Engineering	VSB-TUO
CODE_SATURNE	Engineering	STFC
ALYA	Engineering	BSC
ZFS	Engineering	HLRS
Coupled FVM Solver	Engineering	HLRS

Table 1: List of the codes selected for refactoring (left column), corresponding scientific domain (central column) and responsible PRACE partner (right column).

Specific groups were formed to work on each of the selected codes. Due to the different features, needs and targets of the codes, each group could adopt a different working methodology, strategy and schedule in order to reach the envisaged objectives in the WP8 time frame (see [4] for details).

In order to keep track of the software refactoring and algorithm re-engineering activities, to share information and experiences, to collect, organise and publish documentation and to exchange and distribute code, a specific web infrastructure was set up, based on the FusionForge [9] and wiki [10] technologies.

The basic requirements in the design of such web infrastructure were the effectiveness in managing and publishing information, suitability to the expected contents, and sustainability. This last requirement is necessary to preserve the web site, its contents and its accessibility beyond the work package duration, in order to be a stable reference for the involved communities.

Design and Technical details on the web infrastructure will be given in the Section 2. The following sections will summarise the status of the work and the content of the web site for each of the applications under development at the time of the submission of the present deliverable. More specifically, Section 3 is dedicated to the three codes, RAMSES, EAF-PAMR and PFARM, developed in the astrophysical domain, while, in Section 4, we focus on the Climate domain, where the work is split between the refactoring of widely adopted simulation codes, like OASIS, ICON, NEMO and Fluidity-ICOM and the implementation of common high performance "I/O services", developed within the ENES community, for use by all climate models. In Section 5, we present the work on Material Science codes. In this area, most of the effort was dedicated to the optimisation of linear algebra problems, crucial for the four codes under investigation, namely two big community packages like ABINIT and Quantum ESPRESSO and two more specific simulation codes, like SIESTA and Exciting/ELK. Section 6 is devoted to PLQCD, developed in the domain of quantum chromo-dynamics. Here, the work was focused on the refactoring and optimisation of the main computational kernels, encompassing a number of different computational challenges. Finally, Section 7 presents applications in the Engineering area. In this area the main focus was on parallel mesh generation and solver scalability in a number of codes, namely Elmer, Code_Saturn, Alya, ZFS and Finite Volume solvers. Eventually, Section 8 summaries the content of the present document and the next steps.

2 The web infrastructure

In order to deploy an effective web infrastructure, the following requirements had to be taken into consideration:

1. It must be suitable to manage the expected contents expected, namely, source codes, documentation, different media files (like pdfs, images, movie);
2. It must support collaborative work, so that all the groups and the code developers can contribute contents;
3. It must support different levels of access, in order to differentiate roles and permissions (read-only or read and write access);
4. It must be based on solid and mature technologies, easy to maintain in order to ensure longevity.

Such requirements are fulfilled by the FusionForge platform [9]. FusionForge is a software package for collaborative development for the software community. It provides a fully configured development system with versioning, a project web site and tools for communication between members of a development team. The tools provided by FusionForge allow team members to communicate and organise their work; this allows the creation of a knowledge base. It is a free software application arising from the Forge web-based project-management and collaboration software, originally created for running the SourceForge.net platform. FusionForge is licensed under the GNU General Public License.

A specific instance of the FusionForge software, called HPC Forge (https://hpcforge.org/) was used to build the web site for WP8. This instance is already configured for other projects related to high performance computing. It is managed and maintained by CSCS, who also uses it extensively for other internal, national and international projects. This ensures the preservation of the web site and the accessibility to its contents also beyond the end of the work package.

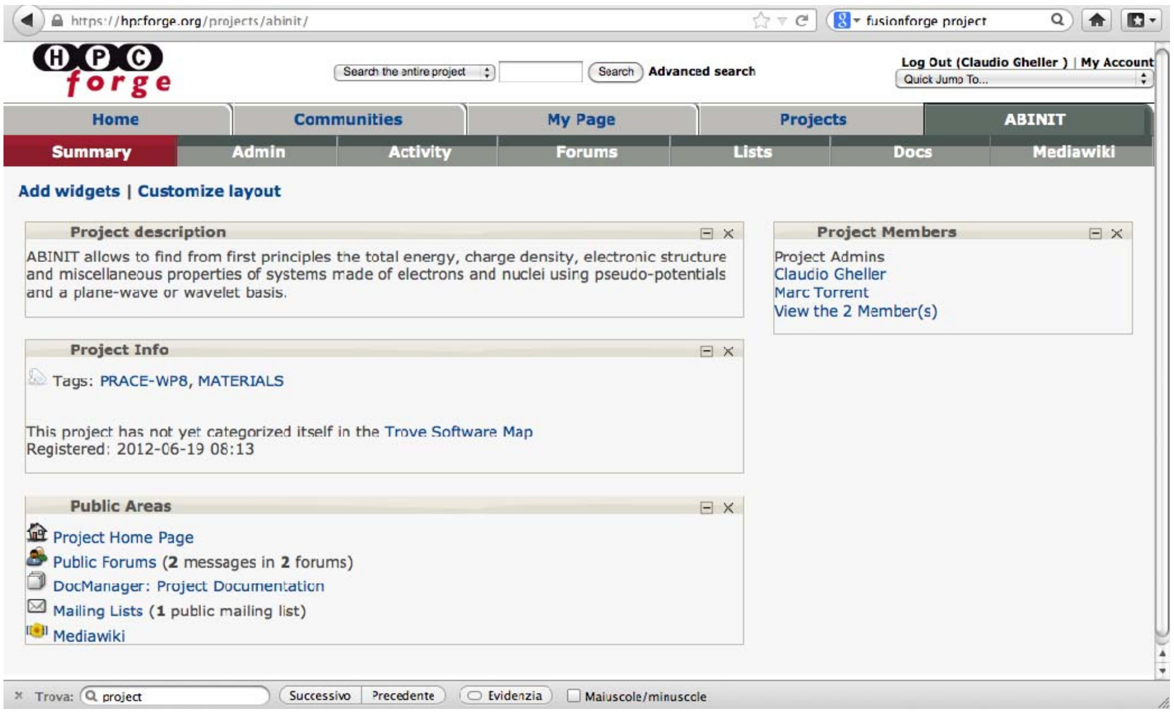
In the web site, we have extensively used *wiki* pages [10]. A wiki is a website which allows its users to collaboratively add, modify, or delete its content via a web browser using a simplified mark-up language or a rich-text editor. FusionForge wiki is powered by MediaWiki technology [11]. MediaWiki is a free software open source wiki package written in PHP, originally for use on Wikipedia [12]. It is now also used by several other projects of the non-profit Wikimedia Foundation [13] and by many other wikis.

Other features, like versioning tools, are exploited by several groups. However, for most of the codes, such utilities were already deployed by the corresponding communities pre-dating WP8. In these cases, we simply use these tools, linking them from the WP8 web site.

The web site design

Based on the above technology, the WP8 web site was designed with the following structure.

Each code developed in the work package is represented in the web site by a ***specific instance*** of a **FusionForge** ***project*** (notice that in this section the term *project* will always refer to a *FusonForge project*). The ***WP8 project*** was created as the main entry point to the web site. Each project has its private and public components. The private part (https://hpcforge.org/) is accessible only by the project administrators and it is essentially used to configure and manage the public component and to collect and store documents with restricted access.



Project description

ABINIT allows to find from first principles the total energy, charge density, electronic structure and miscellaneous properties of systems made of electrons and nuclei using pseudo-potentials and a plane-wave or wavelet basis.

Project Members

Project Admins
Claudio Gheller
Marc Torrent
View the 2 Member(s)

Project Info

Tags: PRACE-WP8, MATERIALS

This project has not yet categorized itself in the Trove Software Map
Registered: 2012-06-19 08:13

Public Areas

Project Home Page
Public Forums (2 messages in 2 forums)
DocManager: Project Documentation
Mailing Lists (1 public mailing list)
Mediawiki

Figure 1: HPC Forge interface of one of the WP8 projects (ABINIT)

The architecture of the private component can be customised, but, in general, the default structure provided by HPC Forge proved to be suitable to the needs of the developers. It is organised in different sections (the corresponding user interface is shown in Figure 1):

- Summary, that provides and overview of the project and allows customising the layout and adding or removing widgets.
- Admin, that allows configuring all the low-level details of the web site, to define users, roles and permissions, to activate/deactivate available tools, like forums, mailing lists, wiki pages etc.
- Activity, Forums, Lists and Docs, which provide tools to organise collaborative work and documents exchange. These tools are seldom used, being partially replaced by corresponding functionalities implemented in the wiki part.
- MediaWiki, which links to the wiki component.

All the WP8 projects are characterised by two tags, which allow the HPC Forge user to query them. The first tag is PRACE-WP8, which identifies all the projects related to the work package. The second is the scientific areas identifier, which can have the following values: MATERIALS, ENGINEERING, CLIMATE, ASTRO, QCD. This tag allows grouping codes from the same scientific domain.

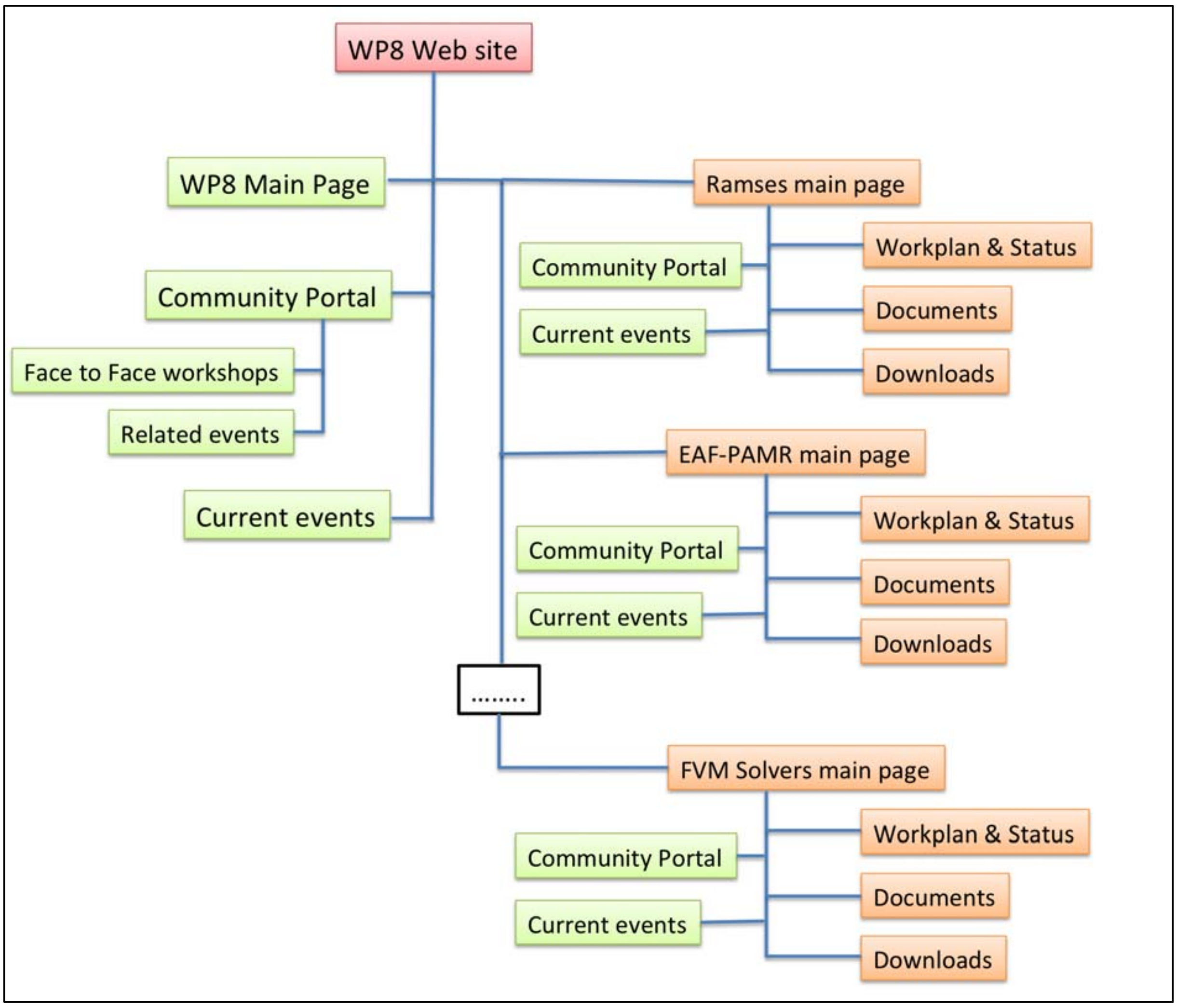
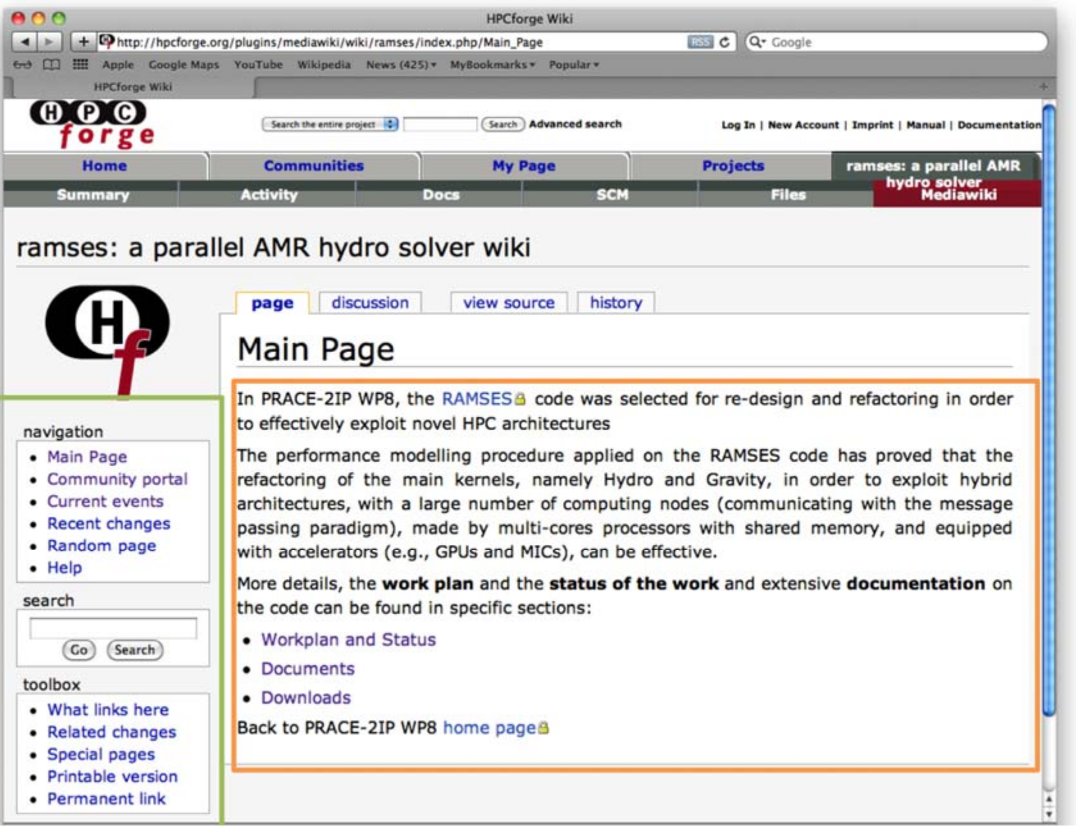


