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

D9.1.2
Support for Industrial Application Year 2

Final

Version: 1.1
Author(s): Paul J Graham, EPCC
Date: 22.6.2013

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: < **D9.1.2**>	
	Deliverable Nature: <DOC_TYPE: Report / Other>	
	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: Support for Industrial Application Year 2**	
	ID: D9.1.2	
	Version: <1.1 >	**Status: Final**
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D9.1.2.docx	
Authorship	**Written by:**	Paul J Graham, EPCC
	Contributors:	John Donners, SurfSARA; Paride Dagna, CINECA; Joerg Hertzer, HLRS; Georgi Prangov, NCSA, Peicho Petkov, NCSA
	Reviewed by:	Daniel Ahlin, KTH; ThomasEickermann, JUELICH
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	17/07/2013	Draft	PJG: first draft to partners
1.0	08/08/2013	Review version	PJG: release for internal review
1.1	22/08/2013	Final version	PJG: updated following review comments

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure, Industry, OpenFOAM, Delft3D, CP2K, GROMACS, Elmer

Table of Contents

List of Figures

List of Tables

References and Applicable Documents

[1] http://www.prace-project.eu
[2] CINECA, Fermi User Guide. http://www.hpc.cineca.it/content/ibm-fermi-user-guide
[3] HLRS, Hermit User Guide. https://wickie.hlrs.de/platforms/index.php/Cray_XE6
[4] Massimiliano Culpo, Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters, PRACE-1IP Whitepaper http://www.prace-project.eu
[5] OpenFOAM white paper, TBA
[6] Elmer white paper, TBA
[7] Delft3D white paper, TBA
[8] http://oss.deltares.nl/web/opendelft3d/home
[9] http://www.deltaressystems.com/
[10] http://www.scalasca.org/
[11] D9.1.1 Support for Industrial Applications Year 1 http://www.prace-project.eu/IMG/pdf/D9-1-1_2ip.pdf
[12] Biovet, http://www.biovet.com/
[13] Log P, http://en.wikipedia.org/wiki/Partition_coefficient
[14] GROMACS, http://www.gromacs.org/
[15] CP2K, http://www.cp2k.org/
[16] METIS, http://glaros.dtc.umn.edu/gkhome/views/metis
[17] CHARMM, http://www.charmm.org/
[18] “Calculation of Intramolecular Force Fields from Second-Derivative Tensors”, Jorge M. Seminario, International Journal of Quantum Chemistry Volume 60, Issue 7, pages 1271–1277, 1996
[19] http://www.csc.fi/elmer

List of Acronyms and Abbreviations

CFD	Computational Fluid Dynamics
CHARMM	Chemistry at HARvard Macromolecular Mechanics
CINECA	Consorzio Interuniversitario, the largest Italian computing centre (Italy)
CP2K	derived from the CPMD code and the original meaning was the CP (Car-Parrinello = ab initio MD) code for the new Millenium
CPU	Central Processing Unit
CSC	Finnish IT Centre for Science (Finland)
DP	Double Precision, usually 64-bit floating point numbers
DIC	diagonal incomplete-Cholesky
DILU	Block-diagonal, incomplete lower and upper triangular matrices
EC	European Commission
EPCC	Edinburg Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)
GROMACS	GROningen MAchine for Chemical Simulations
HLRS	High Performance Computing Centre, Stuttgart
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
I/O	Input/Output
MKL	Math Kernel Library (Intel)
MPI	Message Passing Interface
NCSA-BG	National Centre for Supercomputing Applications (Bulgaria)
NUMA	Non-Uniform Memory Access or Architecture
Open MP	Open Multi-Processing
OpenFOAM	Open source Field Operation And Manipulation
PCG	Preconditioned Conjugate Gradient
PRACE	Partnership for Advanced Computing in Europe
SP	Single Precision, usually 32-bit floating point numbers
SURFsara	Dutch national High Performance Computing & e-Science Support Center
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
UHEM	Turkish National Centre for High Performance Computing
VSB	Technical University of Ostrava, Czech Republic

Executive Summary

The PRACE-2IP-WP9 work has been concerned with the provision of support for industrial partner applications in order to better exploit HPC resources. This report summarises the activities in the second year of task 9.1, which is the work on enabling several applications to more efficiently use Tier-0 and Tier-1 systems, and to generally enhance the parallel performance of these industrially relevant codes.

Initially, three codes were identified for enhancement: OpenFOAM, a CFD software package; Elmer, a multi-physics finite element code; and Delft3D, a modelling suite for investigating hydrodynamics, sediment transport and water quality for fluvial, estuarine and coastal environments. Since the previous report, an additional investigation has been added to the process: a framework has been developed which utilises both the GROMACS and the CP2K open source packages for molecular dynamics simulation, in this case in the field of pharmaceutical research.

