SEVENTH FRAMEWORK PROGRAMME
Research Infrastructures

INFRA-2010-2.3.1 – First Implementation Phase of the European High Performance Computing (HPC) service PRACE

PRACE-1IP

PRACE First Implementation Project

Grant Agreement Number: RI-261557

D7.1.2
Applications Enabling for Capability Science

Final

Version: 1.0
Author(s): Jussi Enkovaara (CSC)
Date: 25.06.2012

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: RI-261557**	
	Project Title: PRACE First Implementation Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: < **D7.1.2**>	
	Deliverable Nature: <Report >	
	Deliverable Level: PU *	**Contractual Date of Delivery:** 30 / June / 2012
		Actual Date of Delivery: 30 / June / 2012
	EC Project Officer: ThomasReibe	

* - The dissemination level are indicated as follows: **PU** – Public, **PP** – Restricted to other participants (including the Commission Services), **RE** – Restricted to a group specified by the consortium (including the Commission Services). **CO** – Confidential, only for members of the consortium (including the Commission Services).

Document Control Sheet

Document	**Title:** <**Applications Enabling for Capability Science**>	
	ID: <**D7.1.2**>	
	Version: <1.0 >	**Status:** Final
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2007	
	File(s): D7.1.2.docx	
Authorship	**Written by:**	Jussi Enkovaara (CSC)
	Contributors:	Fabio Affinito (CINECA), Kevin Stratford (EPCC), Marzia Rivi (CINECA), Vladimir Slavnic (IPB), Sami Saarinen (CSC), Fernando Nogueira (UC-LCA), Paul Melis (SARA), Eric Boyer (CINES), Alan Simpson (EPCC), Peter Michielse (NWO)
	Reviewed by:	Manual Fiolhais (UC-LCA), Dietmar Erwin (FZJ)
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.2	26/Mar/2012	Draft	Draft based on D7.1.1
0.3	09/May/2012	Draft	Included most project reports
0.5	25/May/2012	Draft	Draft for comments from WP7
0.6	05/June/2012	Draft	Draft for internal review
0.7	21/June/2012	Draft	Suggestions from internal review included
1.0	25/June/2012	Final	Final version

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure, Applications enabling, Capability science

Table of Contents

List of Figures

List of Tables

References and Applicable Documents

[1] PRACE-1IP Project deliverable D7.1.1 - Applications enabling for capability science.
[2] http://www.prace-ri.eu/T7-1-Whitepapers
[3] von Alfthan, S. *et al*.: Scaling Vlasiator using Hybrid MPI and OpenMP parallelization, PRACE white paper, http://www.prace-ri.eu/T7-1-Whitepapers
[4] Roe, P. L.: Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes, Journal of Computational Physics, 43, 357-372, 1981.

List of Acronyms and Abbreviations

BLAS	Basic Linear Algebra Subprograms
BSC	Barcelona Supercomputing Center (Spain)
CCE	Cray Compiler Environment
CEA	Commissariat à l'Energie Atomique (represented in PRACE by GENCI, France)
CINECA	Consorzio Interuniversitario, the largest Italian computing centre (Italy)
CINES	Centre Informatique National de l'Enseignement Supérieur (represented in PRACE by GENCI, France)
CPU	Central Processing Unit
CSC	Finnish IT Centre for Science (Finland)
CSCS	The Swiss National Supercomputing Centre (represented in PRACE by ETHZ, Switzerland)
CSR	Compressed Sparse Row (for a sparse matrix)
CUDA	Compute Unified Device Architecture (NVIDIA)
DEISA	Distributed European Infrastructure for Supercomputing Applications. EU project by leading national HPC centres.
DGEMM	Double precision General Matrix Multiply
DMA	Direct Memory Access
DNA	DeoxyriboNucleic Acid
DP	Double Precision, usually 64-bit floating point numbers
EC	European Community
EPCC	Edinburg Parallel Computing Centre (represented in PRACE by EPSRC, United Kingdom)
EPSRC	The Engineering and Physical Sciences Research Council (United Kingdom)
ETHZ	Eidgenössische Technische Hochschule Zuerich, ETH Zurich (Switzerland)
ESFRI	European Strategy Forum on Research Infrastructures; created roadmap for pan-European Research Infrastructure.
FFT	Fast Fourier Transform
FP	Floating-Point
FPU	Floating-Point Unit
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
GHz	Giga (= 10^9) Hertz, frequency =10^9 periods or clock cycles per second
GigE	Gigabit Ethernet, also GbE
GNU	GNU's not Unix, a free OS
GPGPU	General Purpose GPU
GPU	Graphic Processing Unit
HMPP	Hybrid Multi-core Parallel Programming (CAPS enterprise)
HP	Hewlett-Packard
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
HT	HyperTransport channel (AMD)
IB	InfiniBand
IBA	IB Architecture
IBM	Formerly known as International Business Machines
IDRIS	Institut du Développement et des Ressources en Informatique Scientifique (represented in PRACE by GENCI, France)
IEEE	Institute of Electrical and Electronic Engineers
I/O	Input/Output
JSC	Jülich Supercomputing Centre (FZJ, Germany)
KB	Kilo (= 2^{10} ~10^3) Bytes (= 8 bits), also KByte
KTH	Kungliga Tekniska Högskolan (represented in PRACE by SNIC, Sweden)
LBE	Lattice Boltzmann Equation
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
MFlop/s	Mega (= 10^6) Floating point operations (usually in 64-bit, i.e. DP) per second, also MF/s
MGS	Modified Gram-Schmidt
MHz	Mega (= 10^6) Hertz, frequency =10^6 periods or clock cycles per second
MKL	Math Kernel Library (Intel)
ML	Maximum Likelihood
Mop/s	Mega (= 10^6) operations per second (usually integer or logic operations)
MPI	Message Passing Interface
MPP	Massively Parallel Processing (or Processor)
NAS	Network-Attached Storage
NCF	Netherlands Computing Facilities (Netherlands)
NFS	Network File System
OpenCL	Open Computing Language
Open MP	Open Multi-Processing
OS	Operating System
OSS	Object Storage Server
OST	Object Storage Target
PGAS	Partitioned Global Address Space
PGI	Portland Group, Inc.
PI	Principal investigator, a person responsible for research project.
POSIX	Portable OS Interface for Unix

PRACE	Partnership for Advanced Computing in Europe; Project Acronym
PSNC	Poznan Supercomputing and Networking Centre (Poland)
QCD	Quantum Chromodynamics
QDR	Quad Data Rate
QR	QR method or algorithm: a procedure in linear algebra to compute the eigenvalues and eigenvectors of a matrix
RAM	Random Access Memory
SARA	Stichting Academisch Rekencentrum Amsterdam (Netherlands)
SGEMM	Single precision General Matrix Multiply, subroutine in the BLAS
SGI	Silicon Graphics, Inc.
SIMD	Single Instruction Multiple Data
SMP	Symmetric MultiProcessing
SNIC	Swedish National Infrastructure for Computing (Sweden)
SP	Single Precision, usually 32-bit floating point numbers
STFC	Science and Technology Facilities Council (represented in PRACE by EPSRC, United Kingdom)
TB	Tera (= 240 ~ 1012) Bytes (= 8 bits), also TByte
TFlop/s	Tera (= 1012) Floating-point operations (usually in 64-bit, i.e. DP) per second, also TF/s
Tier-0	Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1

Executive Summary

The WP7 "Enabling Petascale Applications: Efficient Use of Tier-0 Systems" in PRACE-1IP is responsible for providing petascaling support to European researchers for PRACE Tier-0 systems. The task 7.1 (Applications enabling for capability science) provides optimization service by organizing calls, by evaluating proposals for optimization projects, and by performing the actual optimization work. The optimization service is provided via PRACE type C Preparatory Access.