Figure 2: Basic structure of the WP8 wiki web site.

The public component, based on MediaWiki, is accessible, at least read-only, by anybody (http://prace2ip-wp8.hpcforge.org). Differently from the private component, which has a "flat" organisation of the projects, the wiki part was created with a simple hierarchical structure. This was necessary due to the intrinsic nature of the wiki, which tends to create loose and intricate web pages collections, linking to each other with no specific logic.

The diagram of the WP8 wiki site structure is shown in Figure 2 In the figure, the green boxes represent pages that can be reached from the links in the left frame of the wiki page (corresponding to the green box in Figure 3), while orange boxes can be reached from links embedded the "Main Page" (orange box in Figure 3).

Each project has a "Main Page", that is its main entry point. The Main Page of the *WP8 project* is also the Home Page of the web site. Each project is then characterised by a "Community Portal", containing general information useful for the users of the specific community the project refers to, and a "Current Events" page, with information about workshops, conferences and other events relevant for the community and for activities that are going to happen. For instance, for the *WP8 project*, "Community portal" briefly describes the basic ideas of the work package and lists the past main events related to WP8 (linking to the related web pages).


HPCforge Wiki

http://hpcforge.org/plugins/mediawiki/wiki/ramses/index.php/Main_Page

Apple Google Maps YouTube Wikipedia News (425) MyBookmarks Popular

HPCforge Wiki

Search the entire project | Search | Advanced search

Log In | New Account | Imprint | Manual | Documentation

Home | Communities | My Page | Projects | ramses: a parallel AMR hydro solver

Summary | Activity | Docs | SCM | Files | Mediawiki

ramses: a parallel AMR hydro solver wiki

page | discussion | view source | history

Main Page

In PRACE-2IP WP8, the RAMSES code was selected for re-design and refactoring in order to effectively exploit novel HPC architectures

The performance modelling procedure applied on the RAMSES code has proved that the refactoring of the main kernels, namely Hydro and Gravity, in order to exploit hybrid architectures, with a large number of computing nodes (communicating with the message passing paradigm), made by multi-cores processors with shared memory, and equipped with accelerators (e.g., GPUs and MICs), can be effective.

More details, the **work plan** and the **status of the work** and extensive **documentation** on the code can be found in specific sections:

- Workplan and Status
- Documents
- Downloads

Back to PRACE-2IP WP8 home page

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go | Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Figure 3: The wiki Main Page for one of the WP8 projects (RAMSES).

Project's Main Page, Community Portal and Current Events are always reachable from within the project, being statically linked in the menu on the left frame of the wiki page.

From the Main Page of a project, the following three sections can be reached:

- Work plan & Status, where the WP8's work plan of code development and refactoring is summarized and the current status of the work is stated.
- Documents, where papers, white papers, user guides and any other document relevant for the code are collected.
- Downloads, where developed codes are published (sometimes in the form of links to official public repositories).

All these pages can contain links to other wiki internal web pages or to external web sites. Any further level of organisation is left to the needs of the developers.

As a final remark, it must be stressed that the previously described project structure was not imposed as mandatory and several projects preferred to slightly change and adapt it to the needs and features of the specific work plan and numerical code. Hence, a few deviations from the described basic design can be found. However, they do not compromise the accessibility and usefulness of the web site.

3 Astrophysics Section

This section describes the on-going work and the achieved results in the Astrophysics domain on RAMSES, a code for cosmological and galaxy evolution simulations, EAF-PAMR, describing the magnetohydrodynamic behaviour of the interstellar medium and PFARM, a suite of programs for the calculation of electrons-atoms interactions in the inter and intra-galactic gas.

3.1 RAMSES

RAMSES is an adaptive mesh refinement (AMR) multi-species code, describing the behaviour of both the baryonic component, represented as a fluid on the cells of the AMR mesh, and the dark matter, represented as a set of collisionless particles. The two matter components interact via gravitational forces. The AMR approach makes it possible to get high spatial resolution only where this is actually required, thus ensuring a minimal memory usage and computational effort.

The performance modelling procedure applied to the RAMSES code proved that the refactoring of the main kernels, namely Hydro and Gravity, in order to exploit hybrid architectures, with a large number of computing nodes (communicating via the message passing paradigm) made by multi-cores processors with shared memory and equipped with accelerators (e.g., GPUs and Intel Xeon Phi), can be effective. The Hydro and the Gravity kernels can strongly benefit from shared memory nodes both for the performance and because the large available memory which can be optimally used, avoiding demanding data replication. Such replication can limit the maximum size of the simulated problem on distributed systems with a small amount of memory per core. Furthermore, the usage of MPI message passing is limited to internode communication, allowing efficient local data access.

In order to exploit shared memory, the Hydro and Gravity kernels were re-implemented with a hybrid OpenMP+MPI approach. Special care was devoted to managing the access to shared memory. The OpenMP implementation has addressed the following:

- Preserving data locality also inside the shared memory
- Handling effectively concurrent data access

The code kernels that have been parallelised with OpenMP are:

- The hydro kernel
- The conjugate gradient gravitational solver
- The multigrid gravitational solver

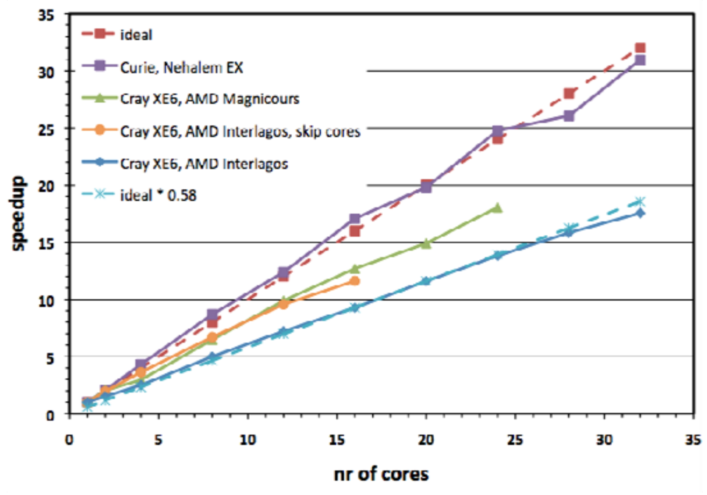


Figure 4: RAMSES hydro kernel scalability on different HPC architectures

The obtained performance depends on the algorithm. Particularly crucial proved to be the memory access patterns. The hydro kernel, thanks to its data memory locality, proved to be highly scalable up to the entire size of the shared memory node (in the cases presented here, 32 cores – see Figure 4). For the conjugate gradient solver, scalability strongly depends on the available hardware (Figure 5), while for the multigrid solver results are still unsatisfactory and more work is still needed to obtain acceptable performance.

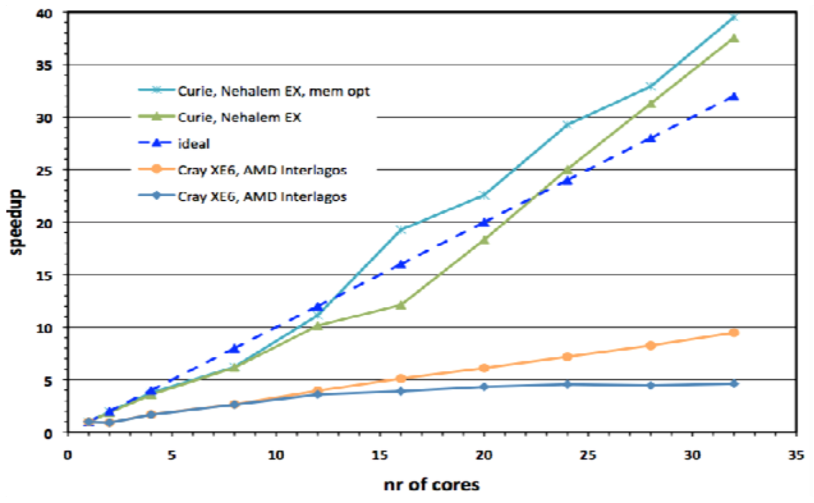


Figure 5: RAMSES conjugate gradients gravitational solver scalability on different HPC architectures

GPUs can be effectively used thanks to the intrinsic data parallelism of the hydrodynamics algorithms. This is effected by the peculiar data structure adopted by RAMSES, that leads to a memory intensive activity, which penalises the accelerator's throughput. Nevertheless, the performance model predicts relevant benefits in using the GPUs.

The Gravity kernel turned out to be not particularly suitable to the GPU architecture. Possible solutions can be designed, but they require deep changes of the algorithm, that are probably beyond the scope of the current project. They will be considered only if time and resources permit.

The implementation of the GPU version of the hydro kernel was based on OpenACC [18], a standard for parallel computing designed to support parallel programming of heterogeneous CPU/GPU systems by means of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator. In order to get good performance from the GPU code, different solutions have been explored.

In a first attempt, the idea was to move the full hydro kernel to the GPU. However, this produced unsatisfactory results, due to various factors, the most relevant being the inefficient access to memory and the large amount of off-loaded data. In order to circumvent the performance limiting factors found with the first approach, a different strategy was adopted. With this approach memory is reorganised in order to be more efficiently accessed and data arrays are properly split in order to support the overlap in time of data transfer (between CPU and GPU) and work. The implementation of this second solution is almost completed.

Organisation of the website

All the details of the accomplished and on-going work can be found in the RAMSES website (http://hpcforge.org/plugins/mediawiki/wiki/ramses/index.php/Main_Page), that is organieed in the three standard sections:

Workplan and status

In this section a summary of the code re-design and refactoring work expected in WP8 can be found. Specific pages describe the work performed for the hybridisation and the development on the GPU, with algorithmic details, tests and performance.

Documents

The Documents section collects guides, papers and presentations relevant to RAMSES and the accomplished work performed.

Downloads

For the management of the developed software a SVN [19] repository is available at

https://svn.physik.uzh.ch/repos/itp/ramses

The repository is organised as follows:

- a root directory (ramses) contains to main source code repositories, trunk, where the public (and stable) code is stored and managed, and branches;
- In branches, the ramses_omp and the ramses_gpu development lines can be found, where source codes under development in WP8 are managed.

Finally, additional information on the RAMSES code can be found in the Community portal page.

3.2 EAF-PAMR

The EAF-PAMR – HD & MHD OCL Module is a hydro and magnetohydrodynamical code for describing interstellar medium, that uses the OpenCL platform [25] for heterogeneous parallel computing. The main objective of this code is to create a tool that can take advantage of both the new graphical processing units while still keeping the traditional CPU clusters as an option to run simulations, although OpenCL in itself does not work in multiple node clusters there are options to allow that possibility such as using MPI or, for example, the SnuCL extention to OpenCL [26].

While the original idea was to adapt previously existent code to OpenCL, the characteristics of the OpenCL platform meant that it was much easier and faster to create a new independent code.