The principal results in this second year are:

- OpenFOAM – has had an OpenMP hybridisation implemented in several further modules. Several real industrial test cases have been collected and benchmarked. OpenFOAM is now available on several more PRACE platforms.
- Elmer – parallel performance has been further enhanced via hybridisation and an innovative approach to rotating boundary conditions.
- Delft3D – has had its portability significantly improved and been demonstrated to work on several new platforms. Has been benchmarked in detail and bottlenecks are now being tackled.
- GROMACS/CP2K – a framework has been developed to facilitate in the identification of new drug candidates via HPC.

1 Introduction

The PRACE-2IP-WP9 workpackage is concerned with providing support for applications from industrial partners in order to better exploit HPC resources. Task 9.l of the workpackage had a particular focus in identifying Open Source codes which are both of relevance to industry and potentially amenable to making efficient use of Tier-0 and Tier-1 systems. In addition, the task had been instructed to actually analyse and, where possible, enhance the parallel performance of a selection of these codes.

The previous report from this task, D9.1.1 [11], discussed the results of two surveys undertaken which addressed both industry end users and independent software vendors. It also outlined the criteria for selection of codes suitable for consideration in the HPC-*enabling* process. Through the survey and the criteria, three codes were deemed suitable for enabling: OpenFOAM, Elmer, and Delft3D. The D9.1.1 report also described the initial work achieved thus far in the enabling of the chosen codes.

This document is a follow on from D9.1.1, in that it describes the continuation of the enabling work on the chosen codes, and the results obtained, in the second year of this task. Section 2 looks at OpenFOAM, whilst sections 0 and 4 look at Elmer and Delft3D respectively. In addition to the three original codes chosen, a further enabling project, on a framework using GROMACS [14] and CP2K [15] in the context of the pharmaceutical industry, was started. This is discussed in section 5. Finally, section 6 summarises all the work undertaken in years 1 and 2 of task 9.1.

Note that for OpenFOAM, Elmer and Delft3D, the technical details of the enabling work have been reported in three white papers (see [5], [6] and [7] respectively), so this document will focus on the progress and achievements of the work on those codes. The reader is invited to access the white papers if more in depth technical information is required.

The PRACE partners involved in the technical aspects of the enabling work were: SURFsara, NCSA-BG, CINECA, HLRS, UHEM, VSB and CSC. EPCC and CINECA coordinated and administrated the work in this task.

2 OpenFOAM

2.1 Introduction

The OpenFOAM® (Open source Field Operation and Manipulation) CFD Toolbox is a free, open source CFD software package produced by OpenCFD Ltd. It has a large user base across most areas of engineering and science, from both industrial and academic organisations. OpenFOAM has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics.

By being open source, OpenFOAM offers users complete freedom to customise and extend its existing functionality. From a general point of view, it can be thought as a framework where Computational Fluid Dynamics (CFD) programmers can build their own code, as it provides them with the abstraction sufficient to think of a problem in terms of the underlying mathematical model.

OpenFOAM is a MPI-parallelised application whose performance, scalability and parallel efficiency have been tested over many different system architectures, including massively parallel clusters such as Tier-0 systems.

To increase scalability, the parallelisation strategy employs the "Zero-Halo Layer Decomposition" approach whereby a given volume of cells is assigned to each MPI process and communication is only with neighbouring cells. According to various benchmarks on some of the most commonly used OpenFOAM solvers, the MPI communications consistently dominate the simulation time going over some hundreds of cores.

CINECA have led the OpenFOAM enabling work, with input and assistance from HLRS, UHEM and VSB. All related software installations and test runs on the systems of HLRS were done by personnel located there.

2.2 Workplan

The enabling work in PRACE-2IP-WP9 has focussed on the continued investigation of a hybrid multi-threaded solution (introduced in PRACE-1IP WP7, [4]), with the intention of reducing MPI communications and increasing scalability. This solution has been extended to four OpenFOAM solvers, and its behaviour on a BG/Q [2] and Cray [3] system investigated.

Key to this investigation has been engagement with industry users in order to obtain real test cases of interest to that community. As such, where possible these test cases have been shared amongst PRACE partners to allow comparative studies to be made.

It is well known that the scaling of programs parallelized with MPI (Message Passing Interface) might be limited by the overhead of the communication between the MPI tasks and by the reduced amount of operations left to each of these tasks. Hybrid parallelisation, where MPI and shared memory parallelisation, for example via OpenMP directives, work together, allows the use of a group of computing cores by several shared memory threads within one MPI task. In this way, for a given number of cores the total number of MPI tasks is reduced by the factor given by the number of shared memory threads within each MPI task. This should enable a reduction in the overhead of the MPI communication. It also enables the use of fine grained shared memory parallelisation, in addition to the coarse grained MPI parallelisation, which may be more efficient for certain types and quantities of operations.

