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 Phase Project

Grant Agreement Number: RI-283493

D8.4.1
Plan for the further Refactoring of Selected Community Codes

Final

Version:	1.1
Author(s):	Claudio Gheller, CSCS
Date:	30/11/2013

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: RI-283493**	
	Project Title: PRACE Second Implementation Phase Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: D8.4.1	
	Deliverable Nature: Report	
	Deliverable Level: PU *	**Contractual Date of Delivery:** 31/08/2013
		Actual Date of Delivery: 31/08/2013
	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: Plan for the further Refactoring of Selected Community Codes**	
	ID: D8.4.1	
	Version: <1.1>	**Status: Final**
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D8.4.1.docx	
Authorship	**Written by:**	Claudio Gheller (CSCS)
	Contributors:	Thomas Schulthess, CSCS; Fabio Affinito, CINECA; Alastair McKinstry; 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; Andrew Porter, Stephen Pickles, STFC; William Sawyer, Anton Kozhevnikov CSCS, Ioannis Liabotis and Nikos Anastopoulos GRNET
	Reviewed by:	Peter Michielse, SURFsara; Dietmar Erwin, FZJ
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	10/10/2013	First skeleton	
0.2	5/11/2013	Various inputs added	
0.3	7/11/2013	Further inputs added	
0.4	8/11/2013	First revision completed	
0.5	11/11/2013	Executive summary added	
1.0	13/11/2013	Draft ready for internal revision	
1.1	20/11/2013	Final 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 ... vi

References and Applicable Documents ... vi

List of Acronyms and Abbreviations ... vii

Executive Summary ... 1

1 Introduction ... 1

2 Selected Codes ... 3

2.1 EAF-PAMR ... 3
2.1.1 Overview ... 3
2.1.2 Workplan ... 4
2.2 Couplers: OASIS ... 6
2.2.1 Overview ... 6
2.2.2 Workplan ... 6
2.3 Input/Output: CDI, XIOS (ICHEC) ... 7
2.3.1 Overview ... 7
2.3.2 Workplan ... 7
2.4 ICON ... 8
2.4.1 Overview ... 8
2.4.2 Workplan ... 8
2.5 Fluidity-ICOM ... 9
2.5.1 Overview ... 9
2.5.2 Workplan ... 10
2.6 QuantumESPRESSO ... 10
2.6.1 Overview ... 10
2.6.2 Workplan ... 11
2.7 SIESTA ... 12
2.7.1 Overview ... 12
Work done in PRACE-2IP WP8 ... 12
2.7.2 Workplan ... 13
2.8 Exciting/ELK ... 14
2.8.1 Overview ... 14
2.8.2 Workplan ... 16
2.9 PLQCD ... 16
2.9.1 Overview ... 16
2.9.2 Workplan ... 18
2.10 ELMER ... 18
2.10.1 Overview ... 18
2.10.2 Workplan ... 20
2.11 ALYA and Code Saturne ... 21
2.11.1 Overview ... 21
2.11.2 Workplan ... 21

2.12 PFARM **22**
2.12.1 Overview 22
2.12.2 Workplan 23
2.13 RAMSES **23**
2.13.1 Overview 23
2.13.2 Workplan 24
3 Summary **25**

List of Figures

Figure 1: Sketch of the working methodology adopted for WP8 and its one-year extension (Task 4)... 2
Figure 2: Speedup factors of the parallel section when using the OpenCL code with a GPU or a multicore CPU. CPU used: Intel core2 Q6600, clock speed 2.4 GHz, number of cores: 4. GPU used: AMD HD 7970, 32 compute units (2048 stream processors) up to 925MHz engine clock.................... 4
Figure 3: diagram with the proposed timeline for the extension period.. 5
Figure 4: T799 ECEARTH3 Scaling Results on Hermit for ECEARTH3 using oasis3 (black lines) and oasis3-mct2.0 (red lines). The dashed lines show sacling results using a total processes per NEMO process ratio of 3. Similary, the full lines present results using a ratio of 6... 6
Figure 5: Performance of Siesta-PEXSI compared to diagonalisation. The example is a DNA strand of varying length. The number of processors grows proportionally with the system size. 13
Figure 6: Times per SCF iteration for solving and saving several problems with different numbers of processors. The examples are C-BN with ~13000 atoms and DNA with ~18000 atoms...................... 14
Figure 7: Left panel: hard scaling of the code with respect to the size of the MPI process grid. Red line – ideal scaling, blue line – actual scaling of the code. Green line – scaling of the ELPA eigenvalue solver used for the Hamiltonian diagonalisation. Clearly, the eigenvalue solver is the limiting factor for the good scaling. Right panel: (top) Time distribution between various parts of the code for the 144 (12x12) MPI ranks job. Generation of the Kohn-Sham eigenfunctions (green bar) takes ~95% of the run-time. (bottom) Time distribution inside the Kohn-Sham eigenfunctions generation: most of the time (yellow bar) is spent in the generalised eigenvalue problem (ELPA)... 16
Figure 8: Angular domain benchmark.. 19
Figure 9: Graph of the strong scalability (Elmer vs. FLLOP)... 20
Figure 10: GPU implementation: approach 2.. 24

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 D8.1.3: "Prototype Codes Exploring Performance Improvements"
[5] Deliverable D8.1.4: "Plan for Community Code Refactoring"
[6] Deliverable D8.2: "Refactoring and Algorithm Re-engineering Guides and Reports"
[7] Deliverable D8.3: "Re-integration into Community Codes"
[8] http://amcg.ese.ic.ac.uk/index.php?title=Fluidity
[9] http://www.mcs.anl.gov/petsc
[10] http://www.cs.sandia.gov/Zoltan
[11] A G Sunderland, C J Noble, V M Burke and P G Burke, CPC 145 (2002), 311-340.
[12] UKRmol: a low-energy electron- and positron-molecule scattering suite, http://oro.open.ac.uk/33130/.
[13] IBM Engineering and Scientific Subroutine Library (ESSL), http://www-03.ibm.com/systems/software/essl/.
[14] Eigenvalue SoLvers for Petaflop-Applications, http://elpa.rzg.mpg.de.
[15] CUDA Toolkit, http://docs.nvidia.com/cuda/cublas/index.html.
[16] Intel Math Kernel Library 11.0, http://software.intel.com/en-us/intel-mkl.
[17] The Matrix Algebra on GPU and Multicore Architectures project, http://icl.cs.utk.edu/magma/.
[18] Atomic, Molecular, Optical and Positron Physics, http://www.ucl.ac.uk/phys/amopp.
[19] Centre for Theoretical Atomic, Molecular and Optical Physics, Queen's University, Belfast, http://www.qub.ac.uk/schools/SchoolofMathematicsandPhysics/Research/CTAMOP/.

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
HP2C	High Performance and High Productivity Computing Initiative
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 Software 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
MIC	Intel Many Integrated Core architecture
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)
Xeon Phi	Processors family by Intel using the MIC accelerator

Executive Summary

This document presents the work plan for the extension of Work Package 8 'Community Code Scaling', which focuses on the re-design and refactoring of a number of codes for scientific numerical applications, in order to effectively run on coming generations of supercomputing architectures, optimally exploiting their innovative features. The achievements and outcomes of WP8 at the end of its two years programme can be considered excellent. The extension was conceived in order to further enhance the results exploiting the developed competencies, skills and synergies. A subset of codes originally in WP8 that promise further improvements will take part to the extension. Thirteen codes were selected according to the proposed objectives and the available resources. We will give a short introduction to each of the codes, summarizing the main accomplishment in the framework of WP8 and describing the targets for the extension together with the details of the related work plan.

1 Introduction

Work Package 8 (hereafter WP8) was designed with the idea of supporting science by enabling numerical applications and simulation codes for coming generations of High Performance Computing (HPC) architectures. This was accomplished by involving a number of scientific communities in the process of enabling some of their most relevant simulation codes on innovative supercomputing architectures. Such process relied 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 program in order to optimally map applications to emerging supercomputing architectures. An equally important objective was to transfer HPC skills into the communities.

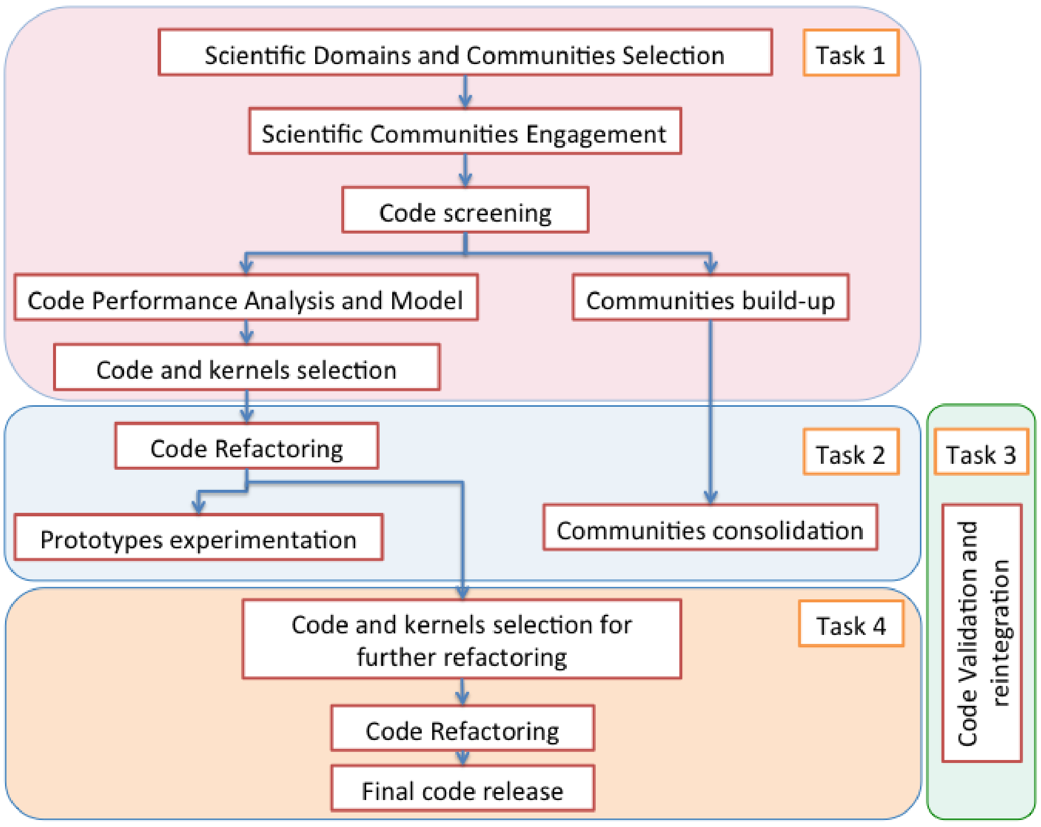
Scientific Domains and Communities Selection
Task 1
Scientific Communities Engagement
Code screening
Code Performance Analysis and Model
Communities build-up
Code and kernels selection
Code Refactoring
Task 2
Task 3
Prototypes experimentation
Communities consolidation
Code Validation and reintegration
Code and kernels selection for further refactoring
Task 4
Code Refactoring
Final code release

Figure 1: Sketch of the working methodology adopted for WP8 and its one-year extension (Task 4).