The following points represent the objectives already completed

- Implementation of dimensionally split 3D hydrodynamic methods based on the Roe's linearised solver, the HLLC solver and the Piecewise Parabolic Method interpolation.

- Implementation of dimensionally split 3D TVD magnetohydrodynamical method.
- Domain and problem decomposition to allow the program to run in several GPUs or CPUs (see Figure 6)

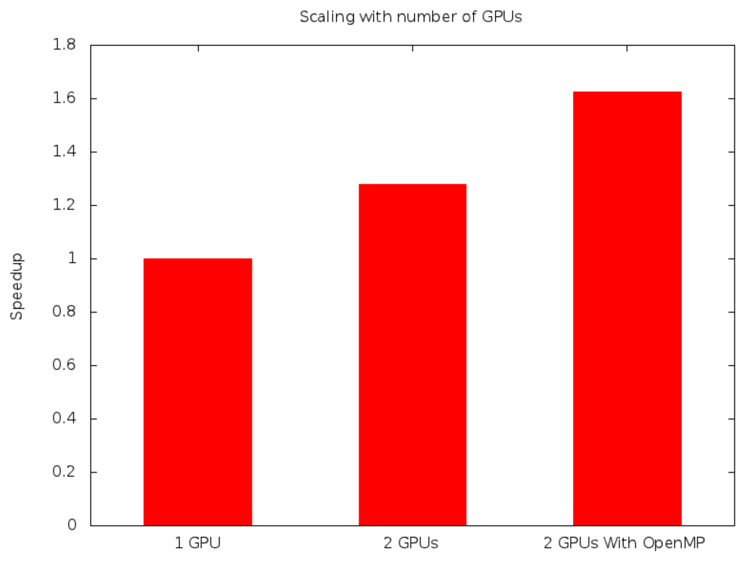


Figure 6: Speedup of the EAF-PAMR code in a 3D Sedov Blast test with a 128^3 computational cells mesh, using 1 or 2 NVIDIA GeForce GTX 260 GPUs and 2 GPUs and two cores.

In addition to the points above, the list bellow represents the objectives that are being worked on:

- Implementation of a gravity solver in OpenCL
- Implementation of unsplit HD and MHD 3D methods using PPM interpolation
- Implementation of Adaptative Mesh Refinement in OpenCL

The code will be made available for download at a later date including documentation.

As of now the wiki site of the EAF-PAMR code (https://hpcforge.org/plugins/mediawiki/wiki/eaf-pamr/index.php/Main_Page) describes the code and the work status and presents the results of the preliminary performed tests and benchmarks.

3.3 PFARM

PFARM is part of a suite of programs based on the ‘R-matrix’ ab-initio approach to variational solution of the many-electron Schrödinger equation for electron-atom and electron-ion scattering. The package has been used to calculate electron collision data for astrophysical applications (such as: the interstellar medium, planetary atmospheres) with, for example, various ions of Fe and Ni and neutral O. This approach implements the Baluja-Burke-Morgan (BBM) method and a two-stage approach is taken. Firstly, parallel sector Hamiltonian diagonalisations are performed using a domain decomposition approach with the ScaLAPACK-based code EXDIG. Secondly an energy-dependent propagation (EXAS stage) across the sectors is then performed using an MPI-based approach with functional parallel groups of processes, a large proportion of which form multiple systolic pipelines computing different sector calculations. Detailed descriptions of the BBM methodology and its associated parallelisation can be found in the ‘Documents’ section of the Wiki.

A preceding PFARM enabling project in the UK [21], involved implementing, amongst other improvements, parallel sub-groups for concurrent parallel Hamiltonian diagonalisations, multicore-node optimisation via shared-memory numerical libraries and more efficient output of results via multiple manager tasks. This work improved the parallel scalability of the code on high-end machines comprising of several thousand cores, such as the Cray XT4. However, performance analyses of the code in the early stages of this project identified three main bottlenecks that needed to be addressed in order to prepare PFARM for the next generation of HPC architectures: the limited scalability of parallel eigensolver methods in EXDIG, the sequential input of data to the process pipelines in EXAS and ineffective load-balancing of the code on petaflop architectures. To address these issues, the main areas of development work undertaken are:

1. Investigation of the parallel performance and suitability of new parallel eigensolvers for PFARM (EXDIG). These new solvers have recently been made available as part of the ELPA (Eigenvalue SoLvers for Petaflop-Applications [22]) project through a collaboration of several German centres and IBM.
2. Parallelisation of the input routines for surface amplitude data to the multiple process pipelines in EXAS.
3. New algorithms for load-balancing the EXAS stage of the code to reflect computation/communication ratios on the latest HPC architectures, such as the IBM Blue Gene/Q.

Additionally, results are presented from an analysis of new computational accelerator technologies - NVIDIA GPU and the Intel Xeon Phi – for undertaking the core computations in the PFARM code. An almost complete port of the EXAS code to GPUs has also been achieved.

Under the ‘Workplan, Status and Results’ section of the PFARM Wiki on the HPCforge WP8 website detailed descriptions of the software enabling project can be found. This also includes workplan objectives and performance results from both powerful HPC architectures, such as the IBM Blue Gene/Q and novel prototype systems such as the NVIDIA K20 GPU and the Intel Xeon Phi. A description of the accomplished work is listed alongside the workplan objectives and performance results demonstrating the impact of the code optimisations. The PFARM source code is downloadable via a link to a repository under the ‘Downloads’ section. The ‘Documents’ section currently provides links to the main paper on PFARM from Computer Physics Communications; a report on a recent, related optimisation project; a PFARM User Guide; a description of numerical library usage on the Intel Xeon Phi and a graphic that summarises the parallelisation.

It is expected that over the remaining duration of the project the ELPA eigensolvers will be fully integrated into the PFARM EXDIG code and performance results provided for large numbers of cores. Also the wrapper controlling scripts (written in Perl) will be further developed to better automate load-balancing in the parallel code for the user base. In conjunction with one of PFARM’s user communities, a very large flagship scattering calculation involving electron –FeII models with over 5000 channels is planned to be undertaken in the coming months, enabled by these code improvements.

4 Climate Section

This section is dedicated to the codes developed in the Climate domain: OASIS, a software interface between different climate models, ICON, a general circulation model, NEMO, a modelling framework for oceanographic research, and Fluidity-ICOM, a three-dimensional non-hydrostatic parallel ocean model. Furthermore, the work on a common high performance “I/O services” module is presented.

4.1 Couplers: OASIS

The OASIS coupler was identified as a key component of European climate models, and its performance was shown in D8.1.2 to be a bottleneck to petascale and exascale modelling. In D8.1.3, a new version of OASIS, OASIS3-MCT, was examined and tested against OASIS4 with pre-computed weights. Following this, it has been decided to optimise OASIS3-MCT as the coupler for scaling current climate models, and prepare a new generation of coupler for future models.

OASIS3-MCT, combining the coupler from CERFACS with the Model Coupling Toolkit was tested on the PRACE Tier-0 Bullx computer “CURIE” up to 2048 cores. A first public release of this code was done in August 2012, with further developments at the CERFACS repository svn://memphis.cerfacs.fr/home/oasis/OASIS3MCT.

Scaling at ICHEC discovered bottlenecks during initialisation at 120 nodes locally on Stokes, due to file handling. These have subsequently been removed by code development at CERFACS. A further bottleneck at 1000 cores was discovered and under investigation; further scaling and porting work within WP8 is now being undertaken on Hermit, for low-memory nodes, for the OASIS3-MCT is being ported and tested within the EC-Earth v3.1 model: http://dev.ec-earth.org/

For D8.1 it was agreed to work on developing the next generation coupler, targeting models with icosahedral or unstructured grids (e.g. ICON, the Fluidity-ICOM ocean model, etc.). For this Open-PALM (developed at CERFACS and ONERA) was chosen, and work was undertaken to implement the necessary conservative interpolation in this coupler. Joel Chavas of CEA/GENCI of “La Maison de la Simulation” has produced a new version of the SCRIP library used in OASIS and Open-PALM couplers to implement conservative regridding when grid edges are unknown, as is needed for the Open-PALM. This work was included in the modified SCRIP library in the August 2012 release of OASIS3-MCT [20].

Organisation of the website.

The IS-ENES community has its own website support for the development of the OASIS code. This website has been linked from the HPC Forge WP8 page. The development version of the code can be accessed via this page.

The work made in the WP8 is part of a continuous integration of OASIS through a Concurrent Version System (using a subversion server) accessible by every developer.

The website has 3 main sections:

Workplan and status

This section restates the work plan detailed in D8.1.4 report and gives the status of the accomplished work.

First the objectives for the WP are detailed. Then an overview of the achieved work in the WP8 framework is presented. The content of this section is regularly updated and the work done by each developer group feeds into it.

Documents

The “Documents” section mainly contains presentations from members of the OASIS developer group (*pdf* tutorials and reports) giving more detail on the accomplished work. These are presentations (and posters) given at workshops on various aspects of the project.

This section will be completed with performance benchmarks realised with the last version of the code.

Downloads

The work done in WP8 is part of a continuous integration thanks to the concurrent version system maintained by CERFACS. This work is then integrated into point releases, the latest of which was the August 2012 release of OASIS3-MCT including work done under WP8.

A link to the subversion repository is provided. In the “Downloads” section, several snapshots of the development version of the code will be linked as well as all the compiling instructions to build the executable from it.

4.2 Input/Output: CDI, XIOS, PIO

As described in [4], a common “I/O services” module is being developed within the IS-ENES community, for use by all climate models. NeverthelessProgress has been made on the larger design of the proposed “I/O services” module and more specifically:

- I/O services implement a writer service, reading data from the model nodes (those nodes running the climate model itself) via RDMA / single-sided communications. This enables the model to continue while I/O services handles the parallel write (read is not an issue for global models).
- The I/O services implement parallel writes to a number of potential formats, principally netCDF and GRIB, using the CDI library or a variant.
- The I/O services are implemented using separate I/O nodes, to enable buffering of I/O in memory. This balances the large transient communications internal to the compute cluster (e.g. 250 GB/s) to the typically smaller but sustained I/O bandwidth; then I/O scaling becomes a matter of adding additional nodes for I/O.
- Post-processing is then handled on the fly within the I/O services, based on the XIOS model from IPSL. This work is done in collaboration with IPSL (XIOS developers), MPI-M (CDI developers), in parallel to the G8 “ICOMEX” dycore initiative and IS-ENES efforts.
- Within PRACE, ICHEC is implementing an initial template version of the common I/O services based on the ScaLES CDI, in comparison to the existing PIO developed at NCAR. This implements the API, in which the post-processing services will be implemented in parallel by IS-ENES partners (IPSL).

  Work on this has been stalled at ICHEC until recently due to staff shortages. A new CDI interface has been implemented for IFS, within the EC-EARTH framework, and is undergoing testing. This has been tested using GRIB format and the serial version (1.5.6) of the libCDI interface, and is being merged in the EC-Earth development portal. Testing under the parallel (PIO) version of libCDI is planned.

  Development work is on-going at ICHEC to extend XIOS to add “memory cache” nodes: these enable the offloading of model data from compute nodes at large transient communication rates (e.g. 250 GB/s over Infiniband, seen under the ScaLES project within IS-ENES) to buffer writing by the I/O nodes at smaller but sustained I/O bandwidths (eg. 30 GB/s). This is essential to enable the I/O services module to work on memory-constrained systems, e.g. Blue Gene or Cray-based systems. These “memory cache” nodes operate transparently, enabling the “XIOS” (XML I/O Server) subsystem to scale limited by I/O bandwidth rather than node memory.

Work on adding a CDI-XIOS mapping layer, to enable this “memory cache” system to be used with CDI has been postponed, and the work is being tested using the XIOS test suite and NEMO ocean model. XIOS changes are available at the XIOS Subversion repository http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS; IFS changes are being committed to the EC-Earth repository. At the moment the IFS code is closed, but it is being merged and made public as part of the OpenIFS codebase.

Within IS-ENES, work continues on the “PIO” version of the CDI writer library. At CEA, interpolation has been added to the post-processing layer within XIOS. A workshop is planned for May 2013 on merging the components of the I/O services and interpolation systems used.

Organisation of the website.

Work done on the I/O server is done in collaboration with the IS-ENES community; much of this is then documented on the IS-ENES website, and on member websites at DKRZ and IPSL. These websites are linked from the HPC Forge WP8 page. The development version of the codes can be accessed via these pages.

The website has 3 main sections:

Workplan and status

This section recapitulates the work plan detailed in D8.1.4 report and gives the status of the accomplished work.

First the objectives for the WP are detailed.
Then an overview of the achieved work in the WP8 framework is presented. The content of this section is regularly updated and the work done by each developer group feeds it.

Documents

The “Documents” section mainly contains presentations from members of the I/O developer group (*pdf* tutorials and reports) giving more detail on the accomplished work. These are presentations (and posters) given at workshops on various aspects of the project.

This section will be completed with performance benchmarks realised with the last version of the code.

Downloads

Accomplished work is then available in the component repositories for IFS (currently in the development section of the EC-Earth repository, as it is being tested under this framework, later to be merged into the IFS and CDI repositories) and the XIOS repository at IPSL.

A link to the subversion repositories are provided. In the “Downloads” section, several snapshots of the development version of the code will be linked as well as all the compiling instructions to build the executable from it.