This is the final deliverable of Task 7.1. The deliverable reports the optimization projects carried out by the task 7.1. It reports also the current processes used for reviewing the proposals and assigning work to PRACE partners together with statistics about project proposals.

During PRACE-1IP, the task 7.1 performed six evaluation rounds of Preparatory Access proposals and carried out optimization work in 14 Preparatory Access projects. Projects have been largely successful, and in most cases the application code is now ready for petascale production usage.

1 Introduction

The role of task 7.1 is to enable researchers to efficiently use the PRACE infrastructure by providing petascaling and optimization services. The optimization tasks in 7.1 are short (6 month maximum) and intensive (up to 6 PMs), with the ultimate goal to improve the performance of an application on PRACE Tier-0 systems. Larger, long-term projects are handled by task 7.2 in collaboration with scientific communities.

The optimization service is available through two sets of competitive calls: an internal 7.1 call and regular preparatory access calls. The internal call was a one-off case, and it was reported in interim deliverable D7.1.1 [1]. The main route to optimisation service is within the PRACE preparatory access.

Currently, task 7.1 is divided into four subtasks: subtask A develops best practices for a PRACE optimization service, which includes, for example, executing calls and defining the evaluation processes. The subtasks B, C, and D focus on the actual optimization work on JUGENE, CURIE, and HERMIT, respectively, as these are the currently available Tier-0 systems. When more PRACE Tier-0 systems become available, subtasks related to these system architectures will be defined.

This report is organized as follows: Section 2 presents the current process for evaluating the regular preparatory access calls and organization of the related PRACE optimization work, together with statistics about performed evaluations. In section 3 the results of optimization projects are reported and section 4 summarises the work.

2 Preparatory access calls

The main process for scientific users to obtain petascaling and optimisation services from PRACE is through regular preparatory access calls as announced at www.prace-ri.eu/hpc-access. There are 3 types of preparatory access: type A (for scaling tests) and B (porting and scaling/optimisation work) involve only the applicants, while proposals for type C request support from PRACE experts, as delivered by task 7.1. Currently, the systems available for preparatory access are the IBM Blue Gene/P – JUGENE – hosted by GCS in Jülich, Germany, were the maximum number of available compute cores is 294912, and the Bull Bullx cluster – CURIE – funded by GENCI and installed at CEA, Bruyères-Le-Châtel, France. CURIE is based on general x86 architecture with a mix of thin and fat nodes interconnected through a QDR Infiniband interconnect. From December 2011 on also Cray XE6 system HERMIT at Höchstleistungsrechenzentrum Stuttgart (HLRS), Stuttgart, Germany has been available for preparatory access.

The objective of preparatory access calls is to prepare users and their application codes for PRACE regular calls, which require good scalability.

In addition to type C preparatory access projects, in the beginning of PRACE 1IP there was also work on 6 internal optimization projects selected from PRACE centres. The selection process of internal projects and their results were reported in the interim deliverable D7.1.1 [1].

2.1 Evaluation process

The preparatory access call was opened in November 8th 2011. The call is continuous, so that proposals can be submitted at any time, although, the proposals are evaluated only at fixed intervals. Currently, the evaluation cut-off is approximately every three months.

The type C proposals undergo both a technical evaluation by the computing centres hosting the requested Tier-0 system(s) and evaluation by task 7.1. The technical evaluation ensures that the project can be executed, e.g. that requested libraries, compilers etc. are available on the system.

For the 7.1 evaluation, a review group was formed in late 2010. Currently, the review group consists of 35 experts from different PRACE partners having a wide range of scientific and high performance computing expertise. The purpose of the 7.1 evaluation is to ensure that the proposed optimisation work is feasible, and that the PRACE contribution is justified e.g. as the work benefits a larger group of researchers. There are two aspects in the 7.1 evaluation:

1. Actual evaluation (feasibility of work plan etc.)
2. Search for partners to perform the enabling work

The actual evaluation is performed by two members of the review group per proposal answering the following questions:

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

The search for PRACE partners is done by the task leader in order to keep the evaluation process reasonably short it is done in parallel with the actual evaluation. The evaluations by 7.1 have been typically carried out in two weeks after the cut-off date, and the actual access to Tier-0 systems is granted then by PRACE AISBL. The time from the cut-off date to official approval by PRACE AISBL has unfortunately been relatively long, some times almost two months.

The process for finding PRACE partners is described in the following subsection.

2.2 Work assignment

The main guiding principles when searching for the PRACE partners for the optimization work are: benefit to the project and equality among PRACE partners. Based on these principles, the PRACE partners (with enough remaining 7.1 effort) will be contacted in the following order:

1. A partner having a strong link with the proposal. This kind of link could be for example previous work with the application code by the partner. Same country of origin between the Principal Investigator and PRACE partner is not necessarily a strong link in this case.
2. If no strong link exists, a brief summary about the requested work is sent to 7.1 mailing list so that interested PRACE partners can volunteer for the work.
3. If there are no volunteers, the list of experts is consulted and suitable PRACE partners are contacted directly. Suitable partners are those having knowledge e.g. about the algorithms or programming approaches used in the application code of the proposal.
4. From the volunteers or suitable partners, those with the least amount of previous enabling work with preparatory access proposals are contacted first.

If there are several partners with the same priority for work assignment, these partners will be contacted in random order.

Each accepted type C preparatory access project is assigned a contact person from WP7 who is responsible for coordinating or performing the PRACE work and who acts as contact person between the applicant and PRACE.

2.3 Evaluations performed

After the launch of preparatory access call in November 2010, six evaluation rounds have been performed by PRACE-1IP in 2011 with the following cut-off dates: January 26th, March 6th, May 8th, July 1st, September 1st, and December 1st. The evaluation rounds in 2012 are carried out by PRACE-2IP which continues the work. The number of submitted and accepted type C proposals together is shown in Figure 1, and the requested and accepted PMs for work by PRACE staff are shown in Figure 2.