During the initial part of WP8 a number of codes have been analysed, their performance evaluated, their suitability to the refactoring program investigated, by deriving an analytical model for the anticipated performance (the “performance modelling” approach). As a result a number of codes were selected from five different scientific domains: Astrophysics, Material Science, Climate, Engineering and Particle Physics. Specific goals and work plans have been defined for each code, and working groups were formed composed of different project partners and community members dedicated to each code. Due to the different features, needs and targets of the codes, each group adopted a different working methodology, strategy and schedule in order to reach the envisaged objectives in the WP8 time frame.

Overall, excellent results in many of the constituent applications have been achieved. Such results can be further improved by exploiting the experience, expertise, technological instruments and collaboration acquired and built along the work package life span.

This will be possible thanks to the one year extension (Task 4) during which the WP8 re-design and refactoring programme will be continued, adopting the working methodology successfully used in the first two years of the project. Figure 1 shows how Task 4 is integrated in WP8 as a natural extension and improvement of the previous work.

During Task 4 we will focus on a number of “success stories,” identifying qualified groups of interested PRACE-2IP partners and community experts, in order to further enhance the quality, the effectiveness and the efficiency of the developed software and its readiness for the new HPC architectures that the PRACE-RI will deploy as Tier-0 or Tier-1 systems. The codes have been selected according to the achieved outcomes of the first two years, the proposed objectives, and the available resources. The codes selected for further refactoring are listed in Table 1.

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

Table 1: List of the codes selected for Task 4 (left column), corresponding scientific domain (central column) and responsible PRACE partner (right column).

In this document, for each of the selected code, we will give a short description of its main features (details can be found in deliverables [2] and [3]), a summary of the work accomplished during the first two years of WP8 (described in [4], [5], [6] and [7]) and the objectives and the work plan for Task 4. The deliverable will be completed with a summary of the envisaged work.

2 Selected Codes

Codes selected for WP8 have been chosen according to the procedure described in [3], [4] and [5]. A few of these codes will be part of the one-year extended refactoring program. These codes have been selected according to the achieved results, the proposed work plan and the available effort. For these codes, in the following section, we will summarize the main accomplishments of the first two years and we will describe the work plan for the third year.

2.1 EAF-PAMR

2.1.1 Overview

One of the main objectives for WP8 was the creation of a grid based hydrodynamical three-dimensional code that would make use of the most recent advances in computation technologies, namely Graphical Processor Units (GPU) and other types of acceleration devices (for example, the Intel Xeon phi board). In order to do so, the OpenCL platform was chosen as the base for this work since it presented a high level of development and support and, more importantly, it was a truly heterogeneous platform, meaning that the code could not only use the above mentioned devices but still retain the capability to use traditional CPUs and, therefore, the capability to be used in the already existing supercomputing solutions.

In terms of software development it was decided to create two different solutions to be shared with the community: a complete standalone code and a library of hydrodynamical functions using OpenCL. This was decided due to a couple of factors: first it is much easier (and advisable) to develop an OpenCL code from the ground up. This helps ensure the best performance since all the elements are set from the beginning in order to achieve the required performance. On the other hand releasing a library of functions with the main hydrodynamical methods would allow the adaptation of existing codes (that sometimes feature other advanced algorithms to simulate specific physical elements) in order to use OpenCL with only a much smaller amount of work by their developers.

During the development stage of the WP8 the workflow consisted in first developing the standalone code first and ensure that it was working correctly and only after that convert the algorithms into the library. Using this approach we can achieve a virtually simultaneous development of both codes.

At the end of the WP8 period the following objectives were achieved:

Standalone code:

1. Implementation of the Roe and HLLC Riemann solvers.
2. Implementation of the Piecewise Parabolic Interpolation Method (PPM).
3. Domain division (to allow the use of multiple computation devices).

Function library:

- Development of functions to setup the OpenCL library.
- Implementation of function to apply the Roe and HLLC Riemann solvers, with and without PPM interpolation.

Regarding performance, figure 1 shows speedups obtained (considering the parallel section of the code) when using a GPU or a multicore CPU using the time spent by a single CPU core as a baseline. The benefits of using GPUs are obvious when we consider that we obtained a speedup factor of about 60. Also to consider that the speedup factor when using all the cores of a quad-core CPU is very close to 4. Also since both the standalone code and the library share most of the code used in the calculations itself, these results also apply to the performance of the library.

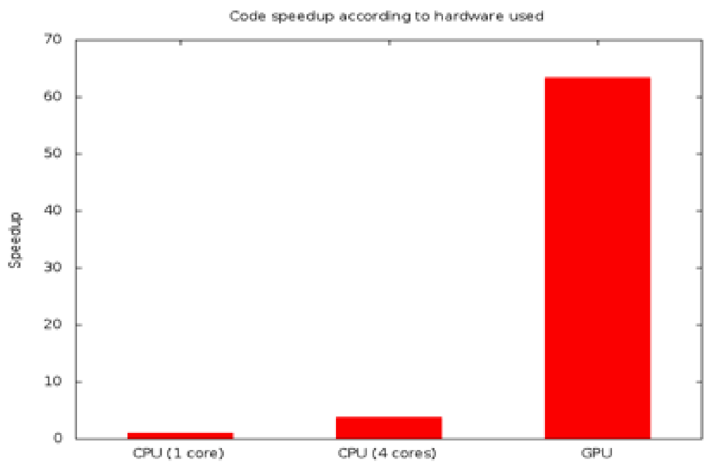

Figure 2: Speedup factors of the parallel section when using the OpenCL code with a GPU or a multicore CPU. CPU used: Intel core2 Q6600, clock speed 2.4 GHz, number of cores: 4. GPU used: AMD HD 7970, 32 compute units (2048 stream processors) up to 925MHz engine clock.

Main Objectives for the extension period

The main goals for the extension period can be divided in terms of scientific and HPC objectives. The scientific objectives are defined as: Implementation of additional methods in the OpenCL code with the inclusion of magnetohydrodynamical algorithms and a self-gravity solver. The other main scientific objective will be the inclusion of secondary heating processes in the Plasma Emission Code. This effect could have a significant impact in the evolution of, for example, the Interstellar medium. The addition of this process to the PEC and subsequent addition to the OpenCL MHD code will make this one of the most capable codes in terms of simulating the thermodynamical evolution of a gas.

The HPC work will focus on optimising and testing the performance of this code in several kinds of platforms. Special care will be given to the use of the recently released Intel Xeon phi accelerator board and the performance comparison of a OpenCL code and codes that use traditional parallelisation tools (OpenMP, MPI) when using this device.

2.1.2 Workplan

For the extension period of the WP8 the following objectives were defined:

- **Extend the code to magnetohydrodynamics (MHD).** The existence of magnetic fields in many physical problems can be a very important element in their evolution and study, therefore the need for algorithms that can simulate fluid dynamics while in the presence of magnetic fields is very high. Therefore a natural evolution of the work done so far with the OpenCL hydrodynamics code would be to include these effects in the algorithms used with special care to include MHD methods that use the PPM interpolation to ensure the code possesses the accuracy needed to be relevant as a research tool when simulating complex problems. Although much of the ground work done for the development of the hydrodynamical algorithms can (and will) be used as a base of work, the higher complexity of the MHD algorithms will require significantly reworking of some elements of the code, specifically the memory structures used. As a complementary goal a Poisson solver written in OpenCL will be developed as it can serve two purposes; first it can be used to calculate self-gravity in the problem and second it can be used as a method to clean the magnetic field divergence, a process that is required for some MHD algorithms.

- **Optimisation**. Although the heterogeneous capabilities of OpenCL are a very strong advantage of this platform it also raises the problem that the code must be optimised for all kinds of devices capable of running the code. This optimisation factor will focus essentially in three different devices: GPUs, CPUs and the Intel Xeon phi (with a special attention to this device). This effort should allow us to determine whether specific versions of the code should be developed for each platform or whether there is a single optimisation that is sufficient for heterogeneous use. With these optimisations we should also be able to compare the performance of OpenCL with other parallelisation paradigms and determine the true viability of OpenCL in the field of scientific computation.
- **Hybridisation**. The code in its present state can divide the problem space in several smaller subdomains. This is required since most problems that are simulated have a considerable size, and therefore memory requirements, much larger that what any current GPU memory. To solve that problem the full problem domain is divided amongst several GPUs with each one only solving its corresponding subdomain. This will require that the commands that prepare and launch the OpenCL routines must also be executed in parallel and that will imply using a second parallelisation tool such as OpenMP or MPI, although this second layer of parallelisation is much simpler that the work done with OpenCL, it is essential to ensure a highly effective parallel code.
- **Secondary heating.** An extension to the Plasma Emission Code, this will simulate the energy inserted into the gas either through ionisation, excitation or heating by the secondary electrons released by photo-ionisation by highly energetic X-rays. This could be a very important factor in the thermal evolution of the plasma and has not been completely considered in interstellar medium evolution studies. For this goal we will implement an algorithm to calculate this effect, add it to the Plasma Emission Code already developed and then integrate the full Plasma Emission Code with the new MHD code.