4.3 Dynamical Cores: ICON

The ICON (ICOsahedral Non-hydrostatic) Model is a joint project of the Max Planck Institute for Meteorology (MPI-M) and the German Weather Service (DWD) for the development of a new generation of unified weather forecasting and climate models using triangular grids arising from a spherical icosahedron (a decomposition of the globe into 20 spherical triangles). Of particular interest in the development of very high-resolution models is the non-hydrostatic dynamical core (NHDC), which solves the equations of atmospheric motion without making the assumption that the atmosphere is in hydrostatic balance. This means it is applicable from low resolutions to very high (less than 10km.). Grid refinement is planned for ICON, in which areas of particular interest will the at much higher resolution than the rest of the globe.

Our goal in this project was to first implement single-node accelerator-enabled versions of the NHDC and perform an extensive evaluation of these, then implement a multi-node (domain decomposed) accelerator-enabled NHDC using the Message-Passing Interface (MPI) for communication and OpenACC directives for porting the code to accelerators (e.g., GPUs). These are to be benchmarked on a wide array of platforms available at CSCS. As a secondary objective we planned to integrate the GPU-enabled ICON physical parameterisations from Lapillonne, et al., stemming from an independent project, in order to offer a fully GPU-enabled atmospheric general circulation model (AGCM). As a final object, the GPU-enabled code should be reintegrated into the development code base with the help of MPI-M and DWD.

The realisation of this project has proceeded roughly as planned. To date the following objectives have been achieved:

- Two single-node GPU-enabled NHDC versions were completed:
    - Fortran + OpenCL implementation with C-wrappers for the kernels
    - CUDAFortran (proprietary compiler from Portland Group) implementation
- A multi-node (domain decomposed) version of the code was implemented in close collaboration with MPI-M. This utilised the MPI for communication and OpenACC compiler directives for offloading execution to the accelerators.
- All versions were extensively benchmarked on platforms with various accelerators, and the results were communicated in various presentations and posters at a number of workshops. The bottom line was roughly a 2x speedup on one NVIDIA Kepler K20x with respect to the main development code running on clusters of Intel dual-Sandybridge nodes, commonly agreed to be the best current CPU architecture currently for the software.
- Discussions with developers have been ongoing during the entire duration of the project, and they have agreed to a reintegrate the work into the main development branch. The exact procedure for this reintegration is currently being defined.

The one open point in the work plan is the integration of the ICON physical parameterisations, which are being enabled for GPUs in a separate project (www.hp2c.ch). That project has been delayed, meaning that we have not been able to start integrating that work.

ICON NHDC Project Web-page

An extensive description of the project can be found in the project web-site:

https://hpcforge.org/plugins/mediawiki/wiki/icon-gpu/index.php/Main_Page

The web site is split up into a several different sections

- **ICON Overview:** contains an overview of the software
- **Work plan and Status:** recapitulates the work plan from D8.1.4, and gives the current status as of the last milestone date, in particular giving an overview of the work accomplished within WP8, as well as a list of open issues
- **Documents:** contains posters and presentations given at workshops on various aspects of the project, as well as performance results, and a QuickStart manual on how to compiler and run the software
- **Download:** contains information on how to download the software.

4.4 Ocean Models: NEMO and Fluidity-ICOM

NEMO (Nucleus for European Modelling of the Ocean) is a modelling framework for oceanographic research, operational oceanography, seasonal forecasts and climate studies. It is of great strategic importance for the UK and European oceanographic communities.

The code itself is written in Fortran90 and parallelised using MPI. Portability and stability of the code base are of great importance. The parallelisation is done via a geographic domain decomposition in the horizontal, but not vertical, dimensions. The fundamental equations are solved using a finite-difference scheme with a simply-connected, curvilinear grid. The model deals with both 2- and 3-dimensional fields, which are represented as Fortran arrays. These arrays are declared with haloes, and field values on the haloes must be exchanged frequently, which involves nearest-neighbour, point-to-point communications.

The MPI parallelisation of NEMO has served it well for many years. However, the trend for increasing core counts on the compute nodes of high-performance computers (*e.g.* IBM's BG/Q can support 64 processes per node) means that it is gradually becoming ill equipped for making efficient use of such machines. Current best practice in programming for these computers is to use OpenMP across the cores in a node and MPI between nodes – the so called 'hybrid' or 'mixed-mode' approach.

In WP8, STFC has explored the performance implications of various approaches to introducing OpenMP threading into the code. In doing this we have also experimented with the choice of array index ordering used for the 3-dimensional fields. NEMO's ancestor, OPA, was developed with vector architectures in mind and thus NEMO itself has the *x* or longitude grid index as the leading dimension for all of its arrays since this is typically the largest of the three grid dimensions. In contrast, codes developed more recently (such as WRF and POLCOMS) for scalar MPP architectures have tended to favour having the z grid index as the leading array dimension. By experimenting with two different kernels taken from the NEMO code, we aim to discover which array-index ordering is best for NEMO as well as the best way to introduce OpenMP.

These conclusions will then be used to inform the work of Italo Epicoco and Silvia Mocavero of the Centre for Mediterranean Climate Change (a member of the NEMO consortium) as they tackle the considerable task of introducing OpenMP throughout the NEMO code base.

Fluidity-ICOM (http://amcg.ese.ic.ac.uk/index.php?title=Fluidity) is an open-source partial differential equation simulator build upon various finite element and finite volume discretisation methods on unstructured anisotropic adaptive meshes. It is being used in a diverse range of geophysical fluid flow applications. Fluidity-ICOM uses three languages (Fortran, C++, Python) and uses state-of-the-art and standardised 3rd party software components whenever possible.

Using modern multi-core processors presents new challenges for scientific software such as Fluidity-ICOM, due to the new node architectures: multiple processors each with multiple cores, sharing caches at different levels, multiple memory controllers with affinities to a subset of the cores, as well as non-uniform main memory access times. Because of this, there is a growing interest in hybrid parallel approaches where threaded parallelism is exploited at the node level, while MPI is used for inter-process communications. Significant benefits can be expected from implementing such mixed-mode parallelism.

In WP8, STFC agreed to contribute 6 months effort to carry on continuous work on developing hybrid OpenMP/MPI parallelism for Fluidity-ICOM which is driven by its developer and user communities. We have threaded advection diffusion equation's matrix assembly kernels using Continuous Galerkin method and Control Volume method. With previous work on momentum equations, all matrix assembly kernels have now been fully threaded. Benchmarking results with HYPRE and threaded PETSc show that mixed mode MPI/OpenMP version can outperform pure MPI version at high core counts where I/O becomes major bottleneck for pure MPI version. With mixed mode MPI/OpenMP, Fluidity-ICOM can now run up to 32K cores job, which offers Fluidity-ICOM capability to solve the "grand-challenge" problems.

Structure of the Wiki Pages

The main page for the NEMO section on the WP8 web pages can be found at: http://hpcforge.org/plugins/mediawiki/wiki/nemo/index.php/Main_Page

This page contains a brief introduction to NEMO and links to pages describing the 'Work Plan, Status and Results', 'Documents' and 'Downloads'. In the first of these we describe the work undertaken in more detail and the results achieved. Results are mainly given for the Intel Sandy Bridge chip although some results for the BG/Q and Intel Xeon Phi are also shown. We also discuss the implications of the choice of array-index ordering for the performance of the code at the strong-scaling limit with the current domain decomposition for MPI.

The Documents page contains a link to the official NEMO documentation and the presentation prepared for the face-to-face meeting at CSCS, Lugano in March 2013. The Downloads page contains a link to the code tarball containing the kernels developed during the work.

The main page for the Fluidity-ICOM section on the WP8 projects web pages can be found at: http://hpcforge.org/plugins/mediawiki/wiki/icom/index.php/Main_Page, There are three components including Work plan, Status and Results, Documents and Downloads. In the Workplan, Status and Results, we have put the original work plan and the current project progress with performance results and analysis. The results show that the OpenMP development enables Fluidity-ICOM running up to 32K cores jobs. We have also shown the Fluidity-ICOM code validation. In the Documents, we have put the Fluidity-ICOM's user manual and relevant link for Fluidity-ICOM. In the download section, we have given the link to download Fluidity-ICOM and the latest development trunk links.

5 Material Science Section

The Material Science domain accounts for four applications all devoted to electronic-structure calculations and materials modelling at the nanoscale. The first two, ABINIT and Quantum ESPRESSO, are large packages which implement a number of different algorithmic solutions (density-functional theory, plane waves, and pseudopotentials). The other two codes, Siesta and Exiting/ELK, adopt more specific approaches to describe the systems under investigation.

5.1 ABINIT

ABINIT is a package, available under the GNU General Public Licence (GPL), whose main program allows one to find from first principles the total energy, charge density, electronic structure and miscellaneous properties of systems made of electrons and nuclei (molecules and periodic solids) using pseudo-potentials and a plane-wave or wavelet basis. The basic theories implemented in ABINIT are Density-Functional Theory (DFT), Density-Functional Perturbation Theory (DFPT), Many-Body Perturbation Theory (MBPT), and Time-Dependent Density Functional Theory (TDDFT).

Historically, ABINIT uses plane-waves to describe the electronic wave functions; in recent years, a development of wave functions utilising a wavelet basis has been introduced.

The work in *PRACE-2IP Work package 8* has been driven by CEA-Bruyères-le-Châtel (GENCI) and involved four groups of developers:

1. *CEA – Bruyères-le-Châtel (Paris, France)* : Ground-state + plane waves
2. *CEA – INAC (Grenoble, France)* : Ground-state + wavelets
3. *UCL (Louvain-la-Neuve, Belgium)* : Excited States, Linear Response
4. *BSC (Barcelona, Spain)* : Linear Response

From the "Performance analysis" and "Performance improvements exploration" phases, four refactoring tasks have been defined:

1. *Ground-State calculations using plane waves:*
   Shared memory approach, load balancing, processes distribution automation
2. *Ground-State calculations using wavelets:*
   Automatic code generation for convolutions
3. *Excited States calculations:*
   Hybrid approach, Scalapack implementation, new processes distribution
4. *Linear Response calculations:*
   Parallel I/O, parallelism in new sections, distribution of wave-functions

All the above listed tasks have been achieved.

The Ground-State, excited states and Linear response codes can now take benefit from a fully hybrid approach (MPI+openMP); a special effort has been made in the refactoring of FFTs.

The load balancing of the ground-state code is now improved: while load balancing due to band distribution was easy to correct, load balancing due to plane-wave distribution required more work and a small communication before each FFT call could not be avoided (Figure 7). While (before) the user had to settle ten parameters to effectively launch Abinit in parallel, the code is now able to start without any indication; by default, the code uses a simple heuristics to distribute processes and tasks. On demand ABINIT can also execute series of small benchmarks in order to refine the processes distribution and to choose the best library (ScaLapack, ELPA, Cuda…).

ABINIT has been interfaced with ELPA and PLASMA [23] matrix algebra libraries. ELPA brings a small improvement (10 to 20%) on large physical systems; PLASMA is more promising but it has to be fully tested (at the time of the implementation, PLASMA did not provide the computation of eigenvectors).

The input/output routines of the "excited states" and "linear response" sections of ABINIT have been improved by the use of MPI-IO; these I/O sections were detected as bottlenecks at the beginning of this work; these bottlenecks have now disappeared.

A prototype of "automatic code generation" has been written for the wavelet approach; it is assumed to be used for the convolutions used in the BigDFT library. Several versions of these convolutions have been written and can be automatically selected according to the architecture (hardware and software).

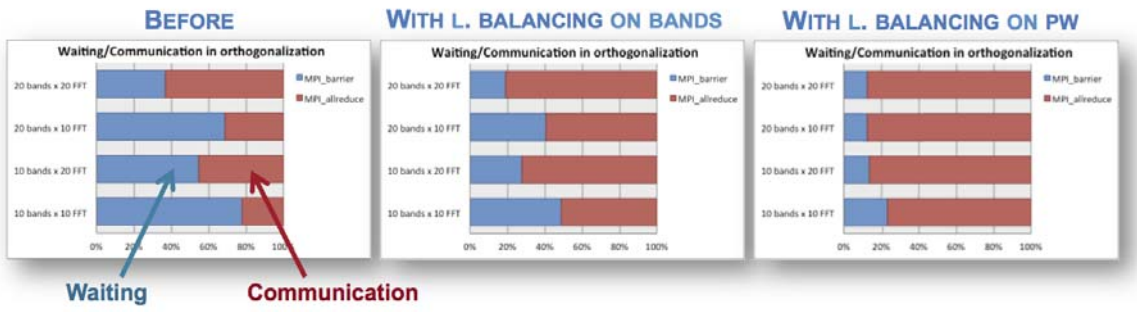


Figure 7: Repartition of time (%) for the MPI communication before the orthogonalisation of wave-functions. In blue, the waiting time; in red the actual communication time. The first graph shows the status before the present work. The second graph shows the status after the improvement of load balancing due to band distribution only. The third graph shows the status after the full improvement of load balancing. Test case: 107 gold atoms on TGCC-Curie Prace computer.

Organisation of the website.

ABINIT developer community (around 50 people) has its own website whose purpose is a support for the development of the code. This website has been linked from the HPC Forge WP8 page. The development version of the code can be accessed via this page.