In this way, the same hardware resources can potentially be used more efficiently, provided that a highly efficient shared memory parallelization can be realised.

2.3 Implementation

Following the guidelines developed in the previous work [4], hybridisation using OpenMP directives has been implemented on OpenFOAM version 2.1.1 in the linear algebra libraries, the PCG solver, and DIC, diagonal and DILU pre-conditioners.

The main computing phases executed by the PCG module for the solution of the linear system can be categorised in five main phases which are repeated until convergence criteria are reached:

- preconditioning
- scalar product and reduction of the partial sums (requires MPI communication)
- cells update
- matrix–vector multiplication and reduction of the partial results (requires MPI communication)
- solution and residual update (requires MPI communication)

The only hybridisable parts, where OpenMP directives were introduced, are preconditioning, matrix–vector multiplication and cells update. Only the diagonal pre-conditioner was fully

hybridised, while the others were only partially hybridised because of some internal sections with data dependency issues. Also the two cycles of cells update were fully hybridised.

2.4 Results

Four main test cases were used to examine the performance of the hybrid OpenMP/MPI OpenFOAM code (more detail can be found in the white paper [5]):

- Lid-driven cavity flow: exercises the *icoFoam* solver
- NACA airfoil: *simpleFoam*
- DTMB Hull: *interFoam*
- Cold flow: *coldengineFoam*

Rather disappointingly, the hybrid version of OpenFOAM was not faster than the pure MPI version in all but one case for the various configurations investigated. This is discussed in more detail in the white paper, but is explained principally by:

- Greater time spent in linear algebra operations (matrix vector multiplication) by the hybrid version due to reordering of the operations and instructions necessary to remove data dependencies.
- The time spent in the non-hybridisable parts, where MPI communications are present, falls as the number of cores increases. This behaviour is essentially related to the decreasing of the subdomains of the initial mesh and the Zero-Halo Layer approach. In the hybrid version, once the number of MPI processes is fixed, the dimension of subdomains, and as a consequence the time spent in non-hybridisable parts, is fixed too, reducing the possible benefits derived from using OpenMP threads.
- For high core numbers, the time per iteration becomes very small, above all for preconditioning and interface update, making the overhead introduced by the activation of the OpenMP parallel region not negligible.

2.5 Summary

A further hybridisation of OpenFOAM, following previous work in PRACE-1IP WP7, has been implemented using OpenMP directives in the linear algebra libraries, the PCG solver, and DIC, diagonal and DILU pre-conditioners. Several industrial test cases have been gathered to benchmark and profile the OpenFOAM performance. Unfortunately performance of the hybrid version has not exceeded that of the pure MPI version. However, the hybrid version does allow additional benchmarking on future architectures/system which may be more amenable to exploiting a hybrid strategy.

It should also be noted that there were various challenges in getting stable installations of OpenFOAM working on the various systems used. This positive outcome of the enabling work is that now OpenFOAM is accessible on these systems to researchers and industrial users.

3 Elmer

3.1 Introduction

Elmer is one of the most popular multi-physical simulation software packages published under open source [19]. Elmer has been mainly developed by CSC: development was started in 1995 in collaboration with Finnish Universities, research institutes and industry. After its open source publication in 2005, the use and development of Elmer have become largely international. Elmer consists of several components: *ElmerGUI* is the graphical user interface; *ElmerPost* is the post-processing tool; *ElmerGrid* the mesh manipulation and partitioning tool; and most importantly *ElmerSolver* is the finite element solver.

ElmerSolver includes generic finite element library functionalities that may be called by dynamically linked modules describing the particular physical phenomena. The modules include models for fluid dynamics, structural mechanics, electromagnetism, heat transfer and acoustics etc. The physical models may be weakly coupled without any *a priori* defined way. Due to the modular structure new equations can be added to Elmer without touching the main library. The ElmerSolver library includes its own standard preconditioned iterative and multilevel methods but also an extensive list of available linear algebra libraries.

Elmer has been initially written as a parallel code and has demonstrated reasonable scalability for more than a decade. However, as the number of computational cores and the size of the problems grow, the old methods are not sufficient as new bottlenecks start to appear. Also, there may be new application areas including their specific features that must be implemented to enable the use of the code.