The work plan is summarized by Figure 3

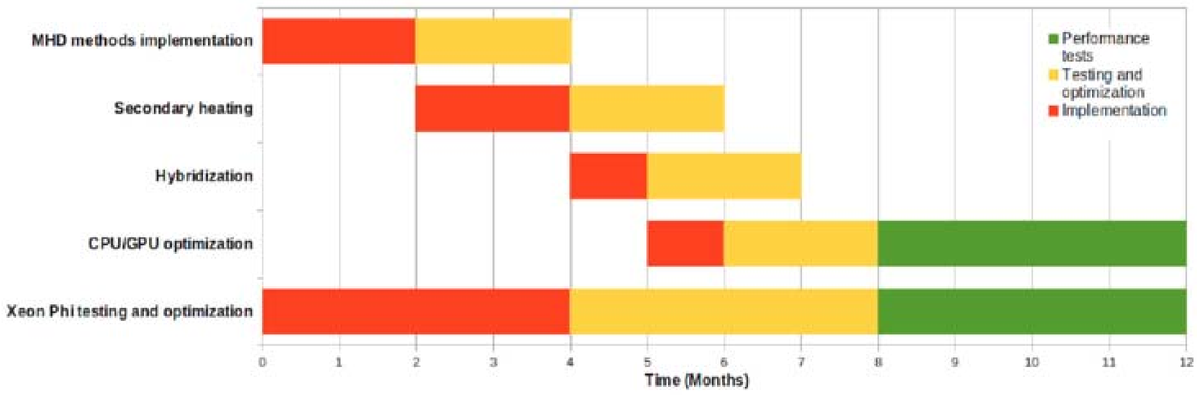

Figure 3: diagram with the proposed timeline for the extension period

2.2 Couplers: OASIS

2.2.1 Overview

The main achievements of work done in WP8 have been scaling and debugging of OASIS3-MCT, and testing within the EC-Earth3 framework. This now works on at least 4000 MPI processes, as tested on Hermit using the EC-Earth3.1 development code. Scaling results for a two-day high-resolution simulation are presented in Figure 4. OASIS3-MCT2.0 shows a clear speedup over OASIS3 for large core counts. A ratio of three total processes per NEMO processes was optimal. Bugfixes found during development are now part of the OASIS3-MCT2.0 mainstream release.

OASIS3-MCT2.0 now scales successfully to over 4000 MPI processes on its standard test cases, run on Hermit (HLRS), overcoming low-memory issues seen in OASIS3. However when run within EC-Earth 3.1, coupling IFS 36r1 and NEMO 3.3.1 atmosphere and ocean models, crashes are seen at these process counts. A separate study has also shown a hard limit of ~3600 processes for the T1279 configuration on IFS.

Hybridisation EC-Earth does not currently contain the changes made to NEMO by STFC to allow OpenMP parallelisation on either vertical or horizontal domains. Including this work within EC-Earth should allow scaling of EC-Earth to over 100,000 cores.

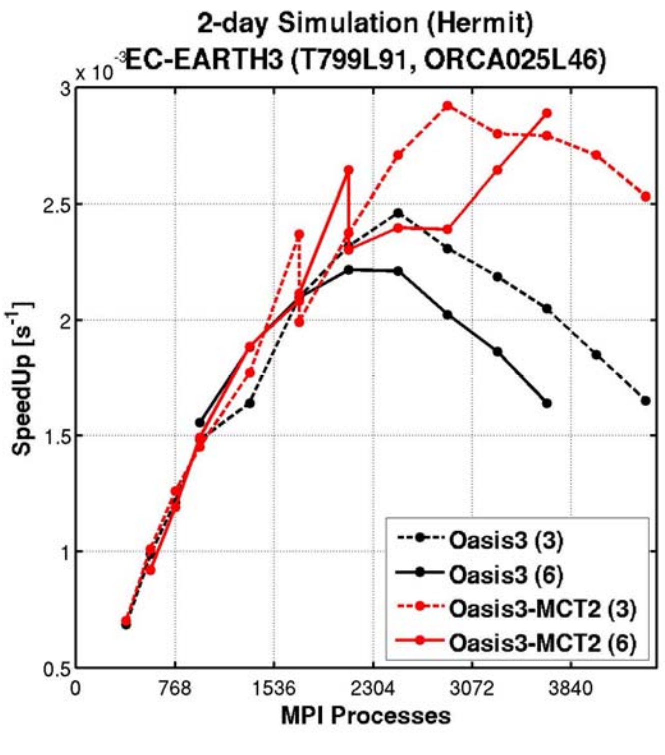

Figure 4: T799 ECEARTH3 Scaling Results on Hermit for ECEARTH3 using oasis3 (black lines) and oasis3-mct2.0 (red lines). The dashed lines show sacling results using a total processes per NEMO process ratio of 3. Similary, the full lines present results using a ratio of 6.

2.2.2 Workplan

The OASIS3-MCT2.0 changes are now in the EC-Earth3.1 development tree, but have not been merged with NEMO changes. Planned work for the extension is:

- Merge with the NEMO 3.5 changes for XIOS communications (already completed October 2013).
- Investigate the crashes at large core counts.
- Time permitting, merge SFTC NEMO changes for OpenMP hybridisation.

2.3 Input/Output: CDI, XIOS (ICHEC)

2.3.1 Overview

The aim of this work is to produce components for a unified I/O server for the climate community, capable of working "everywhere" and on multiple models, writing both GRIB2 and NetCDF. At a meeting of the climate community in Hamburg, 2011, a number of bottlenecks were identified.

There is no agreed API for a unified I/O server. The two most actively developed interfaces are the XIOS and CDI interfaces. CDI scales well for GRIB output, while XIOS scales well on NetCDF, and the XIOS transport layer is optimal on generic hardware, while CDI has a simpler design but relies on properly implemented single-sided MPI communications. In addition, XIOS has additional post-processing facilities that make it desirable. However, XIOS was understood to have issues on low-memory nodes for large configurations. Hence the following workplan was agreed:

- A CDI-XIOS interface, to allow CDI-based models to use the XIOS IO Server
- A CDI implementation for the IFS model, used within EC-Earth
- A "Memcache" buffer layer for XIOS to allow XIOS work on low-memory cache nodes.

The EC-Earth v3.1 code would be then used as a testbed for these components. As NEMO, the ocean component of EC-Earth, uses XIOS (as of version 3.5), this would create a unified I/O server for all components, which other climate models could use.

As reported, IFS cy36r4 was refactored to allow new I/O interfaces, and a CDI API was implemented, using the serial CDI implementation with data gathered on rank 0; this is identical to the parallel API under development at DKRZ, which was not ready at the time.

The memcache layer was added to XIOS. This has been validated on Hermit using existing testcases.

2.3.2 Workplan

XIOS Memcache The work has since been ported to the latest releases of XIOS (v477) and the NEMO ocean model (v3.5 which uses XIOS) for testing. This configuration has revealed bugs which are under investigation.

Following discussions with XIOS developers, it is agreed that optimal implementation for GRIB writes would include the coalescence of I/O writes at the memcache layer. In such an implementation, the memcache code would be optimised to allow clients to write to a single memcache node (rather than each client to multiple memcache nodes) and minimise MPI message pressure, seen to be an issue at high node counts by CEA. A workshop with the XIOS developers is planned at ICHEC in Dublin in December on this, at which the work required will be assessed in detail. Time permitting, this change will then be implemented within WP8.

IFS In a follow up IS-ENES2 meeting in Hamburg, October 2013, it was apparent that CDI-based models would not use XIOS, as they have since optimised post-processing within the model itself. Hence this component has lost its value. and the IFS->CDI->XIOS approach would be suboptimal. Hence the IFS atmosphere model of EC-Earth is being modified to call XIOS directly rather than using the CDI-XIOS interface, with the CDI interface being used in the XIOS I/O code to write GRIB2 from XIOS.

The XIOS interface for IFS will be completed. This is then tested with the EC-Earth v3.1 (IFS cy36r1) and the ARPEGE configuration (cy37t1) from Meteo-France; code will be committed to the EC-Earth branch of IFS/ARPEGE and hence merged back to the community.

CDI Writing GRIB2 via CDI will be implemented in the server component of XIOS. This is being designed now.

2.4 ICON

2.4.1 Overview

The Icosahedral Non-Hydrostatic (ICON) model is a climate model jointly developed by the German Weather Service (DWD) and the Max Planck Institute for Meteorology (MPI-M). It is intended to replace the popular ECHAM model in the next years. Broadly speaking, ICON consists of a non-hydrostatic dynamical core (NHDC or “Dynamics”), which calculates the motion of the atmosphere, and physical parameterisations (“Physics”), which calculate the influence of phenomena occurring on a sub-grid scale. ICON employs an icosahedral mesh (and its hexagonal-pentagonal dual) as well as static grid refinement to enhance resolution in geographical areas of interest.

In the WP8, a test-bed, distributed-memory version of the ICON NHDC was rewritten for accelerators using the OpenACC standard for accelerator directives. After considerable optimisation effort, the performance results [6] indicated a factor 2x increase in performance over dual-socket Intel Sandybridge nodes for NHDC configurations of interest (where the GPU memory is almost fully exploited). These results were entirely in line with the NHDC performance model proposed in [5]. The test-bed NHDC supported only a single resolution, i.e., no locally enhanced regions.