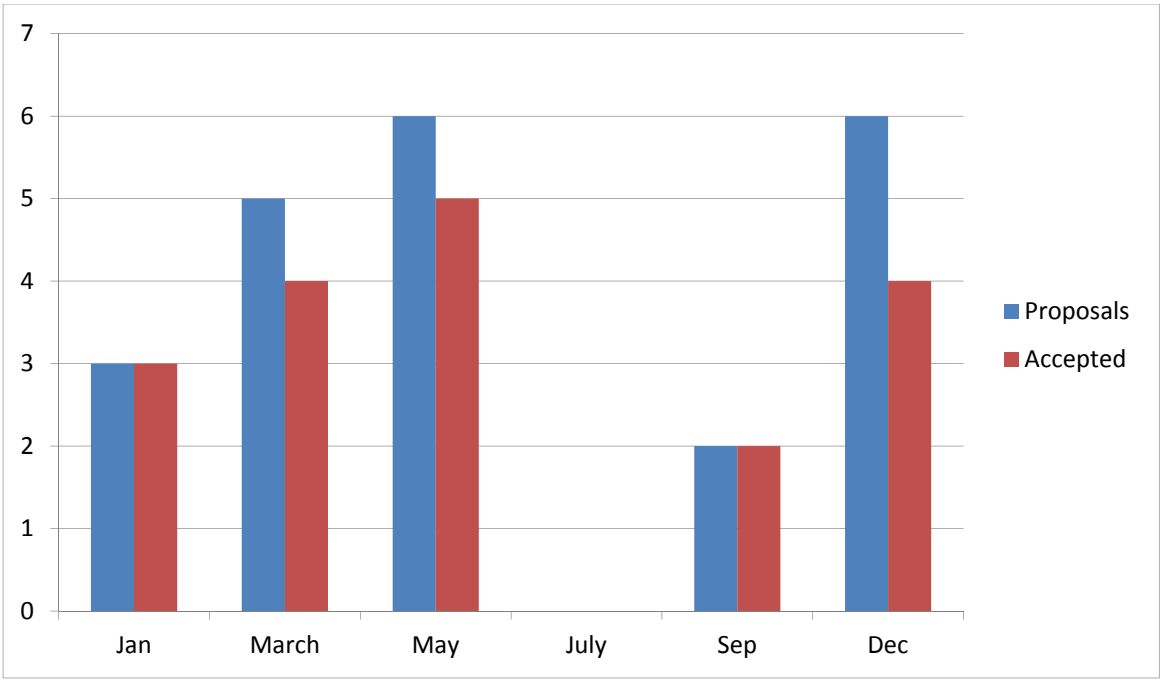


Figure 1: Number of submitted and accepted preparatory access proposals.

The variation in the number of proposals has been high, for example in July cut-off no type C preparatory access proposals were submitted, while in December there were again six submitted proposals. Similar variation is observed also in the number of type A and type B proposals. Overall, the number of type B proposals (27 accepted projects in 2011) has been similar to type C, while there have been clearly more type A proposals (49 accepted projects in 2011). The acceptance rate in all preparatory access types has been similar, typically 0-2 rejected proposals per evaluation round.

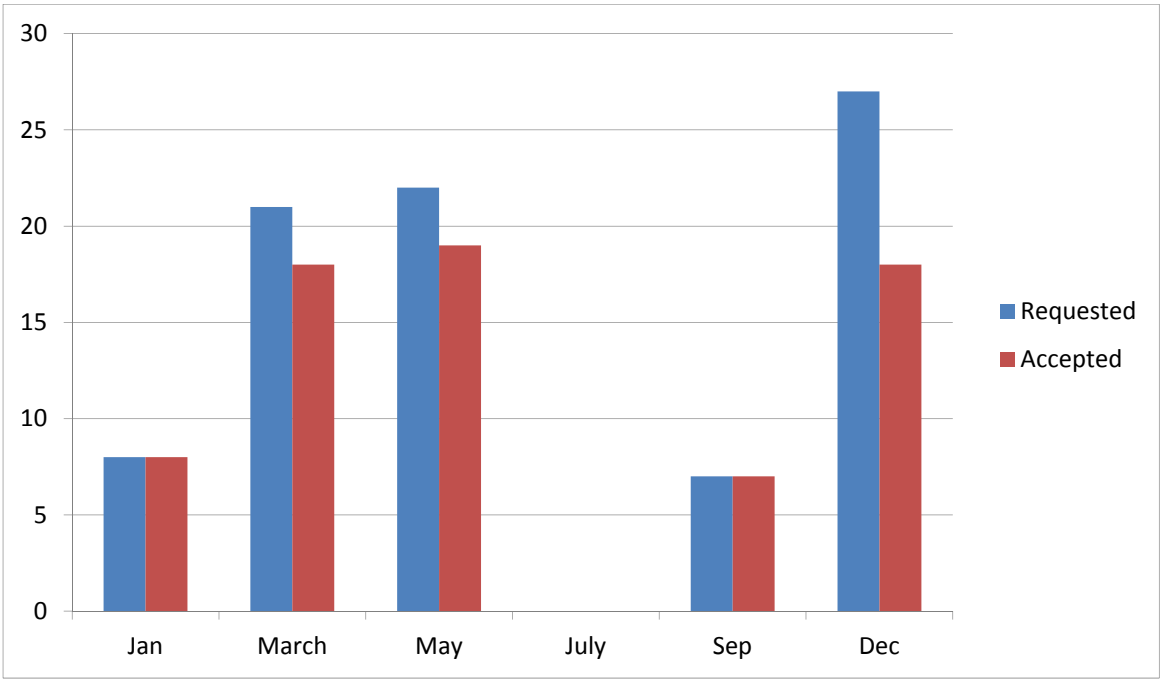


Figure 2: Number of requested and accepted PMs in preparatory access proposals

2.4 Dissemination

Especially in fall 2011 the number of submitted type C proposals was much lower than expected. In order to raise awareness of type C Preparatory Access among European researchers, an article about PRACE Optimization service available through type C Preparatory Access was published in the 5th PRACE Newsletter. Also, an email advertising type C Preparatory Access was distributed by PRACE partners to their users.

Possibilities of type C Preparatory Access are disseminated also by white papers about the successes in optimization work. These white papers are published in PRACE RI www-site [2].

3 Work performed in Preparatory Access projects

Here, work performed in twelve of the type C projects is reported. The projects accepted in September of cut-off are still ongoing, so work is not reported yet. In some of the projects the work has been done in collaboration with other WP7 tasks, e.g. with 7.5 in project 3.4 and with 7.6 with project 3.7.

3.1 Quantum Monte Carlo methods for biological systems

Project leader:	Leonardo Guidoni
PRACE staff:	F. Affinito, CINECA
PRACE facility:	JUGENE and CURIE
Research field:	Chemistry and Materials
Application code:	TurboRVB

3.1.1 *Project objectives*

Quantum Monte Carlo (QMC) methods are a promising technique for the study of the electronic structure of correlated molecular systems. QMC algorithms are highly parallel in nature and thanks also to the relatively small memory requirements also for large systems, they have good performances and scalability for highly parallel computers.

Recent implementations in the TurboRVB code were able to calculate in an efficient and scalable way the ionic forces, providing us with the possibility to perform full geometry optimization of molecules at the Variational Monte Carlo level. Three different run types are usually necessary for QMC calculations: Variational Monte Carlo (VMC), optimization of the variational wavefunction and Lattice Regularized Diffusion Monte Carlo (LRDMC).

The technical of the present project was to demonstrate the scalability of the TurboRVB code for a series of systems having different properties in terms of number of electrons, number of variational parameters and size of the basis set. The scientific goal will be the geometry optimization of a protein chromophore (about 100 atoms and 500 electrons) within the field of the protein matrix. The achievement of the targets of the present proposal are the basis for producing an accurate electronic structure tool with almost unique scaling capabilities among all other packages.

3.1.2 *Optimization work and results*

The TurboRVB code is already structured as based on a “embarrassingly parallel” kind of algorithm such as Quantum Monte Carlo. A large part of the code is mostly based on matrix-matrix multiplications using standard LAPACK routines, such as DGEMM. This makes the code highly efficient.