The work made in the WP8 is part of a continuous integration of ABINIT through a Concurrent Version System (using a Bazaar server) accessible by every developer.

The website has 3 main sections:

Workplan and status

This section recapitulates the work plan detailed in D8.1.4 report and gives the status of the accomplished work.

First the objectives for the WP are detailed.
Then an overview of the achieved work in the WP8 framework is presented. The content of this section is regularly updated and the work done by each developer group feeds it. In order to demonstrate the validity of the new contributions to ABINIT several tables and graphs are presented, comparing the performances of the code before and after the WP8 project.

Documents

The “Documents” section mainly contains presentations from members of the ABINIT developer group (*pdf* or *ppt* files) giving more detail on the accomplished work. These are presentations (and posters) given at workshops on various aspects of the project.

This section will be completed with performance benchmarks realized with the last version of the code.

Downloads

Differently from other codes (except Quantum Espresso), the work made in the WP8 is part of a continuous integration thanks to the concurrent version system ("Bazaar"). The "Launchpad" integrated web application is used, allowing the ABINIT developers to maintain the software, sharing their developments (source code) via a convenient web interface.

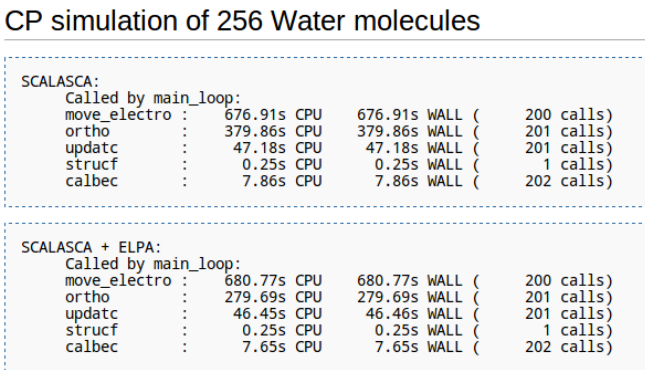
A link to the Launchpad forge will be provided. Unfortunately, the access the forge is filtered (*login* and *password*) and only certified people can access it; it is nevertheless possible to ask for an access by justifying the need for it.

For that reason, in the "Downloads" section, several snapshots of the development version of the code will be linked as well as all the compiling instructions to build the executable from it.

5.2 Quantum ESPRESSO

Work in WP8 has been driven by CINECA and involved ICHEC, IPB, and University of Sofia. Each partner, according to the declared effort, is taking part to the following action items.

- Refactoring and optimisation of linear algebra kernels by tuning of the existing kernels and implementing ELPA libraries (work in progress - see Figure 8)
- Band parallelisation. The parallelisation level based on band indexes has been extended from the PW and CP modules to GIPAW and Phonon (completed)
- Parallelisation of the EPW module (work in progress)
- Testing and validation of the EXX calculations using the G2 test set. (completed)
- Porting and testing to GPUs and Intel Xeon Phi architecture (work in progress)


CP simulation of 256 Water molecules

```
SCALASCA:
     Called by main_loop:
     move_electro :    676.91s CPU    676.91s WALL (     200 calls)
     ortho        :    379.86s CPU    379.86s WALL (     201 calls)
     updatc       :     47.18s CPU     47.18s WALL (     201 calls)
     strucf       :      0.25s CPU      0.25s WALL (       1 calls)
     calbec       :      7.86s CPU      7.86s WALL (     202 calls)
```

```
SCALASCA + ELPA:
     Called by main_loop:
     move_electro :    680.77s CPU    680.77s WALL (     200 calls)
     ortho        :    279.69s CPU    279.69s WALL (     201 calls)
     updatc       :     46.45s CPU     46.46s WALL (     201 calls)
     strucf       :      0.25s CPU      0.25s WALL (       1 calls)
     calbec       :      7.65s CPU      7.65s WALL (     202 calls)
```

Figure 8 Profiling of QuantumESPRESSO CP module for a WAT64 system with (above) and without (below) ELPA.

Organisation of the website.

Since a website for Quantum Espresso package exists, the HPC Forge infrastructure has been considered only as a support for the development and work in the WP8 activity. A large part of the material already available on the Quantum Espresso portal has been linked and it is freely accessible through the HPC Forge WP8 page.

The website has 3 main sections:

Workplan and status

The content of this section is updated by each partner that contributes to this WP activity. This is meant to be a tool to know the advancement of the work performed by each partner in a way that permits to know the global status.

Documents

In the documents section we store results and documentation relative to the accomplished work. While writing this document, for example, in this section is possible to find benchmarks from the implementation of ELPA and a preliminary testing and validation for the exact-exchange pseudopotentials.

This section will become more rich when the WP8 activity will be close to its end, since every report of activity for each partner and benchmarks for the final achievements of the task will be available.

Downloads

Differently from other codes, the accomplished work is part of the continuous integration of Quantum Espresso through a svn server open to every user and developer. This ensures the possibility of a continuous feedback and a fast way to find eventual bugs. As soon the modifications on the code are synchronised with the main svn trunk, they are made available through the Quantum Espresso website. Then, to avoid a redundancy, we preferred to provide in this website only a link to the Quantum Espresso portal from where the last svn available version can be downloaded.

We are, as well, considering to use this section of the website, to public release scripts and testing codes that could be useful to users to test the improvements obtained thanks to the WP8 activity.

The content of this website is still growing and, by the end of the work package, all the documentation will be made available to the community together with the last downloadable version of the code.

5.3 Siesta

SIESTA (Spanish Initiative for Electronic Simulations with Thousands of Atoms) is both a method and its implementation as a computer program, to perform electronic structure calculations and ab-initio molecular dynamics simulations of molecules and solids.

For larger systems the most costly step is the diagonalisation, currently done with ScaLAPACK, a widely used and robust method. But in turn it features limited scaling and does not use the sparsity of the Hamiltonian and overlap matrices generated by Siesta.

The accomplished work can be summarized as follows:

- Implemented and tested a prototype of the Sugiura Sakurai method
- Tested the FEAST library, which is based on an algorithm similar to the Sugiura Sakurai method
- Tested the GPU-based eigenvalue solver of MAGMA [24]
- Cooperated with a group at the Lawrence Berkeley National Laboratory for testing and developing a prototype of the PEXSI algorithm.
- Work on the performance of the “Orbital Minimisation Method”

Tests showed, that some features of the matrices generated by Siesta create serious problems with the load balancing of the Sugiura Sakurai and FEAST algorithms. Thus the work on those methods has been stopped.

A new approach is given by the PEXSI (Pole EXpansion and Selected Inversion) method, which avoids the diagonalisation. By taking advantage of the sparsity of the density matrix, this method reduces the computational complexity to O(n^2) for 3D systems, to O(n^3/2) for quasi-2D systems, and to O(n) for quasi-1D systems. Moreover it offers the possibility of massive parallelisation. Those advantages make it particularly interesting for very large systems. The prototype implementation showed that systems up to tens of thousands of atoms can be addressed already.

Organisation of the website.

The organisation of the web page follows the standard structuring into “Workplan and Status”, “Documents”, and “Downloads”.

Workplan and Status

This section is divided according to the algorithms and libraries tested, in chronological order. Each section gives a short overview, describes the work done, and lists the results. Due to the importance of the PEXSI method, the presentation of its results is more detailed and contains also a description of the test examples. For the Sugiura Sakurai method there is also a reason for stopping the work on it given.

The work on the eigenvalue solvers is described in much more detail in a report, which is accessible in the “Documents” section.

Documents

In this section a list of links, references, and downloadable reports provides guidance for further reading.

Downloads

By now there are no downloads like code segments available, which might change in future.

Expected future content

Most of the additional content will be related to the implementation of PEXSI into Siesta. It will involve:

- Performance analysis for full SCF runs regarding weak and strong scaling.
- Analysis of the behavior of PEXSI and the effect of reusing data during a full SCF run. This can also be extended to MD simulations.
- Comparing the performance with other solvers (ScaLAPACK and maybe ELPA) and giving some heuristics about when to use which solver.
- Robustness and convergence of PEXSI for different types of problems.
- User-guide about usage and parametrisation of PEXSI

5.4 Exciting/ELK

Exciting and Elk are the two rapidly developing full-potential linearized augmented plane-wave (FP-LAPW) codes with a rich set of features and a large user community. Both codes were forked from one predecessor code and thus bear a lot of common functionality.

The principal requests from the Exciting community for the code refactoring was to make it possible to run electronic structure simulations for large unit cells containing up to a thousand atoms. During the preparatory phase of the WP8 the major bottlenecks of the codes were identified and the decision was made to re-implement the basic low-level LAPW functionality in a standalone prototype library, which we have named SIRIUS.

In the time frame of the WP8 project we focused only on the optimisation and scaling of the ground state calculations, i.e. a path from the crystal structure input to the ground state charge density and magnetisation. We didn't touch other parts of the code like phonons, forces, linear response calculations, etc.

At the moment we have accomplished all the prototype functionality of the library, which is required to run ground state calculations at a scale (the ground state runs with >1000 atoms in the unit cell have been demonstrated). Now we need the feedback of the community, which will help to plan the future road map of the library.

The organisation of the hpcforge.org web page follows the standard structuring into "Workplan and Status", "Documents", and "Downloads".

Workplan and Status

This section contains the basic motivation for re-implementing LAPW functionality in the standalone library, the list of accomplished tasks and possible future road map of the library. As was mentioned above, we need the community feedback, which will guide the future development of the library.

Documents

This section contains the short description of the library and it's design, the step-by-step compilation and linking instructions and the detailed explanation (with formulas) of the internal wave-function representation with the corresponding example of charge density generation.

Downloads

The SIRIUS library in its current state and the patched version of the Elk code are uploaded into the project's Git repository and are available both for anonymous read-only and full read-write access. We also provide the link to the slides with the presentation of SIRIUS library given at the March PRACE-2IP WP8 face-to-face meeting at CSCS (Lugano) and the partial documentation of the code generated by the Doxygen.

6 Particle Physics Section

In the Particle Physics area, the focus is on Quantum Chromo-dynamics. Here a single code, PLQCD, has been selected for re-design and refactoring.

6.1 PLQCD

TmLQCD and Chroma are two software suites used by the Lattice QCD community. TmLQCD is developed by the European Twisted Mass Collaboration (ETMC) while Chroma is developed by the USQCD collaboration in the United States. These software suites provide functionality for performing lattice QCD simulations as well as lattice calculations of physical observables. While tmLQCD is mainly focused on Twisted-Mass and Clover fermions, Chroma provides codes for almost all types of formulations of lattice fermions including Domain-Wall, Overlap and Staggered fermions. At the beginning of this project, tmLQCD and Chroma were implemented using MPI for parallelisation. Chroma also has a multi-threaded version, but not a hybrid implementation.

In WP8, we focus on the following goals:

- Improving the implementation of the Dirac operator for Wilson and Twisted Mass fermions within tmLQCD: The Dirac operator is the main kernel in lattice codes and gets called a large number of times. As such, optimizing the implementation of this operator is crucial for performance. The main objective is to provide a stand-alone library for the Dirac operator with MPI+openMP parallelisation. In addition, we implement other improvement aspects for better performance. These aspects focus on overlap of communications and computations, using a compact representation of the gauge-links in order to improve the arithmetic intensity, use of compiler intrinsics instead of inline assembly, and use of compression algorithms for MPI. These improvements will be targeted towards the tmLQCD software suite.
- Implementing efficient linear solver within tmLQCD: The sparse linear solver is an essential component of lattice calculations for computing the hadronic spectrum and other physical quantities. In addition, it is needed for the Hybrid Monte Carlo simulations. Standard sparse solvers, such as Conjugate-Gradient become very slow at the physical light quark masses. Twisted-Mass fermions are also special in the sense that some of the standard solvers such as BiCGStab that are efficient for Wilson and Clover fermions fail to converge for Twisted-Mass fermions. In this work package, we implement efficient linear solvers for Twisted-Mass fermions.
- Improving the parallelisation of the Landau Gauge Fixing code within Chroma: on the lattice, Landau gauge fixing is performed using a local optimisation method, such as Steepest Descent. Such methods suffer from critical slowing down: the number of iterations needed to solve the problem increase with V^z, where V is the size of the problem (in our case, the lattice volume). For gauge fixing, naïve procedures have an associated critical exponent z around 2. A Fourier-accelerated method has been proposed (with z~0), but it requires the use of Fast Fourier Transforms (FFT), whose parallelisation is far from being trivial. The standard and well-known FFTW package provides parallel routines which only parallelize along one dimension. In this work-package we implement the PFFT parallel FFT package which allows for parallelisation in three dimensions. This work will be done within Chroma.
- Tuning and optimisation of the HMC integrator within Chroma: The standard algorithm to generate lattice configurations with dynamical quarks is Hybrid Monte Carlo (HMC). The most time consuming part of the HMC method is the molecular dynamics (MD) step, where we evolve the system using some approximate integrator.

In general, the integrator contains free parameters, which can be tuned such that the acceptance rate is maximized, while keeping the step size as large as possible. The best way to do this is using Poisson bracket measurements. Depending on how well tuned the default integration scheme is, the optimisation of the integrator allows to decrease the number of the inversions of the fermionic matrix needed by HMC. New integrator schemes can be tested (such as force-gradient integrators), hopefully providing further improvements.