While the NHDC test-bed version is of central interest for benchmarking new architectures – it particular with GPUs -- it has limited applicability to the actual ICON development trunk, since the former required numerous code changes in addition to OpenACC directives, and the latter has moved on considerably since the testbed was split off the trunk 1-1/2 years ago. In the WP8 continuation, we plan to introduce the modifications from the test-bed into to actual ICON development trunk, along with mechanisms to validate the accelerated code with respect to a version run on CPUs. The ultimate goal would be to have an accelerator-enabled development trunk, which can be run and validated in dynamical core-only mode, also with local refinement to the grid.

This goal turns out to be ambitious for a number of reasons. First, we do not have large freedom to modify and refactor the code in the development trunk: fundamentally only new OpenACC directives are allowed. Fortran derived types are used extensively in the ICON NHDC, and the support for these in the current OpenACC standard is insufficient. The test-bed NHDC circumvented this limitation by created a set of “shadow arrays” for the derived type members needed, a solution not acceptable for the trunk. From initial discussions with the OpenACC consortium, it is clear that support for the *deep-copy* of derived types must be implemented in OpenACC compilers to allow for a clean realisation within the trunk.

The definition and realisation of the functionality needed for the trunk implementation will be done in close collaboration with the OpenACC consortium, and the team at Cray developing the CCE programming environment. Furthermore, we will continue a strong collaboration with DWD and MPI-M to work toward this common version of the development trunk.

2.4.2 Workplan

In order to realise the goal described in the previous section, we propose a number of concrete tasks:

- One-week visit to DWD German Weather Service (responsible: Guenther Zaengl) to coordinate code integration using a branch of the current trunk, plan the implementation and propose a validation strategy. This visit took place Jul. 22-26, 2013.
- Participation in the OpenACC consortium, and presentation of the OpenACC needs of the ICON NHDC, in particular for derived types. The OpenACC meeting took place Sep. 24-26, 2013.
- Evaluation of the current implementation of the Cray CCE deep copy to/from the accelerator, feedback to the Cray developers (James Beyer, David Oehmke, Jeff Sandoval). Currently underway (Oct. 2013). Subsequent evaluations of future CCE releases.
- Put in place validation framework to compare results on accelerator to those from a single CPU node; ongoing (since Oct. 2013).
- Augmentation of NHDC kernels to utilise OpenACC in the framework agreed upon with DWD. Timeframe: Nov. 2013 – Feb. 2014.
- Augmentation of ICON communication modules to utilise OpenACC. Timeframe: Jan. – Apr. 2014.
- Performance optimisation and tuning. Timeframe: Apr. – Jun. 2014.
- Periodic merging of branch and trunck.
- Final NHDC performance evaluation, dissemination of results. Final update of the trunk. Timeframe: summer 2014.

For this effort 0.25 FTEs (William Sawyer, CSCS) are foreseen until Aug. 2014.

2.5 Fluidity-ICOM

2.5.1 Overview

Fluidity-ICOM is built on top of Fluidity [8], an adaptive unstructured finite element code for computational fluid dynamics. It consists of a three-dimensional non-hydrostatic parallel multiscale ocean model, which implements various finite element and finite volume discretisation methods on unstructured anisotropic adaptive meshes so that a very wide range of coupled solution structures may be accurately and efficiently represented in a single numerical simulation without the need for nested grids. It is used in a number of different scientific areas including geophysical fluid dynamics, computational fluid dynamics, ocean modelling and mantle convection. Fluidity-ICOM uses state-of-the-art and standardised 3rd party software components whenever possible. For example, PETSc [9] is used for solving sparse linear systems while Zoltan [10] is used for many critical parallel data-management services. Both have compatible open source licenses. Python is widely used within Fluidity-ICOM at run time for user-defined functions and for diagnostic tools and problem setup. It requires in total about 17 other third party software packages and use three languages (Fortran, C++, Python). Fluidity-ICOM is coupled to a mesh optimisation library allowing for dynamic mesh adaptivity.

In the previous work in WP8, CG, CV matrix assembly kernels have been threaded, Most effort has been spent in optimising the threaded sparse linear solver. Several scalability bottlenecks have been identified and fixed. The matrix assembly kernels can scale well up to 32768 cores. Although pure MPI runs faster up to 2048 cores, due to the halo size increasing exponentially with the number of MPI tasks, the cost of MPI communication becomes dominant from 4096 cores onwards. There the mixed mode begins to outperform the pure MPI version. This offers Fluidity-ICOM the capability to solve the "grand-challenge" problems. However, in order to enable Fluidity-ICOM efficiently to run on peta-scaling system, the I/O component has now become the major bottleneck. We are now applying the extension effort in WP8 to tackle this issue.

2.5.2 Workplan

STFC plan to spend 6 person month to work on I/O modules for Fluidity-ICOM. The current performance bottleneck in Fluidity is I/O for initialisation, initial domain decomposition and checkpointing, where a separate domain decomposition step is currently required when re-initialising the simulation. This introduces a serial bottleneck limiting the problem size that can be configured for parallel execution. Secondly, the checkpointing and results data is in an in-house format whereby each process writes its own file. This is now causing serious problems for large process counts as parallel files systems such as LUSTER are not optimised to deal with this kind of load. To resolve these problems we propose to integrate PETSc's DMPlex module. This module is able to generate the necessary halo/ghost regions at runtime from the underlying mesh topology, eliminating the need for external domain decomposition and improving start-up times. It furthermore provides native interfaces for common meshing formats, such as CGNS and ExodusII, while PETSc's native I/O interfaces, which include HDF5 capabilities, can be used to improve mesh topology checkpointing and general result output. The additional benefit of this work is that Fluidity will move to using standardised file formats, which will greatly enhance interoperability and improve research data management processes.

The detailed work plan is the following:

- Rewrite the current mesh topology kernels using PETSc DMPLEX interface, validate it with the fluidity automatic test framework. This will also replace the current separated mesh decomposition tools.
- Rewrite the halo kernels with the PetscSF interface, including halo data type, setting up halos, managing halo communication, and validate with the fluidity automatic test framework, this will avoid reading halos from files generated by the current mesh decomposition tools and generate the halos at run time. All above involves a wide range of changes in the source code of Fluidity.

Final testing and benchmarking will be undertaken with the large ocean test cases.

2.6 QuantumESPRESSO

2.6.1 Overview

The work done up to now in this work package has been impacting deeply on the Quantum ESPRESSO distribution. It made the code more flexible, algorithmic improvements more easy to be implemented and parallelisation more efficient. Already existing levels of parallelisation have been extended through the whole code, new ones have been added where they were lacking. This work has been accomplished by CINECA. Further work was carried out by ICHEC:

Petascaling through MPI refactoring: ICHEC has been working on enabling the PHonon code within the QE suite for petaflop machines by introducing a new MPI communication layer into the existing MPI communication hierarchy. This is being implemented by parallelising over so-called ‘electron bands’ within the PHonon code. Much of the communication layer has been implemented within the code and testing is currently underway for the most compute intensive subroutines within the code.

GPU enablement: ICHEC has investigated the potential of porting the PHonon code to GPUs using the PGI OpenACC compiler. Investigations have so far been carried out on the most compute intensive subroutines that are unique to the PHonon code (`incdrous.f90`) with real user ‘C60’ data sets. Compiler bugs have so far prevented any conclusive findings, but there is still scope for further investigations with test harness versions of the subroutine.

2.6.2 Workplan

One of the most important advancement in the Quantum ESPRESSO distribution during the last years was the porting of the numerical algebra kernels to GPUs. This was achieved using a specialised library, the phiGEMM, and preserving the main structure of the code, and obtaining in this way a good maintainability of the software.

The possibility of using Quantum ESPRESSO on hybrid nodes equipped with GPUs was warmly welcomed by the community of users and permitted to enlarge the number of users.

Recently, the world of hybrid solutions for supercomputing was enriched by the introduction of the Intel Xeon Phi platform that was proposed as a competitor to the GPUs as a co-processor/accelerator on a hybrid architecture. The architecture of the Xeon Phi platform is quite different from the standard GPUs. While for the latter one is forced to use a proper low-level approach (i.e. CUDA or OpenCL), Xeon Phi can be programmed by using more standard approaches such as MPI and OpenMP. However getting good performances on the Xeon Phi is far from trivial. A strong use of vectorisation techniques together with a wise use of caching is necessary to get satisfactory results.

Stampede on the US, EURORA in the EU and Tianhe-2 in China are only a few examples of how clusters implementing Xeon Phi cards are going to be more and more wide spread around the world and increasing in relevance.

In this framework, it would be desirable to repeat the same success that we had with phiGEMM and the GPUs.

As a target for the PRACE-WP8 initiative, we propose the following workplan that will involve CINECA:

- identify a part of the code to use as a “prototype” to test the efficiency of the Xeon Phi
- use the Xeon Phi to accomplish the linear algebra calculations using offloaded DGEMM
- assess the impact of the changes on the code/maintainability respect of the gain in term of achieved performance

This work will benefit from the support of Intel who is providing an interface for the execution of the DGEMM on the Xeon Phi. The work that we are proposing consist of an implementation and testing of this interface. Our aims are:

- the assessment of the performances of the Xeon Phi
- the refactoring of the code that we intend to use as prototype to reduce the impact of the implementation

This work will permit to extend in the future the use of Quantum ESPRESSO also to hybrid nodes containing Intel Xeon Phi cards and to further enlarge the community of users.

From ICHEC side the following workplan has been defined:

Petascaling through MPI refactoring: As part of the WP8 extension, ICHEC plans to complete the implementation of the parallelisation over bands within the PHonon code using MPI. Testing is currently with relatively small test cases and so far no major speedup has been seen when running across a small numbers of processes for the given subroutines. As part of the extension, ICHEC will communicate with the QE development and user community to obtain larger test cases for further testing.