The optimization part concerning this Preparatory Access has been consisted on an accurate checking of the memory usage and in a proper way of using OpenMP threading mechanisms. For what concerns the memory access usage, a particular attention has been devoted to exploiting as much as possible the vectorization in cycles where it wasn't possible to use DGEMM factorization. The multithreading approach has been improved, introducing many OpenMP regions, in order to exploit the capabilities of architectures where the memory per

node is quite low (i.e. BG/P). Many analysis tools as Scalasca and Intel Vtune® have been used. These analysis led to some bug discoveries and their fixing.

The enabling process consisted in: a) analysis of the memory usage; b) analysis of the MPI and OpenMP parallelization levels; c) refactoring of the OpenMP regions implementations.

The above mentioned analysis led to the optimization of the memory usage and to the full exploitage of the hybrid MPI+OpenMP approach.

The target of the mostly optimization work was the memory sharing using OpenMP directives. The solution consisted in a refactoring of the #omp parallel do regions, in order to better exploit the parallelization over threads.

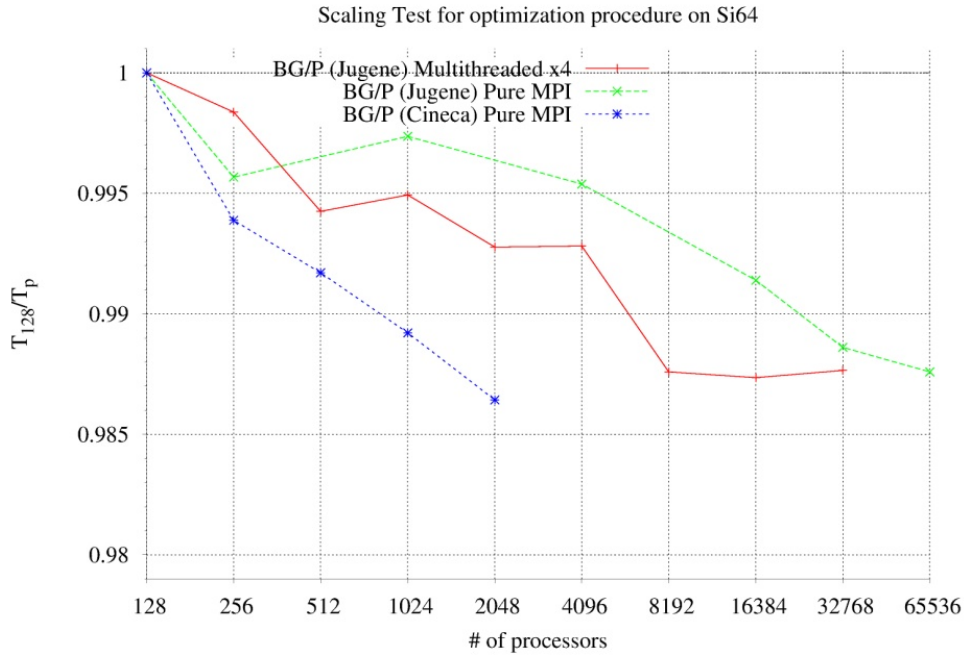


Figure 3: Weak scaling data for the Si64 atomic system (256 electrons) on JUGENE

Furthermore, we performed scalability testing reporting that the TurboRVB code is an ideal code for PRACE Tier-0 machines. In weak scaling the speed-up is 99% with 65k cores on JUGENE and up to 93% with 512 cores on CURIE. Strong scaling preliminary tests were difficult to be made on large systems of interests because total CPU budget and wall-time limitations. Anyway, on a test system we obtained a strong scaling speed-up of 85% on 4096 cores of JUGENE.

3.1.3 *Conclusions*

The TurboRVB code is ideal software for massively parallel computing systems such as PRACE Tier-0 machines. This achievements opens the door to the accurate study of the electronic structure of large (bio-)molecules. Future perspective might be the testing of the performances on hybrid CPU/GPU machines

3.2 Self organisation, pattern formation, and morphological instabilities in suspensions of mircoswimmers

Project leader:	Ignacio Pagonabarraga, University of Barcelona (UB)
PRACE staff:	Andrea Scagliarini, Giovanni Giupponi, Ricard Matas, Francisco Alarcon, Silvio R. Morales (all UB) Kevin Stratford (University of Edinburgh)
PRACE facility:	JUGENE
Research field:	Fundamental Physics
Application code:	*Ludwig* (lattice Boltzmann for complex fluids)

3.2.1 *Project objectives*

Active fluids are composed of particles that are able to move in a fluid environment as a result of their internal metabolism (in the case of microorganisms swiming), or as a result of specific and heterogeneous surface reactivity (e.g., in the case of chemically activated colloidal particles, which can be regarded as micron-sized robots).

In this project we aim to study, by means of large hybrid lattice Boltzmann simulations, the role of hydrodynamic coupling among swimmers in the formation of recognisable patterns, and the stability of these patterns. One common example is the transition of a community of swimming microorganisms to a flocking state. This is a challenging computational problem: as the motion of one swimmer affects another via forces mediated by the fluid, one has a system with many-body and long-ranged interactions.

From the practical point of view, our interest is to test the performance of the code on the JUGENE supercomputer. We have two specific aims: first, to extend the parallel scalability of the code for this type of complex fluid problem, and second, to assess how the scaling is affected by the large (and potentially unbalanced) particle distributions.

3.2.2 *Optimization work and results*

Ludwig is a simulation code specifically written to deal with complex fluids (which include

colloidal suspensions, fluid mixtures, liquid crystals, and so on). Work undertaken at the University of Barcelona extended the capability of code to include `self-propelled' objects, such as bacterial swimmers. The code is based on the lattice Boltzmann method for the solution of the Navier Stokes equations for fluid flow, which is a local algorithm suitable for highly parallel computers. *Ludwig* conforms to the ANSI C standard and uses the Message Passing Interface (MPI). The code was used on one of the first Blue Gene / L machines in Edinburgh, and presented no portability issues in moving to the Blue Gene/P JUGENE. We obtained performance metrics for the code using the inbuilt timing routines.

Objects, such as bacteria, may move freely throughout the problem domain in the *Ludwig* code. Behaviour such as flocking may then cause the distribution of these objects to become inhomogeneous on the scale of the domain decomposition and so induce a load imbalance. To investigate the potential for such load imbalance, we have run simulations on systems of size up to 1024 lattice sites per side of the system, containing up to 65,000 swimmers. On short runs, the code performs very well up to 32,768 processors in terms of both strong and weak scaling. There is some evidence of a slight degradation in performance compared with fluid-only problems when the solid volume fraction reaches 5%. This is likely to be load imbalance, which is exposed by the additional communication required for the particles. It is unlikely in

the runs we have performed so far that significant imbalance has resulted owing to pattern formation such as flocking.

We have also tested the I/O mechanism in the code, which requires some investigation to find the decomposition that most efficiently uses the available bandwidth to disk. This usually means some trial and error for problems of a given size.

3.2.3 *Conclusions*

Our results have confirmed that the code displays very good scaling performance, and we have extended the scalability range investigated with swimmers on more than 30,000 cores. We observed a weak sensitivity to the presence of particles, which suggests that the current implementation is reasonable for these systems. Although only short runs have been carried out, which means that characteristic behaviour such as flocking had no time to evolve, the work provides some confidence that load imbalance is not an overwhelming problem for this type of work. Possible future work may involve a more systematic analysis in the particle-activity parameter space on larger simulation time scales, to explore how the emergence of dynamic interactions among colloids or bacteria affects the scalability.