The enabling work within PRACE concerns the developments in two themes aiming to improve the applicability of the code for industrial applications. These are the implementation of the rotating boundary conditions based on mortar finite elements, and the hybridisation of the code to make use of the multicore processors that may in the future provide the most cost-efficient way of obtaining good performance.

In addition, there has been work performed on improving the FETI (Finite Element Tearing and Interconnect) implementation within Elmer, the results of this work were detailed in the previous deliverable [11] so will not be examined here.

The enabling work on this code has been undertaken by PRACE partners CSC and VSB.

3.2 Workplan

As mentioned earlier, the plan was to look at two themes to enhance the HPC performance of Elmer. For the hybridisation, this involved using OpenMP multi-threading to be applied to the ElmerSolver module. For the rotating boundary conditions, the plan was to implement a *moving mesh* approach to the problem based on the mortar finite element method, which is explained in detail in the white paper [6].

3.3 Implementation

Hybridisation

The ElmerSolver module was not originally implemented to make use of multi-threading, and being a legacy FORTRAN 90 and C/C++ code, static and module variables are commonly used for storing data. To handle this, OpenMP *THREADPRIVATE* directives were used to make some of the global data private to each thread, along with judicious use of *ATOMIC*

directives to enable updates to the global matrices to be performed simultaneously by the threads. In addition, the *matc* library used by ElmerSolver was refactored to be thread-safe.

In terms of performance enhancement, for the linear solvers, an important part of the computations are the sparse matrix-vector multiplications, commonly done within Krylov subspace methods. This operation was parallelised using OpenMP *PARALLEL DO* directives. In addition, an interface was added to utilise the highly optimised sparse vector-matrix multiplication subroutines provided by the *Intel MKL* library.

Rotating Boundary Conditions

In order to improve the opportunity for scalability, a special hybrid partitioning scheme was applied to the problem. In the scheme, elements involved in rotation are partitioned independently of the remaining elements which can be partitioned using a standard algorithm (such as METIS [16]). In 2D rotational problems, the mortar elements must all lie within the same partition, but in 3D, partitioning is allowed along the direction of the rotational axis, which enables better load balancing, and consequently improved parallel performance.

3.4 Results

Hybridisation

For the multithreaded performance investigation of ElmerSolver, a simple model was used looking at the solution of the heat equation in a steady state. The solver used was Conjugate Gradient (CG) without pre-conditioning. The results for the finite assembly and the linear system solution are displayed in Table 1 below, run on a dual socket Intel Xeon E5-2670 based machine.

Table 1: Scalability of hybridised ElmerSolver for a steady state 3D heat equation

	FE Assembly		**CG solve**	
threads	**CPU time (s)**	**Speedup**	**CPU time (s)**	**Speedup**
1	5.5	1	3.40	1
2	2.87	1.9	2.00	1.7
4	1.56	3.5	1.52	2.2
8	1.24	4.4	1.41	2.4
16	0.84	6.5	0.84	4.04

The performance improvements are quite modest, especially in the case of the CG solve. To investigate the causes behind this, two further test models were investigated, another steady state heat equation (*heat*) and a linear elasticity problem (*elas*), for three different mesh sizes (n= 35721, 105300, and 291060), both of which are relatively easy to solve with the conjugate gradient method even without pre-conditioning. Given the CG solver is readily parallelisable, the poor performance must be explained by threading overhead and non-uniform memory access (NUMA) effects. To take this into account, the conjugate gradient algorithm was refactored to reduce false sharing and minimise threading overhead, as well as redistributing the matrix data in an alternative manner to that traditionally used in Elmer, with the Intel MKL providing the local and global matrix-vector operations. This CG implementation was then tested against a textbook CG implementation (taken from the *HutIter* library of Elmer). The timings for this can be seen below in Table 2. This NUMA-aware CG implementation clearly outperforms the original, especially when the matrix is small enough to fit local caches.

Table 2: Comparison of original and NUMA-aware CG methods

Test problem	Speedup			
	16 threads		32 threads	
heat	Original	NUMA	Original	NUMA
small	5.27	7.75	2.95	11.59
medium	4.17	4.43	3.50	16.48
large	3.84	4.00	3.55	7.48
elas	Original	NUMA	Original	NUMA
small	4.50	5.81	3.43	16.44
medium	3.43	3.44	3.22	6.90
large	3.50	3.39	3.39	6.41

Rotating Boundary Conditions

The 2D mortar projector test case was based on a realistic induction machine, a relatively small model where the total time consumption comes mainly from the repeated resolution of time-discretised equations thousands of times. This showed reasonable scalability to 8 processors (6.6 times speedup), but beyond this returns were quickly diminished, for example 7.7 times speedup on 16 processors.