2.7 SIESTA

2.7.1 Overview

SIESTA is both a method and its computer program implementation to perform efficient electronic structure calculations and ab initio molecular dynamics simulations of molecules and solids. SIESTA's efficiency stems from the use of strictly localised basis sets and from the implementation of linear-scaling algorithms which can be applied to suitable systems. A very important feature of the code is that its accuracy and cost can be tuned in a wide range, from quick exploratory calculations to highly accurate simulations matching the quality of other approaches, such as plane-wave and all-electron methods.

The possibility of treating large systems with some first-principles electronic-structure methods has opened up new opportunities in many disciplines. The SIESTA program is distributed freely to academics and has become quite popular, being increasingly used by researchers in geosciences, biology, and engineering (in addition to those in its natural habitat of materials physics and chemistry). Currently there are several thousand users all over the world, and the paper describing the method (J. Phys. Cond. Matt. 14, 2745 (2002)) has had more than 3500 citations.

Work done in PRACE-2IP WP8

The main goal was to find a better solution for solving the eigenvalue problem in the Kohn-Sham formulation of DFT. This part takes, as its computational cost grows with the system size N like $O(N^3)$, the majority of computation time for larger systems. Currently Siesta uses ScaLAPACK, which features only limited scaling with the number of processors, and does not take advantage of the sparsity implied by Siesta's atomic orbital basis.

After examining several options, the PPEXSI library was integrated into SIESTA. PPEXSI stands for the “Parallel Pole Expansion and Selected Inversion” method, which avoids the diagonalisation by using an expansion of the Fermi operator and an efficient sparse matrix inversion. This procedure reduces the computational cost for 3D systems to $O(N^2)$, for quasi-2D systems to $O(N^{1.5})$, and for quasi-1D systems to $O(N)$. Thus this method is much more performant than ScaLAPACK for big (at least a few thousands of atoms) examples. Moreover it can, due to two levels of parallelism, use many more processors efficiently. We are able to use tens of thousands of processors for systems consisting of tens of thousands of atoms. Siesta-PEXSI proved to work reliably and to give accurate results.

Figure 5 shows the time for calculating the density matrix in one SCF iteration for various sizes of a DNA strand, with the number of processors increasing with the system size. For systems containing more than 10000 atoms Siesta-PEXSI is about two orders of magnitude faster than diagonalisation, providing the same quality of results. The time for PEXSI could even be reduced further by using more processors and also in subsequent SCF steps by reusing information from previous steps.

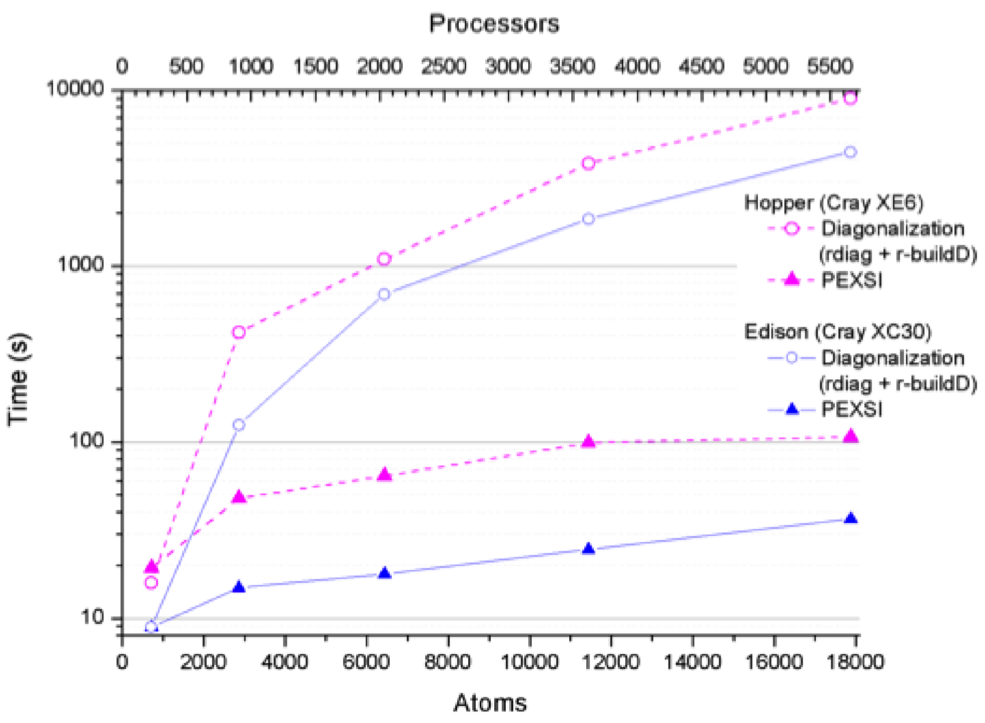

Figure 5: Performance of Siesta-PEXSI compared to diagonalisation. The example is a DNA strand of varying length. The number of processors grows proportionally with the system size.

2.7.2 Workplan

The characteristics of Siesta-PEXSI allows to deal with very large problems, using tens of thousands of cores. For such big simulations it is advantageous to have a system for checkpointing, to enable economical restarts in the event of a failure or to work around queuing systems' restrictions.

Simple checkpointing, in the form of periodic writings of the density-matrix, is already part of the code. However, it is quite inefficient for large systems, since the I/O is buffered through the master node and this creates a bottleneck. Moreover, a truly seamless checkpointing would need the saving of additional state information, such as the mixing history and some internal tables.

In order to implement a checkpointing procedure that does not impact the performance, some means of parallel I/O must be introduced

The diagram on the right hand side shows, that for a layered system (Graphene on BN) with 13000 atoms the solver scales at least to 10240 processors. But the file-IO starts to take over a large fraction of the time, limiting the scaling of the whole procedure. For a sparser system, like DNA, the situation is even worse. The solver allows to treat a system with 18000 atoms in only 35 seconds per SCF iteration, but the IO takes a multiple of that time. In these cases the numbers of processors used for the calculations in Siesta, which can be set independently from the number of processes for the PEXSI solver, have already been optimised.

The next step will be to introduce parallel IO to take advantage of the distributed data and to minimise the time spent in file-IO.

Main steps

1. Evaluate which parallel IO framework should be used.
2. Introducing data structures and parallel IO for writing the current state and mixing information to allow continuing a computation after each SCF step without

introducing differences compared to a continuous run.

3. Analysis and eventually optimisation of the performance of the new file-IO.

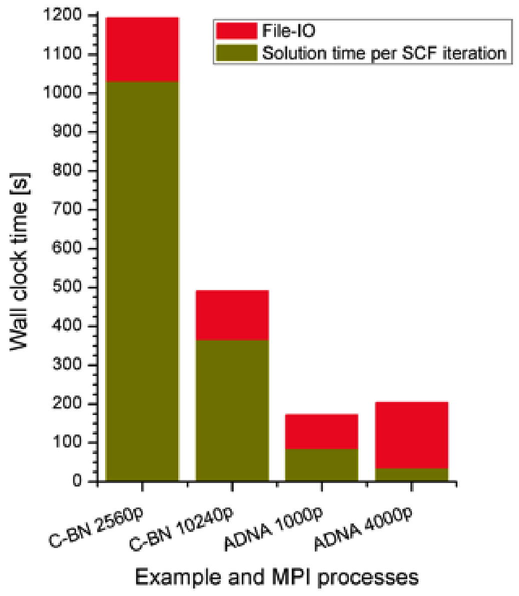

Figure 6: Times per SCF iteration for solving and saving several problems with different numbers of processors. The examples are C-BN with ~13000 atoms and DNA with ~18000 atoms.

2.8 Exciting/ELK

2.8.1 Overview

Exciting

Exciting is a full-potential all-electron density-functional-theory (DFT) package based on the linearised augmented plane-wave (LAPW) method. It can be applied to all kinds of materials, irrespective of the atomic species involved and also allows for the investigation of the atomic-core region. The code particularly focuses on excited state properties, within the framework of time-dependent DFT (TDDFT) as well as within many-body perturbation theory (MBPT). The code is freely available under the GNU General Public License.

Elk

Elk is an all-electron full-potential linearised augmented-plane wave (FP-LAPW) code with many advanced features. Written originally at Karl-Franzens-Universität Graz as a milestone of the EXCITING EU Research and Training Network, the code is designed to be as simple as possible so that new developments in the field of density functional theory (DFT) can be added quickly and reliably. The code focuses on ground state properties with some effort devoted to excited state properties. The code is freely available under the GNU General Public License.

Both Exciting and Elk codes are successors of the original EXCITING FP-LAPW code and have many common algorithms and functionality.

The analysis of the Exciting/Elk codes had revealed the inefficiency of the straightforward optimisations of the existing implementations without a fundamental change in the wave-functions representation. The proposed representation is based on the explicit knowledge of the radial basis functions for each azimuthal quantum number and is already implicitly used in some parts of the Exciting code. The change of the wave-function representation impacts all major (core) parts of both codes such as construction of the first- and second-variational wave functions, setup of the charge density and magnetisation and calculation of operator matrix elements in the basis of first- or second-variational states and requires a significant Fortran code rewrite.

In order to preserve the “canonical” Fortran implementations which are well known to the corresponding scientific communities and at the same time to avoid a redundant job of optimising two codes we have decided to isolate generic LAPW method algorithms in a standalone “LAPW engine” library, independent of a particular LAPW code implementation. The introduction of the low-level library has one more objective, namely, the splitting of the code development into a frontend “physics” part maintained by a particular LAPW code community and a common backend part maintained by advanced code developers.

The SIRIUS library (scalable interface library for augmented waves) was created under the WP8 of the PRACE-2IP project with the following ideas in mind:

- The library must be scalable and capable of handling large unit cells with hundreds of atoms and running on hundreds to thousands of nodes on modern hybrid/multicore systems.
- The library must have a clear MPI + OpenMP (+ GPU) parallel model. The coarse-grained parallelisation must happen on the multi-dimensional (depending on the task) grid of MPI ranks, while the fine-grained parallelism is achieved using the OMP threads (and/or a GPU device).
- The library must be clearly written and documented, must exploit accurate numerical techniques and extensively use the external high-quality libraries.