There is some evidence from other work using Ludwig (involving liquid crystals, where the solid/fluid boundary condition is more computationally expensive) that load balance becomes more pronounced if strong clustering occurs. However, the loss in efficiency is still sufficiently small that it would probably not be worthwhile going to the expense of rewriting the code to perform some form of dynamic load balance (even if that were possible).

3.3 Petascale Astrophysical Simulations of Accretion Processes with PLUTO

Project leader:	Andrea Mignone, University of Turin, Italy
PRACE staff:	Giuseppa Muscianisi, CINECA
	Marzia Rivi, CINECA
PRACE facility:	JUGENE
Research field:	Astrophysics
Application code:	PLUTO

3.3.1 *Project objectives*

Accretion is a fundamental process for many astrophysical objects and the associated release of gravitational energy can power some of the most energetic phenomena in the universe. Accretion mainly occurs through a disk and a basic problem of the theory is to determine how the material in the disk can effectively accrete losing its angular momentum. This process of angular transport in the accretion disk is currently interpreted as the result of magneto-hydrodynamical turbulence driven by the magneto-rotational instability (MRI).

The range of length scales in the problem, going from the global disk scale to the dissipation scale, is enormous and inaccessible to computation, whereas calculations with a sufficient scale separation may give important answers. Keeping the same resolution that is currently achieved in local simulations it is possible to estimate that a grid with $\approx 10^9$ points in necessary to model the whole disk: its evolution then needs to be followed for a few hundred rotation periods, in order to have enough statistics on the disk turbulence properties.

Extrapolating from current PLUTO performances on BlueGene/P platforms, in order to carry out Petascale computations we predict a required total time of a few hundred million processor hours and, being a production run, a reasonable size for the output data would be around ≈300 TB (assuming one snapshot per disk rotation). Therefore, an optimization work aimed at reaching efficient Petascale performances is crucial.

3.3.2 *Optimization work and results*

The main bottlenecks present in the PLUTO code were related to the I/O implementation provided by ArrayLib, a MPI-based library supporting parallel finite difference/finite volume computations on block structured meshes. The library (which has no longer been maintained since 2001) suffered from a number of flaws and implementation bugs which could severely limit the code performance on Petascale systems and make future additional extensions rather difficult to implement. Moreover, the standard procedure for raw binary I/O operations was implemented through collective and blocking I/O calls where every processor independently accessed the same file. In configurations with a very large number of processing units and grid sizes this approach has been found, on some system, to lead to execution hangs and/or considerable slow down and efficiency loss. To overcome such limitations, a number of actions have been performed on both the ArrayLib and the PLUTO code. Part of this work has been performed within task 7.6 of WP7.

The enabling process started with the porting of the PLUTO code on the JUGENE Tier-0 system. Subsequently, the assessment of the code performance “as it was at the start of the project” was performed. In particular, a detailed profiling of both the communication and the I/O parts handled by the ArrayLib has been done.

In its original implementation, PLUTO performed binary I/O operations at specific times during which each processor gained independent access to the file and wrote each variable through blocking and collective calls from within an iteration loop. This step was followed by a number of collective MPI communications not involving the main integration dataset.

By aiming at improving the performance of reading/writing raw binary data in both single and double precision, the ArrayLib has been modified by conveniently replacing the previous I/O calls with 'nonblocking' and 'split collective' calls, available in the MPI-2 I/O standard. As a result, variables are now dumped to disk all together by setting a unique view of the whole file and by building a global sub-array describing how the data of each process has to be written in the file. Tests on a grid of 512x4096x512 points, involving more than 4096 MPI processes and intensive I/O (i.e. output files written at each step of the integration loop) have shown that the new nonblocking version of the code is able to save more that 20% of the time with respect to the previous blocking version. By involving 8192 MPI tasks, the saving increased up to 30%.

Afterwards, following the parallelization strategy implemented in PLUTO, the usage of HDF5 library has been extended also to the static grid version of the code. Two HDF5 available file drivers have been tested: MPI-POSIX and MPI-IO, the latter using both the 'independent' and the 'collective' access. The benchmarks have shown that on JUGENE system the usage of the MPI-IO file driver with a collective access to the dataset yields the best performance. The introduction of the HDF5 file format for static grid represents an improvement for PLUTO both in term of portability and also for post-processing and visualization purposes.

Finally, a bug was fixed in handling staggered arrays in the ArrayLib. In particular, the conversion routines between local and global addressing of the arrays was analyzed and conveniently modified. As a conclusion, ArrayLib is now able to handle both cell-centered and staggered array in a correct and efficient way.

3.3.3 *Conclusions*

The enabling activity on the code can be summarized in the following points:

1) ArrayLib has been largely debugged, upgraded and simplified resulting in a more compact set of routines. The major achievement concerns the correct implementation of the distributed array descriptor handling staggered mesh arrays;

2) modification of writing of raw binary data in both single and double precision by using an asynchronous and split collective approach, available in the MPI-2 IO standard;

3) implementation of the HDF5 file format (previously available only in the adaptive grid version of PLUTO) in the static grid version of the code. We have implemented, tested and compared both the MPI-IO and the MPI-POSIX file driver. On JUGENE, a better performance was obtained using the MPI-IO file driver enabling collective buffering for accessing the HDF5 dataset.

These optimizations allowed to achieve high-quality results which have greatly expanded the code capabilities in terms of flexibility, enhanced I/O features and performances and portability. They warrant a major release of the PLUTO code, from 3.1.1 to 4.0, which will be made available to the astrophysical community by the end of 2012. The achievements give strong and encouraging indications that global disk simulations on Petascale computing systems should now be feasible with the PLUTO code, provided enough computational resources are allocated. This will open for potential scientific innovation in the field of accretion flows and angular momentum transport in disks through high-resolution numerical simulations.

3.4 Optimizing a 6D global Vlasov simulation of Earth's magnetosphere

Project leader:	Sebastian von Alfthan, FMI
Project team members:	Ilja Honkonen, FMI
PRACE staff:	Ata Turk, Gunduz Vehbi, Cevdet Aykanat (Bilkent) Dusan Stankovic, Vladimir Slavnic (IPB)
PRACE facility:	JUGENE and CURIE
Research field:	Astrophysics
Application code:	Vlasiator (6D global Vlasov simulation of Earth's magnetosphere)

3.4.1 *Project objectives*

Space weather is a term used when the near Earth space environment threatens technological systems of humans. Space weather forecasts are currently being tailored, as the simplest numerical techniques can be adopted in large-scale simulations to model the electromagnetic plasma systems within the near Earth space. Finnish Meteorological Institute (FMI) has the only large-scale magnetohydrodynamic (MHD) simulation in Europe, and this kind of simulation is successful in describing systems of large spatial scales.