The major parts of the refactoring plan have been achieved. For the Dirac operator we have implemented an openMP+MPI version with overlapped communications and computations. We have also implemented other improvement aspects including the use of Intrinsics for Vectorisation (SIMD), the use of a compact representation of the gauge links to improve the floating point intensity of the Dirac operator, testing an MPI compression algorithm to reduce the communication time, and overlapping communications and computations. For the linear solver, we have implemented the Incremental Eig-CG linear solver within tmLQCD. A related solver for the non-Hermitian case (Incremental Eig-BiCG) was also implemented. However, it was found not perform as well as expected for the Twisted-Mass case. A Fourier-Accelerated Landau Gauge Fixing using the PFFT library has been implemented within Chroma. An optimisation of the Omelyan Hybrid Monte Carlo integrator has been also implemented within Chroma.

In Figure 9, the scaling of the Dirac operator with MPI and MPI+openMP is shown. Further scaling tests and additional improvements will be implemented before the end of the project.

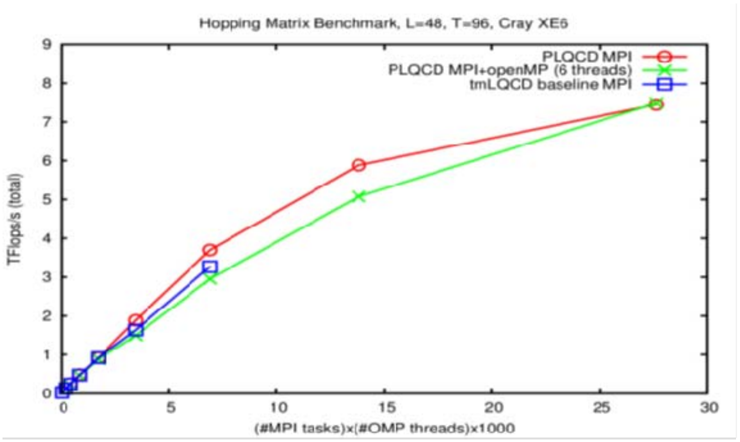


Figure 9: Strong scaling results of the Hopping Matrix on Cray XE6

In Figure 10, we show speedup achieved in the linear solver time on a large Twisted-Mass lattice with light quark mass. In Figure 11, we show scaling of the Fourier accelerated Landau gauge fixing code using the PFFT library. Finally, in Figure 12, we show improved acceptance rates for the Hybrid Monte Carlo algorithm using the optimized integrator

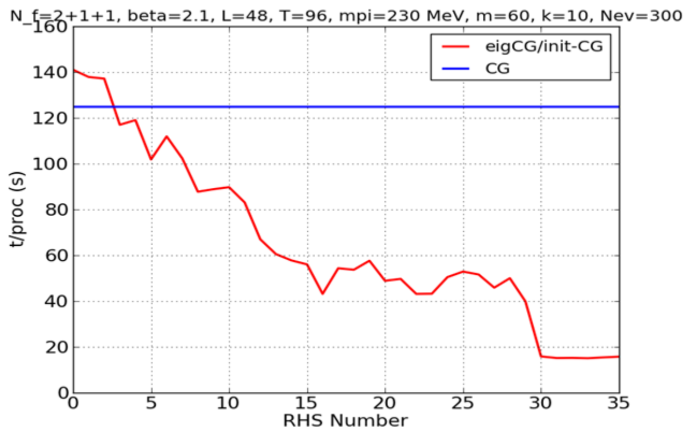


Figure 10: Linear solver time for Incremental Eig-CG as compared to CG for each Right-Hand Side.

By the end of the project we plan to implement and improve three additional linear solvers, namely, a mixed precision CG with reliable updates, GMRES-DR, and Multi-Mass Incremental Eig-CG. We will also perform more work on the SIMD vectorisation section of tmLQCD with extensions to AVX and the wider Xeon Phi registers.

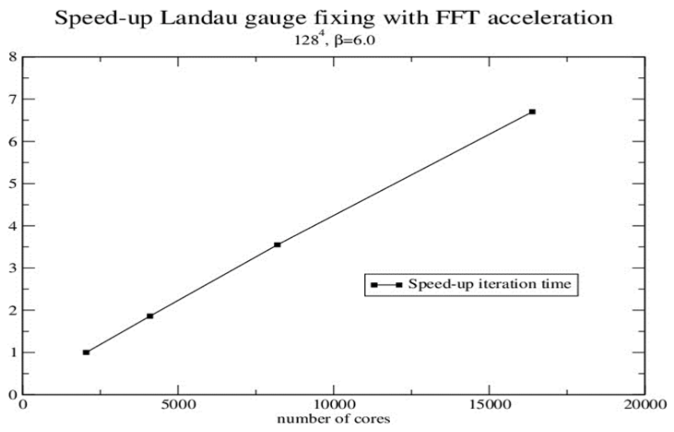
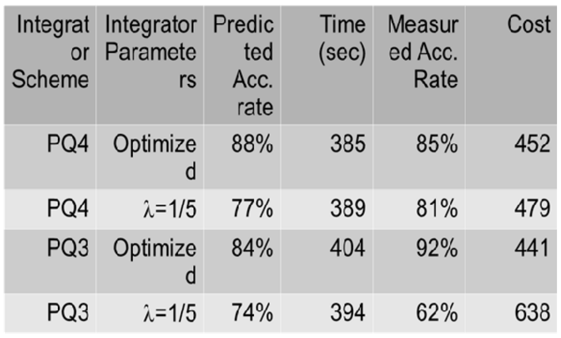


Figure 11: Scaling of the Fourier accelerated Landau gauge fixing.



Integrator Scheme	Integrator Parameters	Predicted Acc. rate	Time (sec)	Measured Acc. Rate	Cost
PQ4	Optimized	88%	385	85%	452
PQ4	λ=1/5	77%	389	81%	479
PQ3	Optimized	84%	404	92%	441
PQ3	λ=1/5	74%	394	62%	638

Figure 12: Acceptance rates for the Hybrid Monte-Carlo with optimized integrator.

PLQCD webpage on hpcforge:

A description of the codes, work plan and documentation can be found on the project's web-page on hpcforge under PLQCD (http://www.hpcforge.org/projects/pracelqcd/). The main sections of the page are:

- Mediawiki

This section provides a description of the codes and the workplan as well as achieved results. The page has links to the main results for the different parts of the code.

- SCM

  This section provides information on the latest version of the code that can be cloned using Git. This will be the main portal for users who would like to download these codes.

- Docs

  In this section, a manual of the code will be provided as well as other documents for presentations and other relevant documents related to the code development.

Other sections of the webpage will be used as needed and the page will be kept up to date.

7 Engineering Section

The Engineering domain focused on several computational challenges, namely parallel mesh refinement, parallel mesh generation and solvers scalability, developing solutions for various codes, like Elmer, Code_Saturne, Alya, ZFS and a custom Finite Volume solver.

7.1 Elmer

Discretisation of most engineering problems, leads to large sparse linear systems of equations

$$\bar{K}u = \bar{f}.$$

This system can be then solved by means of **direct, iterative** or **hybrid** (combining both previous) **solvers**, which can be also categorised into **primal, mixed** (saddle point) or **dual** methods**.**

When the sequential solution is not possible (because of the large problem's size, memory requirements or long computational times), it is necessary to solve it in parallel. Parallelisation of efficient algorithms for the solution of linear systems can be implemented mostly using SPMD technique – **distributing data** portions among processing units

$$\bar{K} = \begin{bmatrix} \bar{K}^1 \\ \vdots \\ \bar{K}^{Nc} \end{bmatrix}, \bar{f} = \begin{bmatrix} \bar{f}^1 \\ \vdots \\ \bar{f}^{Nc} \end{bmatrix}.$$

This allows algorithms to be almost the same for sequential and parallel case; only data structure implementation differs.

Natural and favourable data distribution comes from the **domain decomposition**. To ensure the solution's continuity, overlapping domain decomposition requires sharing of the small overlap by neighbouring subdomains and non-overlapping domain decomposition brings an additional equality constraint ensuring the gluing conditions on subdomain interfaces

$$Ku = f \;\; s.t. \;\; Bu = o,$$

where $K = \begin{bmatrix} K^1 & & \\ & \ddots & \\ & & K^{Ns} \end{bmatrix}, f = \begin{bmatrix} f^1 \\ \vdots \\ f^{Ns} \end{bmatrix} \; R = \begin{bmatrix} R^1 & & \\ & \ddots & \\ & & R^{Ns} \end{bmatrix}, B = [B^1 \;\; \cdots \;\; B^{Ns}].$

The equality constraint can be then enforced by penalty or by projector to the kernel of constraint matrix

$$(K + \rho B^T B)u = f \;\; or \;\; P_B K u = P_B f,$$

or using the Lagrange multipliers resulting in saddle point problem.

One of the most successful classes is called FETI (Finite Element Tearing and Interconnecting). The original primal equality constrained problem can be by means of **duality** and homogenisation transformed into significantly smaller better conditioned dual equality constrained problem

$$F\lambda = d \;\; s.t. \;\; G\lambda = o,$$

which can be solved again by penalty or projector

$$(F + \rho G^T G)\lambda = d \;\; or \;\; P_G F \lambda = P_G d.$$

For the dual problem solved by the CG method classical estimate by Farhat, Mandel and Roux of the spectral condition number is valid, i.e. $\varkappa(P_G F P_G | Im\, P_G) \leq C\frac{H}{h}$ with H denoting decomposition parameter and h the discretisation parameter. Natural effort using the massively parallel computers is to maximize number of subdomains (decrease H) so that sizes of subdomain stiffness matrices are reduced which accelerates not only their factorisation and subsequent pseudoinverse application but also improves conditioning and reduces the number of iterations. Negative effect of that is an increase of dual and null space dimension, which decelerate the coarse problem (CP) solution, so that the bottleneck of the TFETI method is the application of the projector.

The basic advantage of the **primal methods** is absence of the null space R, no need to factorize K matrix and apply K^+, no assembling of G matrix and multiplication by it and its transpose, no need to build GG^T matrix and solve CP $GG^Tx = y$, we do not have to reconstruct the primal solution from dual one, etc. In comparison to primal methods having no preprocessing, in **dual methods** the K regularisation, its factorisation, building G, GG^T, GG^T factorisation is more time consuming and memory needs are very large being so the limiting factor for solved problems. The bottleneck of the FETI methods is also the application of the projector $Q_G = G^T(GG^T)^{-1}G$ dominating to the solution.

The main objective of this subtask is the scalability analysis and bottlenecks' identification of the **Elmer solvers** developed at CSC Helsinki and **FLLOP solvers** developed at IT4I at VSB-TU Ostrava.

Elmer solvers, improvement of Elmer scalability via internal FETI

For the comparison of Elmer parallel solvers of large linear system (scalar diffusion and linear elasticity problems), there are several methods and preconditioners available internally in Elmer and via interfaces to open-source libraries. After a preliminary selection, a candidate group of about 25 different combinations of solvers and preconditioners was selected (CG, BiCGStab, BiCGStab(l), FETI, multigrid and algebraic multigrid solvers from Elmer, Hypre and Trilinos packages; for preconditioners ILU0, ILU1, Parasails, multigrid and algebraic multigrid from Elmer, Hypre and Trilinos packages). Krylov methods without any preconditioning were also tried and they turned out to be surprisingly competitive in terms of wall-clock time. The benchmarks were computed on Cray Sisu.

Elmer incorporates an internal FETI solver which was implemented in cooperation of CSC with VSB and which can be used without any external library dependencies. There is also a possibility to use MUMPS as a solver to factorize the internal domains and to solve the local nullspaces algebraically and to use MUMPS with several cores to solve the coarse problem.

For the first test of Poisson problem in an angular domain discretized with standard hexahedral elements to 4M and 35M degrees o freedom and partitioned into 128 – 1024 subdomains with Metis, the best methods are clearly Trilinos CG, preconditioned with multigrid solver ML and BiCGStab, either without or with preconditioning with Parasails. For this problem, the combination of Trilinos CG and ML is almost an order of magnitude faster. What is also interesting are good scaling properties of FETI when the number of processors is increased.

For the second test of linear elasticity problem, again in an angular domain, tangential force was applied on one end while the other end was fixed, the domain was discretized with standard hexahedral elements to 12M and 95M dofs and evenly partitioned into 864, 1152, 1536 subdomains. Most methods have convergence issues, especially for the large test case. FETI with MUMPS is the fastest method for the small test case, and one of the two methods that converges for the large test case. For the large test case, the timings are of the same order of magnitude with Trilinos, but the achieved speedup is better.

For the third test of Poisson equation in a unit cube with zero Dirichlet boundary conditions , each subdomain was discretized to 8^3, 16^3, 32^3 dofs with standard hexahedral elements while the domain was partitioned into 27 – 3375 subdomains. Elmer FETI implementation exhibits good weak scaling. When the size of the system per subdomain exceeds threshold 16^3, the total solution time grows significantly.

The implementation of FETI into Elmer improved its scalability.