The following features are already implemented in SIRIUS:

- Ground state calculation (non-magnetic, magnetic collinear and non-collinear) and precise total energy.
- Full diagonalisation of the Hamiltonian (not available in Exciting/Elk)
- Spin-orbit and Hubbard-U corrections.
- Distributed Poisson solver and exchange-correlation potential generation (both are not available in Exciting/Elk)
- Forces and structural relaxation.
- Momentum matrix generation

The latest version of SIRIUS library is uploaded to the official SCM repository at hpcforge.org: (https://hpcforge.org/anonscm/git/exciting-elk/exciting-elk.git).
Elk+SIRIUS benchmark using the example of Eu_6C_{60} (120 atoms in the unit cell). The runs were done on a dual-socket AMD Interlagos 16-core platform with 2 MPI ranks per node, 16 threads per rank. The Hamiltonian matrix size is ~30’000.

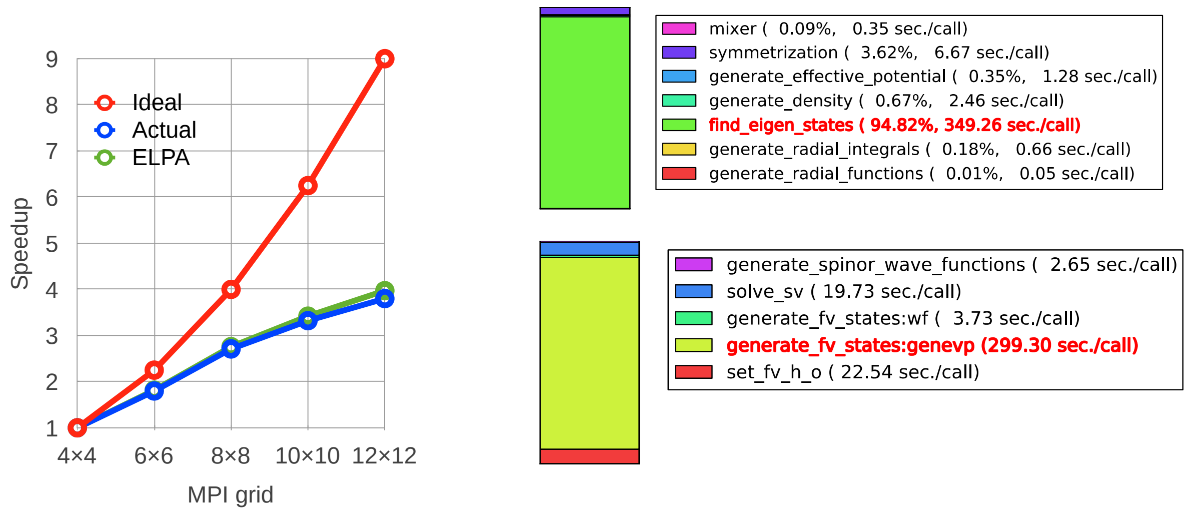

Figure 7: Left panel: hard scaling of the code with respect to the size of the MPI process grid. Red line – ideal scaling, blue line – actual scaling of the code. Green line – scaling of the ELPA eigenvalue solver used for the Hamiltonian diagonalisation. Clearly, the eigenvalue solver is the limiting factor for the good scaling. Right panel: (top) Time distribution between various parts of the code for the 144 (12x12) MPI ranks job. Generation of the Kohn-Sham eigenfunctions (green bar) takes ~95% of the run-time. (bottom) Time distribution inside the Kohn-Sham eigenfunctions generation: most of the time (yellow bar) is spent in the generalised eigenvalue problem (ELPA).

2.8.2 Workplan

We have the following plan for further refactoring of Elk/Exciting (listed in order of importance):

- Based on the experience with data distribution in SIRIUS, modify the Exciting Fortran code (but not very much so that the Exciting community can accept the modifications) in order to eliminate the obvious and well-known I/O bottlenecks and the excessive memory consumption.
- Prepare the public release of SIRIUS library and the modified version of the Exciting code.
- Start porting parts of SIRIUS code to GPU. There are several well-isolated time consuming parts besides the generalised eigenvalue problem. For example: construction of the Hamiltonian and overlap matrices (matrix-matrix multiplication) and the interstitial charge-density summation (FFT) are the best candidates for the GPU implementation.
- Insert GPU-enabled distributed eigenvalue solver when available.
- If time allows, start the work on the fast generation of the plane-wave matrix elements used in the excited states part of the Elk/Exciting codes.

2.9 PLQCD

2.9.1 Overview

In this work package we worked on two community codes, namely tmLQCD and Chroma, for lattice QCD calculations. The goal was to refactor certain key parts of these codes in order to improve their scaling on multi-core architectures. For tmLQCD we focused on the Dirac operator and the linear solver since these two key components are responsible for the largest percentage of computational resources during the course of lattice calculations. For the Dirac operator we developed a stand-alone library, PLQCD, for Wilson-like fermions where parallelisation is implemented using openMP and MPI. This hybrid parallelisation is expected to reduce the communication overhead as we increase the number of cores by using openMP for cores that share the same memory, while using MPI only for communications among cores that do not share the same memory. In addition to this hybrid parallelisation in PLQCD, we also implemented other improvements including overlap of communications and computations by dividing the lattice sites on each MPI process into bulk and boundary sites and re-ordering the computations done on these different sites in an order that allows the overlap of computations and communications as much as possible. We have also implemented the recently available AVX instructions for vectorisation. For the liner solver used in tmLQCD, we have implemented the deflated Conjugate Gradient (CG) solver, EigCG into tmLQCD. This solver has shown excellent performance for Twisted-Mass simulations, speeding up the calculations by a factor of about 3 on the largest lattices used by the ETM collaboration as has been demonstrated in a recent publication.

In Chroma, we focused on two important parts: Landau gauge fixing, and Hybrid Monte Carlo integrators. For the Landau Gauge fixing, we use the PFFT library, which is parallelisable in up to three dimensions. This is an advantage over the popular FFTW library, which is only parallelisable in one dimension. This leads to a much better scaling of the code. The other part was concerned with the optimisation of the integrator scheme used to integrate the molecular dynamics equations of motion during the course of Hybrid Monte Carlo Simulations. We offer a Poisson Bracket measurement routine for several lattice actions, which allows the evaluation of an optimum value for the free parameters in the integrator scheme. Tuning these parameters can increase the acceptance rate of the simulations leading to a speedup in the generation of gauge configurations.

It is worth noting that as part of the code development activities in this work package, we have organised a dedicated session for code development during the 31st International Symposium on Lattice Field Theory that was held at Johannes Gutenberg University Mainz, Germany from Monday 29th July to Saturday 3rd August 2013. This session was supported by PRACE and was the first time to organise such a session. During this session code developers from various groups in the lattice community shared their experience and exchanged ideas concerning code optimisation for new architectures. The session was well attended and will probably initiate a trend for organising such a session during future lattice conferences.

Working on the tmLQCD code, we identified areas where further refactoring will be beneficial specially taking into account the new Intel MIC accelerator cards. In particular we plan to implement the following:

- Refactoring of the PLQCD (tmLQCD) implementation of the Dirac operator for the Intel MIC cards. This requires a special attention for the vectorisation part of the Dirac operator in order to benefit from the wider registers (512 bits) available on the MIC.
- From a more general point of view, one would like to have a vectorised library for basic linear algebra operations in lattice QCD calculations that covers more situations. This means single or double precision data (spinors and gauge fields) stored on registers that could be 128-, 256-, or 512-bits. This will cover the Intel MIC cards as well. We plan to build such a library that we call LQCD-BLAS. In the case of tmLQCD for example, the vectorisation part is only implemented for using SSE2 and SSE3 instructions for double precision data. However, some parts of the code will benefit from mixed-precision calculations and require both single and double precision vectorisation components. This is particularly true for the linear solver where iterative

solvers based on what is called reliable updates uses mixed precision. In these solvers, the bulk of the iterations are done in single precision and occasionally the residual is updated in double precision.

- Finally, we will implement another solver in tmLQCD which is also based on deflation of eigenvectors based corresponding to small eigenvalues. This solver is a combination of GMRES with Deflated Restarting (GMRES-DR) and deflated BiCGStab or Deflated GMRES. This solver has some advantages over EigCG. First it assumes non-Hermitian matrix which is the case for the Lattice Dirac operator. In case of EigCG one has to recast the problem into a Hermitian Positive Definite system using the normal equations. This procedure leads to an increase in the condition number of the matrix involved (Hermitian systems are however easier and more efficiently solved with CG solver than the non-Hermitian systems). The second advantage of GMRES-DR is the fact that the eigenvectors needed for deflation can be computed from the solution of a single linear system while in the EigCG case they are accumulated incrementally during the solution of the first few systems. Having the eigenvectors from the solution of a single system means that the benefit of deflation will be readily available starting from the second system.

2.9.2 Workplan

Currently, we tested PLQCD on an Intel MIC without the vectorisation part. We have been able to compile and make some performance tests. However, to benefit from the vectorisation capabilities of the MIC we need to re-write the vectorisation part for the elementary SU(3) algebra parts. This goes also under our plan to write a more general vectorisation library for different possible size registers. This is expected to be finished by February2014. We will then reintegrate this into tmLQCD and PLQCD. For the GMRES-DR linear solver, we started working on this inside tmLQCD. An initial version of GMRES-DR is implemented in tmLQCD, it is however un-tested. We started completing the interface for this solver in tmLQCD and made some testing. It was did not work as expected due to a possible bug in the code. We are fixing this now. This part is only the first part of the solver. One still has to implement deflation using the eigenvectors computed with GMRES-DR. One option is to deflate BiCGStab, which we plan to do. This work is expected to be finished by the end of December.