For the 3D test case, a more physically simplistic model based on a transient heat equation was used. The mesh for this consisted of 879240 elements, of which 29880 were associated with the mortar projector (a ratio of 1/34). First, the total number of partitions was fixed at 64, and the number of partitions dedicated to the mortar projector varied. This resulted in an optimal ratio of one mortar partition to 8 non-mortar partitions. This ratio was then used to perform calculations for a range of processor counts, as shown in Table 3. As can be seen, the scalability is reasonable given the relatively small size of the problem.

Table 3: Speedup relative to 8-proc run of a simple 3D geometry with fixed ratio of mortar to non-mortar partitions

procs	CPU time (s)	Speedup (vs 8 proc)
8	25.4	-
16	13.3	1.9
32	9.1	2.8
64	4.0	6.4
128	2.4	10.6

3.5 Summary

The Elmer multi-physics code has been enhanced to perform more efficiently on HPC platforms. These enhancements have taken place in three areas:

- FETI: reported in detail in D9.1.1. Whilst speedups in parallel were negligible, the parallelisation of this aspect of the code has been a significant improvement, leading to the ability to solve larger problems as it can now exploit distributed memory architectures
- Hybridisation: this has led to admittedly modest improvements in performance on a small number of cores
- Rotating boundary conditions: this has led to relatively promising speed-ups for both the 2D and in particular the 3D cases

It should be noted that at this stage the HPC performance of Elmer presented here is based on a relatively small number of cores – this is for two reasons: industry requirements, where users currently are content with simulations run in the order of 256 cores; and the current modest scaling of the hybridised code. With regards the industry requirements, these are often for problems exploring a design space, so one can envisage many of these 256 core jobs running as parallel tasks for a design optimisation approach.

Looking to the future, it is envisaged that the hybridisation approach, once the issues of performance have been tackled, will lead to the ability to examine models with a much larger number of degrees of freedom, which in turn will require access to hundreds of cores, enabling Elmer users to explore new areas previously prohibited to them.

4 Delft3D

4.1 Introduction

Delft3D [8] is a world leading 3D modelling suite used to investigate hydrodynamics, sediment transport and morphology and water quality for fluvial, estuarine and coastal environments. As per January 1st, 2011, the Delft3D flow (FLOW), morphology (MOR) and waves (WAVE) modules are available in open source. Delft3D has over 350,000 lines of code and is developed by Deltares [9].

The FLOW module is the heart of Delft3D and is a multi-dimensional (2D or 3D) hydrodynamic (and transport) simulation programme which calculates non-steady flow and transport phenomena resulting from tidal and meteorological forcing on a curvilinear, boundary fitted grid or spherical coordinates. In 3D simulations, the vertical grid is defined following the so-called sigma coordinate approach or Z-layer approach. The MOR module computes sediment transport (both suspended and bed total load) and morphological changes for an arbitrary number of cohesive and non-cohesive fractions.

SURFsara have undertaken the Delft3D work in this task.

4.2 Workplan

The porting of Delft3D to several systems was a key part of the workplan. At the start of the work Delft3D used *automake*, *autoconf* and *libtool* to create a configuration script for installation, however there were some platform-specific dependencies so the first task was to modify this and try Delft3D on various platforms.

The next stage of the workplan was benchmarking. To this end, three test cases were obtained:

- Waal river: a model of one of the main rivers in the Netherlands, used to estimate the effect of lowering the groynes (rigid structures built to prevent movement of sediment) on flood level
- Zeedelta: a simulation of the Rotterdam estuary
- Sediment transport

These benchmarks were chosen in order to represent the real-life usage of Delft3D and to exercise different aspects of the code, in order to identify bottlenecks in performance.

The next stage of the workplan was to investigate the identified bottlenecks and where possible enhance performance.

4.3 Implementation

Porting

The porting process was non-trivial, and raised several inconsistencies and even bugs in the code which have since been tackled, and are detailed in the white paper [7].

Benchmarking

The application consists of mainly Fortran 90, with some routines in C and C++ and some features from FORTRAN 2003. It uses MPI with 1-D domain decomposition as its parallelisation strategy, where it automatically selects the longest dimension to be partitioned. The length of the domain (i.e. the direction with most grid-points) is split across MPI processes. It uses an alternating direction implicit (ADI) method to solve the momentum and continuity equations. The parallel implementation of the ADI method in Delft3D assumes for parts of the computations that the halo regions of the computational domain on the processor are the boundary conditions. Therefore, convergence could become a problem when scaling up to higher process counts. I/O is implemented using a master-only technique.

The MPI routines are wrapped in custom routines that are mostly used for halo exchanges and the reduction of convergence parameters. Halo exchanges are executed by two calls to *MPI_Isend* and *MPI_Irecv*, immediately followed by separate calls to *MPI_Wait* for all communication. The haloes are stored in temporary arrays.