Space weather modelling teams around the world are answering the space weather forecasting challenge by improving their MHD model accuracies using code coupling. FMI develops a Vlasov-hybrid simulation where electrons are a charge neutralizing fluid, and ions are modelled as distribution functions, enabling the description of multi-component plasmas without noise and in scales unreachable by MHD. A large-scale Vlasov-hybrid simulation is highly challenging, as in every grid cell of the ordinary space resides a velocity space, making the simulation 6-dimensional (6D). This sets demanding quality criteria for the solver, because the simulation needs to be executed in $\sim 10^6$ grid cells for $\sim 10^6$ time steps indicating peta-scale simulations. At the moment, the core solver is implemented in a massively parallel grid.

In this project we have strived to improve the scalability and performance to enable petascale simulations using Vlasiator. This will enable global Vlasov simulations of Earth's magnetosphere for the first time, opening up new avenues for research.

3.4.2 *Optimization work and results*

A number of techniques were employed to improve the performance of the Vlasiator code:

Improved load balancing

In general the problem in load balancing is twofold: the time required to perform a load balance should be reasonable at scale, and the quality of load balance should yield good efficiency. At the moment load balancing in Vlasiator is performed using the Zoltan library (http://www.cs.sandia.gov/Zoltan/). This library enables one to use one of the several different load balancing algorithms. In our work we obtained best results using the hypergraph partitioner, but the main limitation was that the time required to partition the system scaled as $O(N^2)$, where N is the number of MPI tasks. At the moment Vlasiator only needs one initial load balancing operation. When adaptive grids will be implemented in the future this will not be the case but regular load balancing operations will be needed. In the load balancing work several algorithms and backends were tested in Zoltan to investigate if results could be improved, both with regards to load balancing time as well as the quality of the result. More details on Vlasiator load balancing can be found in a separate PRACE white paper [3].

Hybrid OpenMP-MPI parallelization

In pure MPI mode there are several bottlenecks where a hybrid approach could help, such as: enlarging total number of cells per process, which means that there are more internal cells on a process helping us to hide communication, better load balancing performance (because of the lower number of processes N) and quality (because good load balancing is difficult with few cells per process), and there are fewer process boundaries, reducing overhead associated with these boundaries.

The threading was implemented in a fine grained manner, where the most time consuming computational loops were parallelized using OpenMP. MPI communication was implemented in a funneled mode, where all MPI communication is handled by the master thread. This was shown to be a major bottleneck, because using fewer MPI processes per node on CURIE reduced network saturation and increased communication times. Unfortunately, no MPI implementation supporting thread multiple mode of communication was available on CURIE, so it is possible that even greater performance gains could be achieved using one if it becomes available to users.

To study the performance boost of the hybrid version we simulated the difficult case, namely simulating very few cells per process. Best performance was obtained when using 8 threads per MPI process, which corresponds to one process per processor on CURIE. When looking at the total performance, we found out that up to 45% more cells are propagated per second when using hybrid approach than when using pure MPI, which is already a significant boost. This is both due to a reduction in the amount of time spent in MPI routines, as well as in the time spent computing. We have also tested hybrid mode with both Intel and GNU compiler suites, to show how choice of the compiler and OpenMP implementation can affect the performance of the application. Code compiled with the Intel compilers showed significantly better performance, and the difference was greater as the number of threads per MPI process increased. This was tested for the case with 64 spatial cells per CPU core, and 8 threads per MPI process was again the best combination.

It can also be noted that doing a load balancing using Zoltan is up to 28 times faster using the hybrid mode, partly solving the problem with long partitioning times. More details on Vlasiator hybrid implementation can be found in a separate the white paper [2].

Investigating scalability and performance of the DCCRG grid

As part of PRACE preparatory access the parallel performance of the grid (dccrg) used in Vlasiator was tested. Several 1, 2 and 3 dimensional MHD tests were run in order to quantify the scalability of dccrg (http://gitorious.org/dccrg). The MHD tests used a first order finite volume method solver based on Roe's approximate Riemann solver [4]. In the tests the total number of cells and MPI processes were varied and one MPI process per core was used. Close to linear scaling is obtained up to 32000 MPI processes by using 1 million cells and the maximum performance obtained from JUGENE in this test is slightly above 150 million total solved cells per second. The computational load was balanced at the start of the simulation by using a Peano-Hilbert space-filling curve. The large memory per node in CURIE allowed us to use up to 64 million cells in our scalability tests on that machine. For 64 million cells close to linear scalability is obtained in CURIE with peak performance of about 350 million total solved cells per second. On the other hand when using the same number of cells in CURIE as in JUGENE parallel performance does not improve beyond 512 MPI processes with peak performance at around 20 million total solved cells per second.

Topology aware partition placement

We have also investigated the influence of placement on CURIE. To do this we developed a small library that is able to optimize the placement of work partitions. In our case we first use Zoltan to partition the work, and then optimize the placement of these partitions on the processes. The optimization is based on fitting the communication graph in the application on to the network topology of the machine. The results obtained with a simple benchmark where each process is allocated exactly one cell, and each process exchanges data with its nearest neighbours show that we can increase throughput by more than 40% using better placement, both for small and large messages. In practice the benefit for our Vlasiator code is not that large, partly due to the hybrid mode already reducing the number of process to 4 per node which makes the default placement optimal on a per processor basis.

3.4.3 *Conclusions*

At scales of over 10000 processes the current load balancing library (Zoltan) is becoming a major bottleneck. In the future we will need dynamic load balancing which will even further put the emphasis on finding a scalable solution to the load balancing problem. Using hybrid OpenMP-MPI mode reduces this problem, as the number of processes is reduced. JUGENE proved to provide excellent scalability, but the compute performance of our C++ code was poor, and the limited amount of memory was also problematic. CURIE showed good weak scalability and good performance. Better network throughput should improve performance, as the code is bottlenecked by MPI communication. MPI+OpenMP hybridization proved to be an excellent match for our code, and helps in all areas of performance. There are still some issues related to hybridization that are not solved at the moment, such as the support for thread multiple communications inside of MPI library. If the adequate MPI implementation (supporting MPI_THREAD_MULTIPLE mode) becomes available, that could help in better network utilization while still having all the benefits of code hybridization.

3.5 Turbulent convection and dynamos in spherical wedges

Project leader:	Petri Käpylä, University of Helsinki, Finland
PRACE staff:	Sami Saarinen (CSC)
PRACE facility:	CURIE
Research field:	Astrophysics
Application code:	Pencil

3.5.1 *Project objectives*

At Solar physicists believe that the magnetic field of the Sun is generated by a turbulent magneto-hydrodynamic (MHD) dynamo operating within the convection zone, i.e., in the outer thirty per cent of the solar radius. In this shell, the radiative transport of energy is inefficient and fluid motions transport the energy to the surface. Such fluid motions are highly turbulent in the Sun are held responsible for the generation of solar differential rotation and magnetism. The goal is to understand and model these phenomena by performing massively parallel direct numerical simulations of the equations of magneto-hydrodynamics (MHD) using purpose-made Pencil Code.

The Pencil Code is programmed in Fortran 90 and parallelization is done via domain decomposition in all three dimensions with MPI. The main performance bottleneck is related to scalability of the code on a large number of cores. An additional issue is related to the use of

spherical coordinates, due to which additional terms need to be included. The performance of the code (i.e. wall clock time/grid point/time step), is worse than expected due to the additional geometrical terms.

3.5.2 *Optimization work and results*

Pencil Code with DrHook (developed at ECMWF by the author) instrumentation calls has been ported to CURIE system in order to find the performance bottlenecks.