2.10 ELMER

2.10.1 Overview

Elmer is an open source multiphysical simulation software developed by CSC - IT Center for Science in Helsinki, Finland. Elmer includes physical models of fluid dynamics, structural mechanics, electromagnetics, heat transfer, acoustics, etc. These are described by PDEs, which Elmer solves by the FEM. Currently Elmer has more than 5000 worldwide users. Elmer has shown excellent scaling on appropriate problems up to thousands of cores. Elmer's developers focused on implementation of more a robust solver, which enables further scaling of Elmer. The standalone tool ElmerSolver has implemented several types of solvers: time integration schemes for the first and second order equations, solution methods for eigenvalue problems, direct linear system solvers (LAPACK & UMFPACK), iterative Krylov subspace solvers for linear systems (GMRES, CG), multigrid solvers (GMG and AMG) for some basic equations, ILU preconditioning of linear systems, the discontinuous Galerkin method. Recently Elmer code was extended by new FETI1 and TFETI domain decomposition.

FLLOP (FETI Light Layer On top of PETSc, http://spomech.vsb.cz/feti/) is a novel software package developed at IT4Innovations, VSB-Technical University of Ostrava, Czech Republic, for solution of constrained quadratic programming problems (QP). It is an extension of PETSc framework. PETSc is a suite of data structures and routines for the parallel solution of scientific applications modelled by PDE. FLLOP is carefully designed to be user-friendly while remaining efficient and targeted to HPC.

Most problems described by PDEs and solved by the FEM result in large systems of linear equations. The main objective of the previous subtask was the scalability analysis and bottleneck identification of the Elmer and FLLOP solvers. By means of systematic testing in the phase "Code performance analysis", the FETI type solvers were identified as the most efficient tool for their solution. Because of the effort to improve the scalability of the Elmer FETI solvers and the long-term experience of VSB-TUO researchers with FETI methods developed and implemented in their FLLOP library, the synergy was achieved - the **Elmer-FLLOP interface** was implemented in the phase "Code refactoring".

The Elmer user who is going to apply more efficient FETI solvers has to convert Elmer matrices and vectors into standard arrays, include "fllopaif.h", and then FLLOP solvers can be called immediately from Elmer. The final contribution is not only the **scalability improvement** of Elmer, but also functionality extension of Elmer via FLLOP enabling an efficient parallel solution of contact problems and other equality, inequality and box constrained QPs. The efficiency of the refactored code was first tested by teams of developers on the model cube linear elastic benchmark in the "prototype experimentation" phase and then on several engineering problems in the "code validation" phase. The possibility to solve larger, more complicated problems faster will be certainly fully appreciated by Elmer users in the near future.

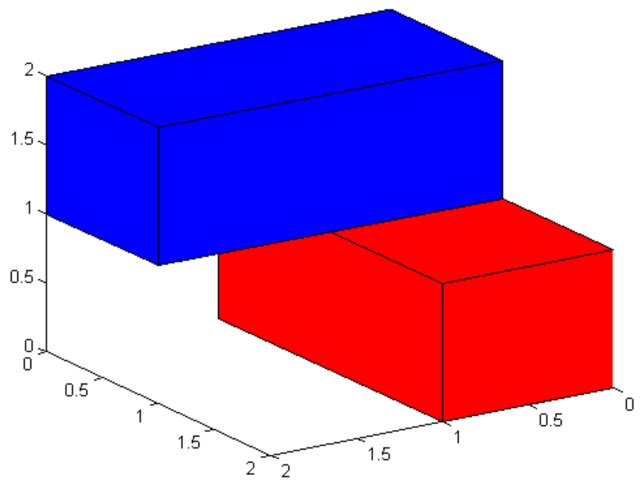

Figure 8: Angular domain benchmark

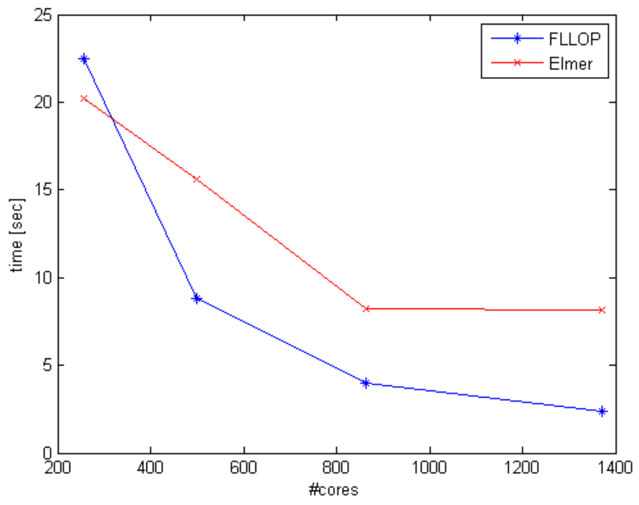

Figure 9: Graph of the strong scalability (Elmer vs. FLLOP)

The main target for an extension is the scalability testing and comparison of optimised implementations of linear solvers (direct, iterative) running on CPUs only, CPUs+GPUs and CPUs+MICs on benchmarks with systems of linear equations with both, symmetric and non-symmetric matrices of various structures and fill-ins, including those arising from FETI-1 and TFETI applications.

dofs	dual	nsubd 3D	iter	FLLOP	nred	nproc/nred	Elmer
13 498 368	1 257 168	256	54	22.47	1	256.0	20.21
13 891 500	1 650 300	500	48	8.79	5	100.0	15.61
14 292 192	2 050 992	864	43	4.00	24	36.0	8.2
14 700 516	2 459 316	1 372	40	2.39	28	49.0	8.17

Table 2: Strong scalability Elmer vs. FLLOP for the angular domain benchmark

2.10.2 Workplan

We are going to pay an extra attention to the implementation, testing and comparison of linear solvers running on

- CPUs only,
- accelerated CPUs by means of GPUs,
- accelerated CPUs by means of MICs.

We plan to test:

- direct solvers – e.g. LU in Magma, SuperLU, MUMPS, etc.
- iterative solvers – e.g. CG, DCG, BiCG in PETSc etc.,

for both matrix formats:

- sparse,
- dense.

As the benchmark we plan to use so called coarse problem appearing in FETI methods in projector application, or in DCG method in search direction update. The results should help us further to improve the scalability of Elmer library.

2.11 ALYA and Code Saturne

2.11.1 Overview

There is a common feeling in the community that in most cases not a single simulation system can provide all necessary features, but coupling the best codes of each discipline will enable more flexibility and simulation quality to the end user.

The main target of the extension is to build an extremely scalable tool to study Fluid-Structure Interaction (FSI), by coupling Code_Saturne for the fluids and Alya for the solids.

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 the two CFD codes selected for PRACE 2iP/3iP benchmark suite. The code can be downloaded from http://code-saturne.org/cms and a general documentation is available at http://en.wikipedia.org/wiki/Code_Saturne where other links are also available.

The Alya System is a Computational Mechanics (CM) code developed at Barcelona Supercomputing Center (BSC) that has two main features. Firstly, it is specially designed to run with the highest efficiency in large-scale supercomputing facilities. Secondly, it is capable of solving different physics problems, each one with its own modelling characteristics, in a coupled way. These two main features are intimately related, which means that any complex coupled problems solved by Alya will still be solved efficiently. The parallelisation is based on a hybrid MPI/OpenMP strategy. Alya is written in Fortran90 and it is divided in services, kernel and modules. Each module deals with a physical problem. In particular, the Alya-Solidz module solves computational solid mechanics problems with large deformations. It has been proved in PRACE-3IP WP9.3 [add a reference to the deliverable] that the code scales optimally up to 8000 processors of Intel Sandy Bridge.

The main achievement of WP8 resides in the improvement of the scalability of Code_Saturne, which has been shown during a test conducted at Argonne Laboratory for a Large-Eddy Simulation using a 105B cell mesh on 524,288 processing cores (IBM Blue Gene/Q). Regarding the Alya code, the target is to test the scalability up to a large number of processors and port the code to other architectures. Although Alya actually solves FSI problems, one of the goals of this project is to provide an extensible platform that allows the Solidz module to be coupled with other CFD codes that usually lack of a robust and scalable solid mechanics code.

2.11.2 Workplan

- Define a simple FSI test case for testing.
- Identify which quantities and variables have to be exchanged between both codes.
- Define a coupling environment (library), which is able to exchange the data between both codes and solvers for partitioned coupling and allow coupling exchange functions to be called for both codes.
- On Code_Saturne's side, investigate the coupling between Code_Saturne and SYRTHES, which is also a Finite Element code, before using the Parallel Location and Exchange (PLE) library (GPL licensing) at the interface between Code_Saturne's and Alya's domains.
- Define a larger test case and run a full simulation.

2.12 PFARM

2.12.1 Overview

PFARM is part of a suite of programs based on the ‘R-matrix’ ab-initio approach to the variational solution of the many-electron Schrödinger equation for electron-atom and electron-ion scattering [11]. The package has been used to calculate electron collision data for astrophysical applications (such as: the interstellar medium, planetary atmospheres) with, for example, various ions of Fe and Ni and neutral O, plus other applications such as plasma modelling and fusion reactor impurities. The code has recently been adapted to form a compatible interface with the UKRmol suite of codes for electron (positron) molecule collisions [12], thus enabling large-scale parallel outer-region calculations for molecular systems as well as atomic systems. Each of the main stages of the calculation is designed to take advantage of highly optimised, numerical library routines. Hybrid MPI / OpenMP parallelisation has also been introduced into the code via shared memory enabled numerical library kernels [13].

The main goal of the work in WP8 was to enable very large-scale electron-atom and electron-molecule collisions calculations by exploiting the latest high-end computing architectures.