The approach taken for benchmarking the test cases was to instrument the code using Scalasca [10], and then identify the dominant routines for each case on a variety of processor counts.

4.4 Results

During the course of this work Delft3D has been ported to several systems including an IBM Power6 system, Intel Xeon Nehalem system and BullX Intel Xeon cluster, utilising a variety of different MPI libraries. Unfortunately attempts to port to the IBM BlueGene/Q system *Fermi* and the Cray system *Hermit* were unsuccessful, but the porting work continues and there is a known issues list.

With regards to benchmarking and performance enhancement, the results obtained for the Waal river test case show that the computational core of Delft3D can scale to 1000 cores with a suitable input dataset that is both large enough and has a regular domain that is homogeneously distributed across processes.

For the Zeedelta model, a real production case, which in comparison to the Waal river, has a much more heterogeneous domain with a high number of inactive grid points, the code scales to a more modest 128 cores. However, much of this runtime is taken up by the single-threaded master I/O, and there are other areas which have been identified for further investigation, optimisation and parallelisation. The I/O issue has started to be tackled but will not be completed within this task.

For the Sediment Transport test case, due to the size of the model and the nature of Delft3D the maximum number of processors that could be used is 160. In this case the I/O is much less significant – in fact, the main bottleneck are several small functions which can be called 10s of billions of times even for short runs. The enhancement possible here is to ensure that they are inlined correctly by the compiler, but this has its own challenges as several are in other libraries, both static and dynamic, preventing some automatic inlining.

4.5 Summary

Delft3D is a complex application used in a broad range of real-world applications, and as such has large number of different modules which can be activated separately depending on the problem at hand. The code has now been enhanced to make it much more portable to different platforms, and has been demonstrated to work on several different architectures.

In the course of this work, three industrially relevant test cases have been comprehensively benchmarked to investigate bottlenecks within the code. These bottlenecks have been documented, and are in the process of being tackled.

5 GROMACS/CP2K framework

5.1 Introduction

During the course of the second year of PRACE-2IP-WP9, the PRACE partners at NCSA, Bulgaria (NCSA-BG) were approached by a local industrial company from the pharmaceutical domain (*Biovet* [12]) concerning better utilising HPC resources. It was felt that this was entirely in line with the goals of task 9.1, so NCSA-BG engaged with the industry partner and their work so far is reported here. As of date of publication there is no white paper planned for this work, so it is explored in more detail in this deliverable than the other investigations. Note that the work is on-going and will extend beyond the end of Task 9.1, under other work packages in PRACE-3IP.

The research collaboration between Biovet and NCSA-BG, within the framework of PRACE-2IP, is in the field of computer aided drug design. The Bulgarian company Biovet is a manufacturer of fermentation-derived active pharmaceutical ingredients and drug products for animal health, human health, food and industrial applications. One of its products, *Tiamulin*, has wide applications in the field of veterinary medicine. Tiamulin acts by inhibiting protein synthesis at the ribosomal level. The company is interested in the discovery of new products, similar to Tiamulin antibiotics, with better biological activity.

The goal of the joint task is for the NCSA-BG team to establish a protocol suitable for drug design study, and to develop and validate it in a test case, together with the partnering industrial company, based on Tiamulin-like drug candidates able to bind the E.Coli ribosome peptidyl transferase centre. The outcomes of this would be a framework with the ability to:

- parameterise any drug like candidates in terms of CHARMM [17] family force fields by means of quantum chemistry calculations;
- create a GROMACS topology file;
- prepare input data for molecular dynamics simulations with GROMACS.

Molecular recognition and drug binding are very dynamic processes: when a small molecule like a drug (for example, a ligand) approaches its target (for example, a receptor) in solution it encounters not a single, static structure, but rather a macromolecule in constant motion. Large biological systems that are of particular interest to biology and medicine still cannot be investigated with ab initio and density functional theory (DFT) molecular dynamics (MD) methods because of their size. Nowadays, these systems are studied with molecular mechanics (MM) MD methods, with all the attendant negative consequences for the accuracy of the study. There are many force fields that can be used in such MD studies: most accurate of them are *all atom* force fields.