FLLOP solvers, improvement of Elmer scalability via interfaced FETI

For the comparison of FLLOP parallel solvers of large linear system (linear elasticity of model cube and of engineering car engine), there are several solvers (Prim CG, Prim DCG, Prim DD, Dual DD-TFETI). Benchmarks were computed on Cray HECToR.

The role of the CP in dual FETI methods propagating the error globally is played by its analogue in DCG method consisting in the solution of addtitional coarse problem of the form

$$W^T\overline{K}Wx = y = W^T\overline{K}r$$

where the columns of the matrix W are a basis for the deflation subspace. For larger coarse problems then SuperLU_Dist parallel solver running on subcommunicators is used. The simplest way of defining this mapping W for a mesh-based system of equations is by agglomerating the nodes of the mesh into subdomains and then defining a polynomial reconstruction over each of them. In our case the entries in W are unity for the dofs of this region, and zero for all other dofs. The dofs with Dirichlet b.c. have to be excluded from the W construction. At the algorithmic level it can be viewed as substracting one more (low-dimensional) search direction Wx from the original CG search direction, so that $p = r - \beta p - Wx$

For the first test of elastic cube discretized to 17M dofs and regularly partitioned into 512, 1000, 4096, 8000 subdomains, the best method up to 1000 cores is TFETI, although for larger number of cores TFETI has the best solution times, the total times because of needed preprocessing are little bit worse than for standard CG applied to undecomposed problem.

For the second test of elastic car engine discretized to 2.5M dofs and partitioned into 102, 1014 subdomains with Metis, the best method without any doubt is TFETI, for 1014 cores it is 10 times faster than the second PrimDD method. Furthermore the scalability of TFETI for this engineering problem is almost perfect.

In both libraries, the FETI solver was identified as the best. Because the performance of that implemented in FLLOP seems to be better, the Elmer-FLLOP interface was created. **Interfacing FLLOP by Elmer improves scalability of Elmer** especially through the CP solution using SuperLU_DIST running on subcommunicators of size N_r/N_c , so that N_r cores are doing redundant work, having so significantly better scalability for the CP than MUMPS in Elmer and furthermore on-going development of other ingredients improving the performance of FETI. For the numerical experiments the cube benchmark was chosen. Machine used for these computations was Sisu (Cray XC30) at CSC.

Content of the website

An introduction to solvers of sparse linear systems of equations is given, including the class of FETI (Finite Element Tearing and Interconnecting) based methods. Additionally an overview to the class of solvers used in the code ELMER is given, which include FETI based ones.

The novel software package FLLOP (FETI Light Layer On top of PETSc) is presented and explained, as well as its interface with the code ELMER. Weak and strong scalability measurements of the Elmer-FETI solver and the FLLOP-FETI interface are demonstrated.

7.2 Code_Saturne

Introduction

Code_Saturne is a multi-purpose CFD software developed by EDF since 1997 and open-source since 2007. The Finite-Volume Method is used to discretise the equations on meshes made of any type of cells. Code_Saturne is distributed under GNU GPL licensing and written in C, Fortran90 and python. MPI is used for communications and OpenMP features have recently been added to the software. Code_Saturne is highly portable and is one of two CFD codes selected for PRACE 2IP benchmark suite.

Four different tasks are dealt with within WP8, i.e. mesh multiplication to generate very large meshes (VSB), code testing at scale (pushing the limit of the algorithms used in the code) (STFC), testing the existing Neumann preconditioners and implementing Chebychev preconditioners (AUTH), and implementing a deflated conjugate gradient to solve the pressure equation (STFC).

The code can be downloaded from http://research.edf.com/research-and-the-scientific-community/software/code-saturne/download-code-saturne-80059.html and a general documentation is available from http://en.wikipedia.org/wiki/Code_Saturne where other links are also available. This information is displayed on the main page of the WIKI.

Tasks

The description of the four above mentioned tasks are found under Workplan and Status of the WIKI, where 4 pages have been created, containing the following items:

1. A Mesh Multiplication capability, which has been developed, integrated in Code_Saturne and tested for hexahedral cell meshes up to 26 billion cells.
2. The code has been ported to STFC Daresbury Laboratory's Blue Gene/Q and the performance of all the algorithms have been tested, with a special focus on the existing linear solvers. The current solver used for the pressure relies on an algebraic multigrid approach. The largest test has been run for one time-step for a 26 billion cell mesh. Several configurations have been run:
   - Case 1: 4096 nodes, 2 MPI tasks
   - Case 2: 6144 nodes, 2 MPI tasks
   - Case 3: 6144 nodes, 2 MPI tasks and 8 OpenMP threads.

   A good speed-up of about 7.5 (ideal would be 12) is observed between Case 1 and Case 3. Note that 6144 nodes correspond to the six racks available on STFC Daresbury Laboratory’s Blue Gene/Q. No test on a larger number of nodes has been conducted on JUQUEEN yet, but this will be done before the end of the project if a sufficient compute time budget is made available.
3. In order to speed-up the pressure resolution, several preconditioners have been tested on two CFD cases, the former to simulate a flow in a bundle of square tubes and the latter to study thermal fatigue in a pipe. The Neumann preconditioners are already implemented in the code and the Chebychev preconditioners have been implemented there, using the power method to get the dominant eigenvalue of the pressure matrix. First results show that for both types of preconditioners, the number of iterations reduces as the polynomial order increases, but also that more time is required for the whole calculation to complete, as more matrix-vector multiplications are needed. More tests are going to be conducted before the end of the project and their outcome will be added to the WIKI.

4. Expecting the current multigrid solver not to be able to scale on a large number of processors, a deflated conjugate gradient solver, similar to the one existing in Alya has been implemented in Code_Saturne. The Lapack library has been linked to the code in order to be used to directly solve the low frequency modes. Testing the whole algorithm is currently underway and results will be added to the WIKI before the end of the project.

Content of the website

The main page is found at
https://hpcforge.org/plugins/mediawiki/wiki/codesaturn/index.php/Main_Page
leading to 3 subpages, namely, Status, Documents and Downloads. Websites and WIKIs for Code_Saturne already exist and references to the official Code_Saturne's webpages are given in Documents and Downloads. Clicking on 'Status' leads to the work performed in WP8, with the following subsections:

- Implementing/testing a Mesh Multiplication in Code_Saturne,
- Testing the algorithms at scale
- Testing and implementing various preconditioners to help solving the pressure Poisson Equation
- Implementing/testing a deflated conjugate gradient solver for the pressure Poisson Equation.

The first section describes the work completed by VSB, the second one by STFC, the third one by AUTH and the last one by STFC.

7.3 Alya

Developed at BSC-CNS, Spain

http://www.bsc.es/computer-applications/alya-system

The Alya System is a Computational Mechanics code with two main features. Firstly, it is specially designed for running with the highest efficiency standards in large-scale supercomputing facilities. Secondly, it is capable of solving different physics, each one with its own modelisation characteristics, in a coupled way. Both main features are intimately related, meaning that all complex coupled problems solved by Alya must retain the efficiency. Alya is organized in modules that solve different physical problems. Among the modules included in the code are:

- incompressible/compressible flows,
- non-linear solid mechanics,
- species transport equations,
- excitable media,
- thermal flows,
- N-body collisions,
- electro-magnetics,
- quantum mechanics,
- Lagrangian particles transport.

The module involved in this work solves for incompressible flows.

The work carried out in WP8 was twofold. On the one hand, the performance of iterative solvers has been studied deeply. On the other hand, the team has been developing a surface correction algorithm for the mesh multiplication algorithm implemented in Alya.

Solvers

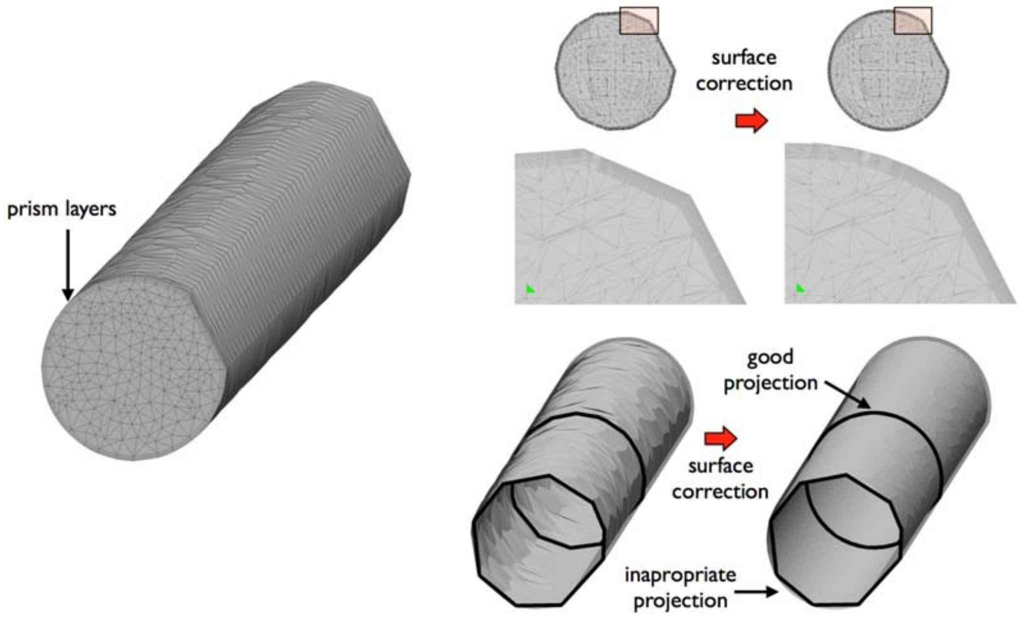
The team has implemented a new solver (Schur complement solver for the Poisson equation), new preconditioners (Block LU, one-level multigrid), renumbering techniques (postordering), and studied deeply the performance of an already existing solver in the code, the Deflated Conjugate Gradient. As far as this last topic is concerned, we focused on the following points: Sparse direct solver for the coarse problem; construction of the groups of the coarse problem; “weak” scalability.

- Preconditioners for symmetric/unsymmetric systems:
  - Deflation for BICGSTAB/GMRES: The Deflated Conjugate Gradient (DCG) was implemented as a preconditioner to be used for non-symmetric systems, together with Jacobi and Gauss-Seidel smoothing. It was tested for Fluid and Solid mechanics problems. The conclusion is that a certain gain in iteration numbers can be obtained but the gain in CPU is low.
  - Block LU: Block LU preconditioner has a more severe behavior. In current implementation, blocks are directly inverted in each CPU. Obviously, convergence depends greatly on the number of CPU's. However, in the case analyzed, even though the gain in number of iterations is very high; the CPU cost is also much higher than a simple diagonal solver. However, this preconditioner can be very useful for ill-conditioned matrices.
- Schur complement solver: A Schur complement solver was implemented for symmetric problem. The solver was tested against the DCG on 1920 CPU's and performance was found much lower for the Schur complement solver with respect to classical CG.
- Deflated CG:
  - Performance: Weak scalability test was performed for a 500M-element mesh, showing the dependence of the convergence and CPU time upon the number of groups.
  - 2 communication strategies: Two different communication strategies were studied for communicating the RHS of the coarse problem. One is based on an AllReduce and the other on an AllGatherv. No sensible difference was found in the scalability of the solver.
  - Strategies for constructing groups: Two strategies for constructing the groups were compared: the first one consists in constructing the groups using an advancing front strategy, independently of the subdomain partition; the other one mimics the partition, choosing one group per subdomain. The first option leads to better convergence while the cost per iteration is exactly the same.
- Direct sparse solver:
  - Post-renumbering: A post-renumbering strategy was implemented when a Sparse direct solver is required. This is the case of block LU preconditioner, Deflated CG and Schur complement solver.

Surface correction for MM

One inconvenience of the MM strategy is that it does not preserve curvature. Indeed, at each multiplication level, new nodes are added on edges and faces (and on centres for hexahedra) by interpolating linearly the coordinates of the previous mesh level. Therefore, a surface correction may be needed. Two options are available, based on geometry reconstruction techniques, or by using a fine STL surface description on which to project each level of the MM algorithm. We considered in this work this last technique. The advantage of this technique is that it does not require user based or arbitrary decisions and is quite automatic. The drawback is that, as it does not recognize geometrical patterns (like corners), the projection onto the surface can fail, as shown in next Figure. The technique consists of the following steps:

1. Generate the coarse mesh (from CAD) and a very fine boundary/STL mesh of the geometry surface;
2. Carry out the MM in parallel;
3. On the multiplied mesh, compute the minimum distance d_{min} of the boundary nodes to the very fine STL mesh;
4. Carry out a Laplacian smoothing by using this minimum distance d_{min} as a Dirichlet condition.