Work previously accomplished:

1. *The latest parallel symmetric eigensolver routines have been introduced* into the PFARM (EXDIG). These new solvers have recently been made available as part of the ELPA (Eigenvalue SoLvers for Petaflop-Applications [14]) project through a collaboration of several German centres and IBM. In order to improve further the parallel performance of EXDIG, ELPA has also been used to calculate each Hamiltonian sector parallel eigensolve concurrently on MPI sub-groups of tasks. Threefold performance improvements at higher core counts have been achieved with this approach.
2. *Parallelisation of the input routines* for surface amplitude data to the multiple process pipelines in EXAS. This approach significantly reduced the set-up time for the propagation stage. Parallel output has also been introduced to EXDIG as an integral feature of the MPI sub-group eigensolver approach.
3. *New scripts have been developed for automatically 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.
4. *Computational accelerator technologies* – e.g. NVIDIA GPU and the Intel Xeon Phi have been investigated for PFARM. Optimisation strategies such as auto-offloading and native execution using new routines from CuBLAS [15], Intel MKL [16] and MAGMA [17] have been analysed. A near complete port of the EXAS code to GPUs has also been achieved.
5. *Petascaling through MPI refactoring:* As part of work within PRACE-2IP WP8, ICHEC has implemented a new MPI communication layer within the EXAS code of the PRMAT/PFARM suite, which has been shown to significantly improve the scalability of the code

Building upon the initial findings from the main WP8 project, PFARM developers will focus on optimising the code for new and emerging architectures during the extension period. The PFARM code is developed in a scalable, modular style and was designed from the start to exploit a range of highly optimised numerical library routines for the computational cores of the overall calculations. These features mean that the code has the potential to fully exploit the computational power of new many-core architectures and computational accelerator hardware. As part of the work under the extension we will also adapt the code for new molecular inputs whilst introducing new inter-program scripts that aid usability and ensure good parallel performance, thereby making the code available to a new user base.

2.12.2 Workplan

1. *Initial port of PFARM and associated third-party software packages to the Intel Xeon Phi architecture:* Firstly port to a single Intel Xeon Phi and then to an Intel Xeon Phi Cluster. Continuing work undertaken in the previous WP8 project (details above), the overall performance implications of in-situ offloading of computational bottlenecks such as Level 1 BLAS and Level 2 BLAS and LAPACK (e.g. eigensolvers) routines will be investigated**.**
2. *Final development and integration of both user-friendly and efficient wrapper sections for input of data for atomic and molecular calculations.* This will be undertaken in partnership with the UK-RAMP project (UK EPSRC grants EP/G055416/1,EP/G055556/1, EP/G055475/1, EP/G055599/1), which is developing the UKRmol packages.
3. *Auto-configuration and auto-load-balancing via Perl scripts for a wider range of problem sizes including new architectures (e.g. Intel Xeon Phi clusters).* Development work will focus on both standard test cases and new, large test-cases for demonstration e.g. CH4 cases from partners at UCL [18] and Fe+ cases from partners at QUB [19].
4. *GPU enablement on Kepler K20s:* ICHEC plans to continue enabling the EXAS code of the PRMAT/PFARM suite on GPUs. Within the two PMs of effort for PFARM development, ICHEC will enable the EXAS code on the NVIDIA K20 architecture and will also test new features offered by the K20 including Dynamic Parallelism. ICHEC will subsequently benchmark the code on a NVIDIA K20 cluster.
5. *Dissemination***:** In collaboration with the STFC, ICHEC plans to disseminate the results of its PFARM enablement activity through whitepapers, deliverables and peer-reviewed journal publications.

2.13 RAMSES

2.13.1 Overview

The RAMSES code was developed to study the evolution of the large-scale structure of the universe and the process of galaxy formation. 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.

During the first two years of WP8, the focus was on two main objectives: the re-implementation of the code with a hybrid OpenMP+MPI approach and the implementation of the GPU version of the hydro kernel, based on the OpenACC standard.

The main achievements are:

- Full shared memory OpenMP implementation of the hydro kernel, with good scalability up to 32 cores per CPU;
- Full shared memory OpenMP implementation of the gravity solver. Due to the intrinsic characteristics of the problem (long range forces), leading to an intense memory usage, scalability is limited. However, it represents an effective solution for memory bound problems;

- GPU refactoring of the hydro kernel, based on OpenACC. Two implementations were developed following different approaches. The first led to limited performance improvement, the second is in an advanced stage of development.

2.13.2 Workplan

For the WP8 extension the main objective is the completion of the GPU porting of the hydro kernel, following the second adopted approach, whose main characteristics can be summarized as follows.

In order to improve the efficiency of memory access, main data structures, hydrodynamic variables (density, pressure and velocity) and gravitational forces had to be reorganized.

This solution is sketched in Figure 10.

In this case, the algorithm consists in the following steps:

- compose large, regular, rectangular patches of contiguous cells at the same refinement level on the CPU;
- off-load the patch and integrate it on the GPU;
- at the same time, using asynchronous operations, composes the next patch on the CPU and start copying it to the GPU;
- once the patch is updated, move it back to the CPU and update the original data structure.

With this solution, a much more effective usage of the GPU memory is guaranteed. At the same time, the off-load overhead is hidden thanks to the usage of asynchronous operations and the overlap with the work performed by the GPU cores. Furthermore, the large-patches re-composition overhead should finally be hidden. This is important since the patch-re-composition stage is not present in the original algorithm, so it represents an overhead due to the usage of the GPU.

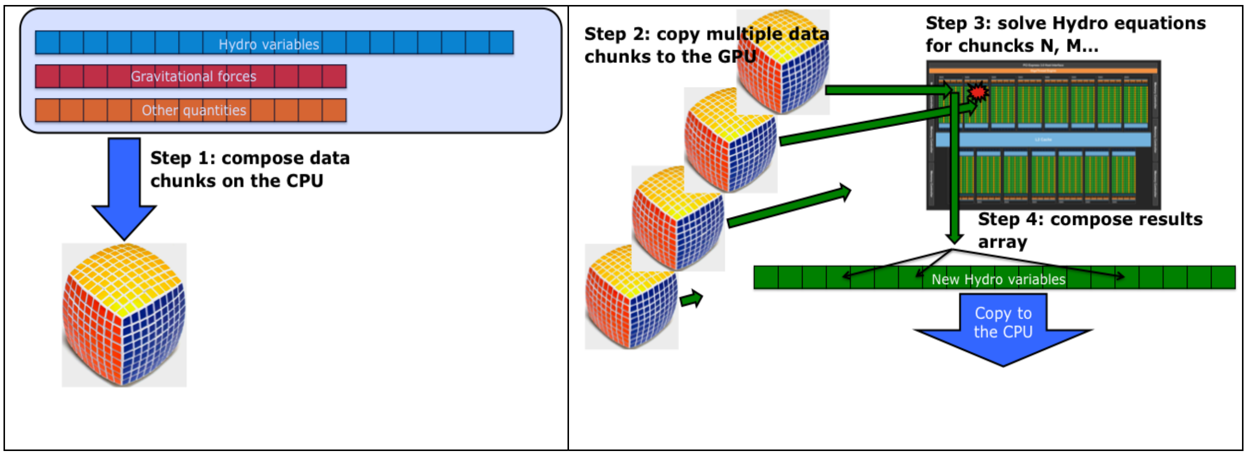
Hydro variables
Gravitational forces
Other quantities
Step 1: compose data chunks on the CPU
Step 2: copy multiple data chunks to the GPU
Step 3: solve Hydro equations for chuncks N, M...
Step 4: compose results array
New Hydro variables
Copy to the CPU

Figure 10: GPU implementation: approach 2

During the last months of WP8, a preliminary version of the algorithm has been developed, acting on a simplified uniform mesh. The work for the extension will be focused on the full implementation of the AMR approach, through the following steps:

1. Tuning and optimization of the uniform mesh version on the GPU;
2. AMR enabling adding restriction and prolongation functions and properly accounting for boundary conditions at cells with different resolution;
3. Debugging of the AMR implementation;
4. Tuning and optimization of the AMR implementation;
5. Merge of the new GPU version with the trunk SVN distribution.

If residual effort is available we will also start investigating possible solutions for porting also the gravitational solvers to the GPU.

3 Summary

The WP8 one year extension will focus on a subset of the codes developed in the first two years of activity, further enhancing the accomplished refactoring work. The codes were selected according to the achieved results, the proposed work plan and the available effort. The main objectives proposed for each application are summarized in the following table:

Code name	**Main objectives**
EAF-PAMR	Tuning and optimization of the OpenCL code on a number of different platforms. Implementation of further numerical methods for the OpenCL code
OASIS	Merge OASIS3-MCT2.0 changes to NEMO
I/O Services	XIOS interface for IFS will be completed XIOS memcache code will be optimised
ICON	Complete introduction of the modifications developed in WP8 into the main ICON development trunk Testing and validation
Fluidity/ICOM	Implementation of high performance I/O modules Integration of PETSc's DMPlex module
QuantumESPRESSO	Xeon Phi enabling of specific kernels Increase scalability of the PHonon kernel on large number of processors
SIESTA	Implementation of an efficient checkpointing procedure
EXCITING/ELK	Porting of the SIRIUS library to the GPU
PLQCD	Enabling to Xeon PHI, with specific care to vectorization
ELMER	Testing and benchmarking the code on various architectures in order to improve its scalability
ALYA/CODE_SATURNE	Coupling of Code_Saturne for fluids to Alya for solids.
PFARM	Enabling to Xeon PHI and Kepler GPU
RAMSES	Completion of GPU enabling through OpenACC standard

The working methodology already in use will be maintained, strongly relying on the close collaboration between HPC experts and scientists. This, not only drives the software development process according to the needs and expectations of research, but also facilitates the validation of the codes which becomes a day-to-day process, with no need of a specific procedure. The developed algorithms in most of the cases are integrated in the official distributions and made immediately available to the community software developers, for further testing and debugging, and to the users, finally, in order to let them experiment the new versions and releases. This strongly increases the impact of WP8's work and makes the developed software promptly available to the users' community.