Many of the R&D groups in small and medium pharmaceutical companies construct new structures of drug candidates primarily based on the structures of known drugs, and clear

empirical relationships such as the partition coefficient (realised as log P [13]). Most of these structures are difficult and time consuming to synthesise, and experimental study of the medicinal properties is an expensive process. An easier way is the initial screening of the drug candidates by computer modelling. A commonly used technique for screening candidates is *docking* (predicting the preferred orientation of a molecule with respect to another), but its results are not always accurate: depending on the docking parameters, algorithms and methodology used, it can lead to either false rejection of potential structures of drug candidates, or to too large structure sets with a lot of non-potential candidates. The second issue is mainly due to the static character of the model, lack of mobile water molecules and inability to equilibrate the drug-target system in conditions close to the biological ones. One can overcome these disadvantages by applying a molecular dynamics approach to a more realistic model. Molecular dynamics is a theoretical technique where the time evolution of an atomic or molecular system is studied by solving classical equations of motion numerically.

The main difficulty is how to calculate the forces acting on every single atom. The most accurate methods are ab initio methods but they are too time consuming and do not have good performance scalability for large systems as are those of interest in drug design studies. Molecular mechanics force fields description works quite well in the case when force field constants are determined appropriately for the system of interest. This however requires very educated and skilled staff to be engaged in the drug design process. Moreover the most popular force fields are specially tuned for proteins and nucleic acids and many general force fields do not describe all possible functional groups of interest in the drug design process.

To address the issues mentioned above, NCSA-BG have developed a framework which builds on two well-known software packages suitable for HPC MD calculations, namely GROMACS and CP2K. Both CP2K and GROMACS are open source packages distributed under GNU General Public License and recognised by the respective HPC communities.

In the context of the research collaboration, the NCSA-BG group relies on the capabilities of the Biovet Research and Development department and their facilities that will allow carrying out several test cases at a later stage. The industrial partner has a full range of equipment as well as the necessary specialists with different competencies to support on-going tasks.

5.2 Workplan

In collaboration with the researchers from the pharmaceutical partner, the NCSA-BG team established the following protocol suitable for drug design study:

1. Geometry optimisation of the drug candidate;
2. Frequency analysis of the optimised drug candidate structure;
3. Creating GROMACS topology;
4. Determine force constants and equilibrium parameters from 2;
5. Setting up the drug-target system;
6. Preparing input data for MD simulation;
7. Molecular dynamics simulation – production runs;
8. Analysis of the trajectories.

There is a need to develop different programme modules to implement the protocol in order to make the process as automatic as possible. Geometry optimisation and frequency analysis are carried out by means of quantum chemistry calculations with CP2K, so the program module should prepare proper input files for steps 1 and 2 of the protocol. For step 3 the *g_x2top* tool

from GROMACS software package is used. For the next step, namely the determination of the force constants and equilibrium parameters, another software model is required to extract the necessary data and calculate those parameters. Step 5 depends on the aims of the particular study, and here the industrial partner will decide how to construct the whole model system. The last programme module will prepare the input data for the MD simulation. Steps 7 and 8 also depend on the aims of the particular study, and it is left to those involved to decide how to proceed.

Considering the system size, the most time consuming part of the protocol are the molecular dynamics production runs, but GROMACS scales well on many HPC platforms. On the other hand the CP2K performance scalability has to be checked in the case of calculations of drug like molecules. The program modules to be developed do not need to be parallel because they perform a simple but important part of the work flow, and thus their benchmarks are not necessary.

5.3 Implementation

The program modules are written in Python, in order to make possible connections with other widely used molecular simulation software packages such as *PyMol*, *Chimera*, *Coot* etc in future development. Figure 1 below shows the overview of the work flow between the modules.

The first program module reads an XYZ file containing the structure of the drug candidate molecule, and generates a CP2K input file adding the necessary keywords for proper structure optimisation. The second one reads the optimised CP2K geometry and writes an input file with the necessary keywords for frequency analysis. The next module prepares the input file for the *g_x2top* program from the GROMACS package, using the optimised geometry of the drug-like candidate, and runs it.