There were four different model inputs for this study: L & XL sizes in Cartesian and Spherical coordinate systems. Instead of running hundreds of thousands of time steps each time, just 50 were run to avoid running out of the allocated time quota on CURIE. This was already sufficient to obtain scaling data because the amount of computations remains the same irrespective of the length or physical state of the simulation. In search for more parallelism, the use of OpenMP as an additional parallelization dimension had to be abandoned for this study as there were potentially only for 3-way OpenMP-parallelism available in each MPI-task. There were also plenty of shared variables in the system and their isolation could be a tedious task. Furthermore 3-way is a bad number when a computer system like CURIE has 32 cores in a shared memory. The MPI-implementation on CURIE also did not support OpenMP thread-safe communication.

There was also an effort to examine whether going away from asynchronous MPI send/receives to more synchronous MPI sendrecv-semantics on 512 tasks and beyond would make boundary exchanges less vulnerable to communication bottlenecks: this exercise significantly, however, increased the total elapsed time and thus was not employed.

Minimal optimization (10-20%) and increase of scalability was achieved by:

- The random number generator was streamlined and is now much faster and gives bit-identical results.

- Some really small I/O done by each MPI-task has been streamlined and in some cases only the root task does that I/O (in fact: output). For example, there was a counter displaying the time-step the program is currently processing. For this purpose each MPI-task was writing a single number to a separate file and flushing it. This was changed so that only the root-process kept this kind of monitoring information coming.
- Scalability bottlenecks were discovered and they were not computation, but communication bound
- There seems to be no benefit in hand-optimization of the expensive derivative calculations as the Intel Fortran compiler (ifort) already does such a good job. Also these routines scale perfectly with the increase of MPI-tasks, so they are really not a big obstacle for scalability. This is to say, they were already professionally coded.
- Spherical is by nature more expensive than Cartesian due to its multiplicative/dot-product –like operations to take into account the coordinate transformation.

Results obtained:

- Modest performance improvements (10-20%) at high (>1024) cores were achieved. The main bottleneck in CURIE appears to be in the communication, which is hard to avoid with the present setup. The planned setup with grid resolution 1024x2048x1024 scales fairly well until 2048 cores (i.e. speedup of 10.5 in comparison to 128 cores) but not higher.
- Scaling on CURIE was somewhat disappointing given the previous experiences in other types of machines (in particular Cray XT4/5 and XE6).
- The main performance bottleneck is due to heavy communication as expected. However, some of the remaining bottlenecks of system are MPI_ALLREDUCE summation of three coefficients, which in this study seem to be equal to zero. If these numbers evolve becoming non-zero, then perhaps communicating them a bit less often could be a wise thing to do. This is of course a potential algorithm changer. Also, boundary data exchange, despite seemingly well coded using asynchronous MPI, holds up the progress of integration and severely limits scalability. It looks as if there is not enough computation between posting asynchronous MPI send/receives and calling the subsequent MPI waits. One suspect is that the Open-MPI message passing library on CURIE is not processing the messages asynchronously at all, but only until MPI waits are issued. The problem gets really exposed at around 512 tasks and beyond.
- At very high task (compared to the model sizes) -- say 2k or 4k -- some additional scalability "barriers" are limiting the progress: for example calls to routine STOP_IF_ANY tests are rather bad.

3.5.3 *Conclusions*

The main interest was to study and improve the scalability of the Pencil Code, in particular a convection setup in Cartesian and spherical geometries relevant to the Sun, on CURIE. This was done in order to be able to apply for Tier-0 access with a large-scale project at a later time. Currently testing with different dimensions – both spatial and processor – requires a complete rebuild of executables and regeneration of processor-wise input data. This should be fixed as a high priority item in order to make testing and prototyping less time-consuming and less error-prone.

3.6 New algorithms in Octopus for the Petaflops computing

Project leader:	Joseba Alberdi Rodriguez, Universidad del Pais Vasco/Euskal Herriko Unibertsitatea
PRACE staff:	Fernando Nogueira, UC-LCA Micael Oliveira, UC-LCA
PRACE facility:	JUGENE and CURIE
Research field:	Condensed Matter
Application code:	Octopus

3.6.1 *Project objectives*

Octopus is a computer code to calculate excitations of electronic systems, i.e., to simulate the dynamics of electrons and nuclei under the influence of external time-dependent fields. The code relies on Density Functional Theory (DFT) to describe the electronic structure of the system under study and uses the Time-Dependent formulation of DFT (TDDFT) for the calculation of electronic excitations. In Octopus the functions are represented in a real space grid and the differential operators are approximated by high-order finite differences. A hybrid parallelization scheme is implemented, based on a distributed memory approach using the MPI library for the communication between processes. This is combined with fine-grained parallelism insid each process using either OpenCL or OpenMP.

The MPI parallelization is mainly based on a tree-based data parallelism approach. The main piece of data to be divided among processes are the Kohn-Sham (KS) states, an object that depends on three main indices: a combined k-point/spin index, the state index, and the space coordinate. Each one of these indices is associated with a data parallelization level, where each process is assigned a section of the total range of the index. The first two levels of parallelism are very efficient, as very little communication is required during the real-time TDDFT propagation of the KS states. The third level is more complex, as it consists in assigning a certain number of grid points to each process. In practice, the space is divided in domains that are assigned to each process. As the finite-difference operators, like the Laplacian, require the values of neighboring points, the boundary regions between domains need to be communicated between processors.

With this parallelization scheme, the code was shown to scale rather well up to several hundreds of cores, but scalability was not so good when using thousands of cores and was thus not suitable for petaflop/s systems. The main bottleneck was identified by the developers of the code and they applied to a type C preparatory access call in order to get PRACE support in removing this bottleneck. This bottleneck was the calculation of electrostatic potentials though the solution of the Poisson equation:

$$\nabla^2 v(\vec{r}) = -4\pi n(\vec{r})$$

where $n(\vec{r})$ is the electronic density and $v(\vec{r})$ is the so-called Hartree potential. The default method in Octopus to solve the Poisson equation is to use FFT's. This is done in serial through the use of the FFTW library and in parallel with the ISF library, which is part of the BigDFT code (http://inac.cea.fr/L_Sim/BigDFT).

3.6.2 *Optimization work and results*

After careful analysis of the code, it was decided with the developers to pursue two different lines of action to improve the scalability of the Poisson solver. The first one was to improve

the use of FFT's, while the second was to try a different method to solve the equation based on the Fast Multipole Method.

Concerning the improvement of the FFT method, we interfaced the code with a new library that performs FFT's in parallel. This library is called PFFT (http://www-user.tu-chemnitz.de/~mpip/software.php) and is better suited for petaflop/s systems. Initial results obtained with PFFT were quite promising, but a new problem was identified in an operation done by the code before and after the call to PFFT. Indeed, when using FFT's, functions need to be represented inside a cube or parallelepiped, while Octopus works with a mesh structure that is not necessarily a cube nor a parallelepiped. Furthermore, PFFT uses a 2D domain decomposition, while the mesh partitioning in Octopus is done in 3D. This means the data distribution is quite different in both cases and it is necessary to transfer the density and the corresponding Hartree potential from one representation to the other before and after calling PFFT. This operation was initially done by gathering the full function in all processes using a MPI_all2all instruction and them redistributing it. This made the time required to perform this operation increase with the number of processes, thus becoming a serious bottleneck. We implemented a new algorithm to perform this task, but where the minimum required amount of data is communicated between processes. This new algorithm was found to scale as well or even better than PFFT.