Shortcoming of the surface correction for the mesh multiplication

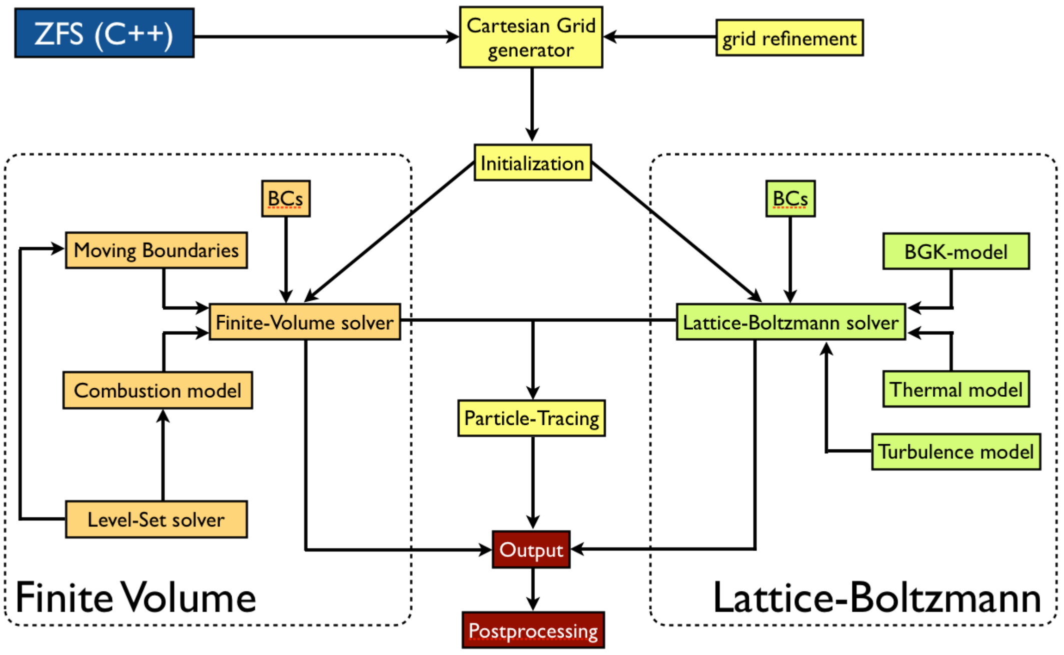
Content of the website

In the website, more information is available on the results achieved during the project, for both the solvers and the surface correction for the mesh multiplication. The results are illustrated with scalability tests and figures. Associated references can also be found.

7.4 ZFS

Introduction

The code ZFS is developed by a group of the Institute of Aerodynamics, RWTH Aachen University. It is a modular flow solver by means of the possible use of different simulation methods within the same simulation run. The code ZFS is written in C++, following the object-oriented programming paradigm.



Modules and their interaction within code ZFS

Problem addressed in WP8

The code ZFS was chosen to be part of the engineering community support of WP8 in PRACE-2IP. With the help of the code developers, the topic of parallel mesh generation has been identified as a subject with improvement opportunity.

Parallel mesh generation is used within code ZFS during grid generation and grid refinement. The generation of a Cartesian grid itself is straight-forward if you know the distribution of the level of detail you would like to have in a computational region. Amongst other things, this distribution depends on the geometry, which defines the computational region by means of boundaries.

This leads to three phases of the Cartesian grid generation:

1. Creating cells with high level of detail along the boundaries defined by the geometry.
2. Filling space with cells by relaxing the level of detail between the geometry boundary cells.
3. Determine which cells are inside the computational region and discard the cells outside of it.

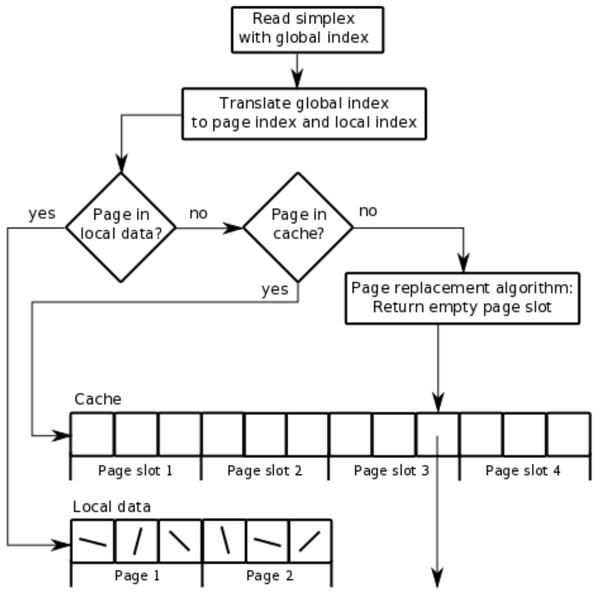
The Cartesian grid generation can be done in parallel by simple region decomposition. As long as the full geometry can be hold by every distributed memory node, there is no communication needed between nodes within phase one and three of the Cartesian grid generation process.

But with growing problem sizes, the geometry data size may very well exceed the available memory on a node. Or at least may be unreasonably large regarding the need to keep the data in memory for mesh refinement during the solution process. So it should be possible to distribute geometry data along the nodes. The aim of the engineering support in WP8 of PRACE-2IP is to develop and implement a method for distribution of and access to geometry data, which minimises the latency caused by access to distributed geometry data.

Work done in WP8

In WP8 the mesh generation code part which accesses geometry data has been analysed. A major part of geometry data access related code deals with the determination which cells of the computational mesh are inside in which are outside of the computational region. This is done by shooting a ray from the respective cell and counting the number of simplex (geometry) hits. By organising the geometry data layout in memory according to an Alternating Digital Tree (ADT), the geometry data can yield spatial locality properties in memory.

By storing mesh cells of the computational region in memory along a space filling curve, we also induce a spatial locality property to the mesh cell data. Regarding the property of spatial locality of both the mesh cell data and the geometry data, the idea of using a paging system emerged in WP8, which would exploit the spatial locality properties. Assuming a temporal locality by the mesh generation algorithm, the paging system can be extended to a caching system, which uses a page replacement algorithm in order to decide which copy of a page can be discarded within a space limited cache memory.


Read simplex with global index
Translate global index to page index and local index
Page in local data?
yes
no
Page in cache?
no
yes
Page replacement algorithm: Return empty page slot
Cache
Page slot 1
Page slot 2
Page slot 3
Page slot 4
Local data
Page 1
Page 2

The idea seemed only to be feasible if asynchronous communication is possible. This can be accomplished using one sided communication. That leads to the process of evenly distributing geometry data across the computational nodes and reserving cache memory on every node. The diagram to the right shows the mode of operation of the cache system on a node when an access to geometry data occurs.

A first implementation using MPI one-sided communication yielded bad bandwidth results, the current implementation using SHMEM on a Cray XE6 yields better results, for example the geometry data size can be reduced to 0.002% of the overall size by distributing it to 512 nodes. Generating a mesh with the “nose” test case, using 35% of the overall geometry data size as the cache size, takes approximately three times more than replicating geometry data on every node.

The ADT is not distributed in the current implementation. The ADT has a size of approximately 24% of the geometry data. It is anticipated that the ADT might be distributed across the computational nodes as well.

Content of the website

The approach used in order to solve the problem of distributing geometry data over computational nodes within the Code ZFS is novel in that the author doesn’t know another parallel mesh generation code which also uses a caching/paging system across computational nodes via one-sided communication. Thus the ZFS-website at hpcforge (https://hpcforge.org/plugins/mediawiki/wiki/zfs/index.php/Main_Page) is organised as the description of this approach within ZFS.

First an overview to the code ZFS is given on the website followed by a more detailed description of the mesh generation process and data structures. It is reasoned why geometry data should be distributed across the computational nodes. A current approach in ZFS is shortly described followed by an overview to the new idea of using a caching/paging system for distributed geometry data within ZFS and the reason for this. After that, the geometry data organisation in ZFS is described followed by an overview of the resulting mappings within the paging/caching system for this data. The cache operation is stated, as well as cache design parameters and a simple cache performance metric. A short overview to the implementation of the paging/caching system within ZFS using C++ is given.

Near the bottom of the website, measurements with the "Nose" test case are presented and interpreted using a cache performance model. A short overview to possible future work on ZFS regarding the caching/paging for geometry data is given at the end.

The description of the caching/paging system on the website marks the current status of the work done for the code ZFS in this project; the paragraph about future work states the workplan for the rest of the project.

The caching/paging system has been directly integrated into the ZFS code as a new branch. The source code of ZFS can be accessed via a subversion server hosted by the Institute of Aerodynamics, RWTH Aachen University, and can be accessed if respective permissions have been granted by the Institute of Aerodynamics.

7.5 Coupled FVM Solver

Introduction

The Finite Volume Method (FVM) is a well-known method in CFD to solve the Navier-Stokes equations. In contrast to the so-called segregated approaches (SIMPLE, PISO) a coupled solver calculates the changes of both the velocity field and the pressure field in each iteration ([14], [15], [16], [17]).

We implemented a coupled solver for the lid-driven cavity case with an incompressible fluid and Cartesian mesh from scratch in order to develop and test a new coupled solver method meeting the requirements of HPC.

Reducing the data transfer between processes

After decomposing the domain we set up "virtual walls" at these domain boundaries: Velocity and pressure are kept fixed at the domain boundary for one iteration so that no data exchange between neighbouring processes is needed and a local, temporary pressure distortion arises at this domain boundary. In the subsequent iteration the corresponding halos are exchanged and velocity and pressure are allowed to relax. Due to the pressure distortion at the virtual wall the fluid is accelerated in the right direction compensating the former slowing down.

Long-range corrections

From the finite volume formulation of fluid dynamics long-range corrections can easily be deduced. In comparison with multigrid methods they need considerably less collective data transfer between the processes. We implemented them in order to accelerate the convergence. Using the "virtual walls" we are able to overlap computation and MPI communication.

Results and outlook

On the website we present first scaling tests. This work is on-going and further results of scaling and performance tests will be presented online when available.
After validating the coupled solver method and demonstrating its scalability we intend to integrate it into a PRACE code, for example Code_Saturne.

Content of the website

On http://hpcforge.org/plugins/mediawiki/wiki/fvm/index.php/Main_Page we briefly introduce the coupled solver and the lid-driven cavity. Furthermore, the pressure distortion is illustrated and first scaling tests are presented.

8 Summary

In this document, the current status of WP8 work has been summarised. The codes selected during the first six months of the work package are currently being re-designed and re-implemented with the objective of developing software tools capable of exploiting the new generations of HPC systems, based on multicore architectures and powerful accelerators. This goal could be reached only with a close synergy between the scientists (with a clear perspective of the scientific challenges and their needs), the application developers (aware of all the algorithmic details of a code) and the HPC experts (with their knowledge of innovative programming models and technological solutions). This has been accomplished by a "daily" interaction between HPC and scientific community experts, working together on the different codes, but also by means of specific workshops, where different communities and codes representatives and experts met to discuss the achieved results and the adopted solutions. This triggered also a continuous test and validation process by the communities' experts, improving the quality of the implemented software, increasing the confidence in the newly implemented algorithms and facilitating the reintroduction of the novel versions of the codes in the users' community.

A web site was specifically implemented to support this collaborative work, to organise and publish contents and software and to promote the WP8's achievement and results among users and scientists. An overview of the structure, the functionalities and the contents of the web site, built based on the SourceForge and MediaWiki technologies, has been provided in this document, together with the description of its overall architecture. In the realisation of the web site, specific care has been devoted to the selection of technological solutions that, besides being suitable to the WP8's activities, guarantee an easy preservation of its contents and its accessibility beyond the PRACE-2IP project time-life, so that it can represent a useful and effective instrument independently from the specific framework in which it was created.

The work on the different codes was planned according to specific schedules, in order to match the characteristics, the requirements and the objectives of each application. In some cases, unexpected difficulties arose, requiring important deviations from the original workplan. However, on overall, all the codes are rapidly converging to their final release, reporting outstanding results, in terms, in particular, of performance, scalability and exploitation of heterogeneous resources (e.g. accelerators).

Clear examples are identifiable in all domains. In Astrophysics, for instance, the RAMSES code was enabled to run on multicore systems, via a hybrid approach mixing MPI and OpenMP. This was a necessary step to support the next generation of cosmological simulations, requiring unprecedented computational resources. In the Climate area, the ICON code addressed the problem of exploiting the latest generation of NVIDIA GPUs, Kepler K20x and exploring different programming solutions, like OpenCL, CUDAFortran and OpenACC, in order to provide increasingly faster software to the users. In Material Science, entire parts of sophisticated packages like ABINIT and Quantum ESPRESSO, were completely re-designed in order to overcome bottlenecks found on novel computational architectures and to efficiently exploit new linear algebra libraries, like ELPA or PLASMA, this last specifically supporting GPUs. For particle physics, the PLQCD code was subject to a detailed performance analysis and extensive refactoring, supporting the hybrid MPI+OpenMP model, in order to achieve linear scalability up to the size of the largest HPC systems currently available (like BlueGene/Q machines at FZJ and CINECA). Finally, in the engineering domain, a number of topics and issues common to several codes, like effective parallel mesh generation and solver scalability, where addressed and proposed solutions implemented in codes like ELMER or ZFS.

These are only a few examples of the accomplished work, whose details together with its outcomes and results, will be reported in the final deliverable D8.3, expected by the end of the PRACE-2IP project.

The final step of WP8 was planned to be the process of validation by the scientific experts and the reintegration in the users' communities. This, however, has been, in practice, continuously performed throughout all the work package, thanks to the adopted working methodology, relying on the close collaboration and integration between researchers and computational scientists. Therefore, in the last four months of WP8, the remaining effort can be dedicated to complete and consolidate the refactoring work, producing and releasing even more reliable and stable software tools.