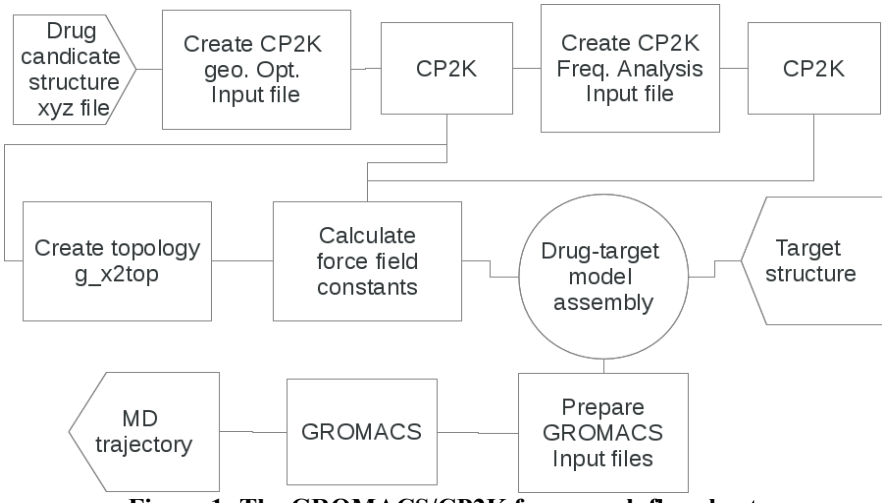

Figure 1: The GROMACS/CP2K framework flowchart

The most important module reads the hessian matrix, masses the optimised structure and the topology created by *g_x2top* and calculates all necessary force field constants. For parameter calculation the method published in [18] was used.

The Hessian matrix:

$$[k]=k_{ij}=\frac{\partial^2 E}{\partial x_i \partial x_j}$$ **Equation 1**

calculated with the CP2K frequency analysis run is used to determine intra-molecular force constants, where E is the potential energy of the molecule, x are the Cartesian coordinates of the N atoms in the system and i=1...3N and j=1...3N. Formulae from equations (10) to (19) from the article [18] have been implemented in the module, which also writes a GROMACS topology file containing the calculated force constants. The last module generates input files for molecular dynamics simulations and runs GROMACS.

The workflow is currently organised for running jobs only on the BlueGene/P and BlueGene/Q systems, but the functionality is easily extendable to other systems.

5.4 Results

As mentioned previously this work has been focussed on interest in the drug Tiamulin, which has wide applications in the field of veterinary medicine. Tiamulin acts by inhibiting protein synthesis at the ribosomal level. The NCSA-BG BlueGene team have established the protocol mentioned above and started development of Tiamulin-like drug candidates able to bind the E.Coli ribosome peptidyl transferase centre, using the developed framework.

The results concerning performance scalability of CP2K 2.5 (development version) running on IBM Blue Gene/P architecture are shown in Table 4 below. Since the number and the wall clock time of the geometry optimisation steps depend on the accuracy set by the user, the time for single point SCF calculation vs. the number of IBM Blue Gene/P cores is presented. The test molecule is a structure of Tiamulin compound with 81 atoms, GTH-BLYP pseudo potential and two types of basis function set: *aug-DZVP-GTH* and *QZV3P-GTH*.

Table 4: Performance scalability of CP2K 2.5

Basis function	*aug-DZVP-GTH*		*QZV3P-GTH*	
Cores	**Calculation time (s)**	**Speed-up (vs 64 cores)**	**Calculation time (s)**	**Speed-up (vs 64 cores)**
64	5.5	1.00	6.1	1.00
128	4.5	1.22	4.9	1.24
256	2.4	1.62	3.6	1.69
512	2.8	1.96	2.9	2.10
1024	2.3	2.39	2.5	2.44

As one can see from the table the speed-up is more than 20% when the number of cores is doubled. For the presented test case, this means one can have a geometry optimisation and frequency analysis performed on 512 Blue Gene/P cores in less than an hour.

5.5 Summary

In this collaboration with the Biovet pharmaceutical company, a framework for computer aided drug design has been developed by NSCA-BG. The established protocol is currently under test with a test case with real parameters of interest to the industrial partner.

It has been demonstrated that molecular systems of interest, with a complexity and size as for the system mentioned above, can be studied in a reasonable time scale only with the help of powerful high performance computing platforms which is the aim of the PRACE-2IP project. All used software is under GNU GPL which gives opportunity to other R&D teams to use and to develop the software further.

At this stage, the framework described misses a graphical user interface (GUI): this is planned to be developed in the near future to make the framework more user-friendly.

6 Summary

In the first year, two activities dominated: the gathering of ISV and industry user data via the surveys, and the establishment of the enabling projects.

The principal results in this second year are:

- OpenFOAM – has had an OpenMP hybridisation implemented in several further modules. Several real industrial test cases have been collected and benchmarked. OpenFOAM is now available on several more PRACE platforms. Performance enhancement has proved to be challenging.
- Elmer – parallel performance has been further enhanced via hybridisation and an innovative approach to rotating boundary conditions.
- Delft3D – has had its portability significantly improved and been demonstrated to work on several new platforms. It has been benchmarked in detail and bottlenecks are now being tackled.
- GROMACS/CP2K – a framework has been developed to facilitate in the identification of new drug candidates via HPC

Although the PRACE-2IP project is coming to an end, in all cases the work on these codes is planned to continue, either in PRACE-3IP or via some other means. The in depth knowledge gained working with the codes on HPC systems will continue to be applied, for the on-going benefit of both industry and academic users. The results of this work can be used by anyone as they are made available as Open Source.