The code was also interfaced with the FMM library, which is a parallel implementation of the fast multipole method. This is an approximate method that reduces the complexity of solving the Poisson equation by using a multipole expansion to approximate the far- field effect from several charges into a single term. The library is designed to calculate the interaction between point charges, therefore, we had to assume that the electronic density in the grid corresponds to an array of point charges and then calculate a correction for the interaction between neighboring points. This correction term, essential to obtain the necessary accuracy, has the form of a finite-differences operator, and is therefore simple to evaluate in the framework of the code.

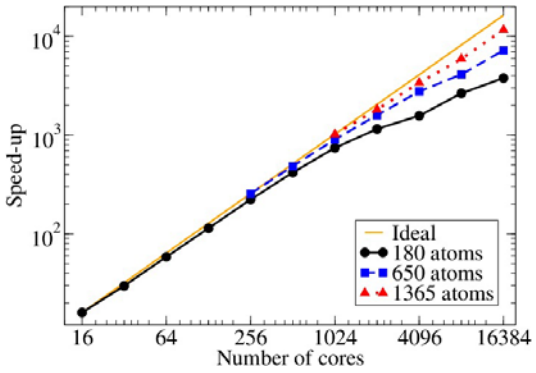


Figure 5: Comparison of the time required to solve the Poisson equation in a parallelepiped box of size 15x15x15 with different solvers.

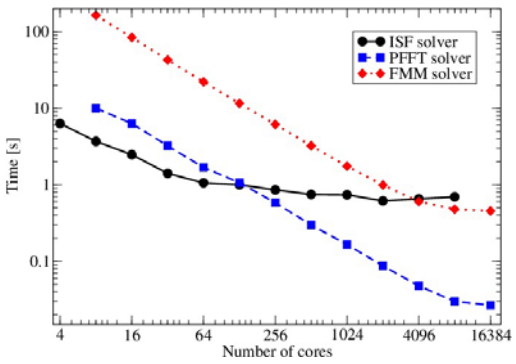


Figure 4: Scaling of time-propagation runs for systems of 180, 650 and 1365 atoms using the PFFT library.

3.6.3 *Conclusions*

In Figure 5 we compare the time required to solve the Poisson equation in a parallelepiped box of size 15x15x15. As we can see, the ISF method is faster for small number of processors, while PFFT and FMM become more competitive as we increase their number. The crossover point is quite low with PFFT method (128 processors) and higher with FMM (4096 processors). In Figure 4, we show the scaling for time-propagation runs for systems of 180, 650 and 1365 atoms using the PFFT library. Almost perfect scaling was achieved up to 512 cores with the system of 180 atoms, and a remarkable speed-up of 3785 was obtained with 16 384 cores, reaching an efficiency of 23% with 91 cores per atom. In the case of the

system composed by 650 atoms the ideal speed-up is reached up to 2048 cores, while for 16 384 cores a value of 7222 is achieved. Even better performance is obtained with the system of 1365 atoms, with a speed-up of 11646 for 16384 cores. Thus, it is possible to maintain an almost ideal speed-up with around 3 cores per atom, and acceptable speed-ups up to 50 cores per atom.

3.7 Visualization of output from Large-Scale Brain Simulations

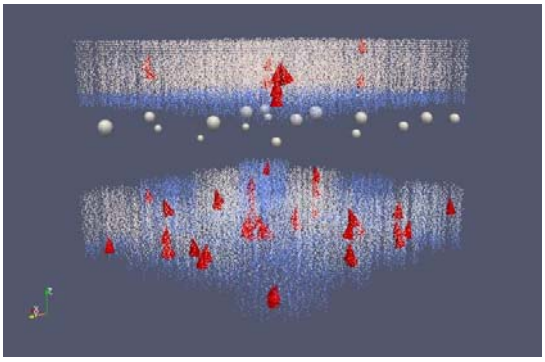
Project leader:	Anders Lansner (Dept of Computational Biology, CSC, KTH Royal institute of Technology)
PRACE staff:	Paul Melis (Visualization Group, SARA, The Netherlands) Vladimir Slavnić & Marko Spasojević (Scientific Computing Laboratory, Institute of Physics Belgrade, University of Belgrade, Serbia) Kiril Alexiev (Department of Mathematical Methods for Sensor Information Processing, Institute of Information and Communication Technologies, Sofia, Bulgaria)
PRACE facility:	JUGENE
Research field:	Computational neuroscience
Application code:	BrainCore

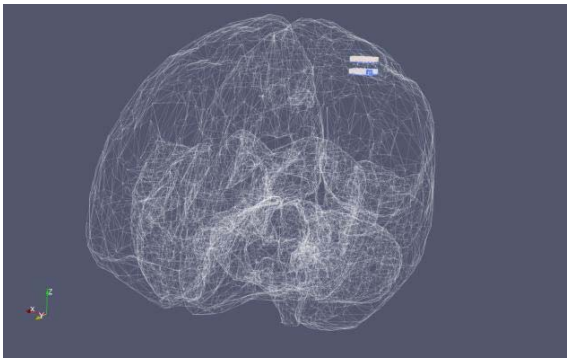
3.7.1 *Project objectives*

This project concerned the development of tools for visualization of output from brain simulations performed on supercomputers. The project had two main parts: 1) creating visualizations using large-scale simulation output from existing neural simulation codes, and 2) making extensions to some of the existing codes to allow interactive runtime (in-situ) visualization.

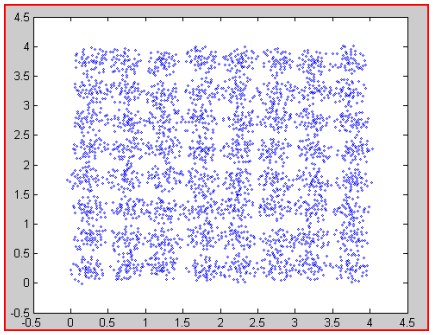
3.7.2 *Optimization work and results*

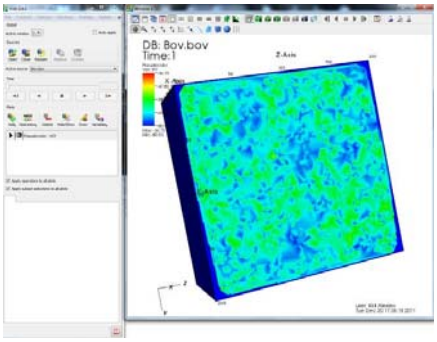
For the first part of the work, simulation data was converted to HDF5 format and split over multiple files. Visualization pipelines were created for different types of visualizations, e.g. voltage and calcium. A number of visualization examples is shown below:





For the second part of the work the VisIt visualization application and its *libsim* library was used. BrainCore simulation code was instrumented so that VisIt could access simulation data directly. The simulation code was instrumented and tested on different clusters where control of simulation was demonstrated and in-situ visualization of neural unit's and population data was achieved:





3.7.3 *Conclusions*

Considering the limited time available for this work quite good results were achieved, which will form the basis for further work in the future. We were able to develop a workflow for visualization of network activity of a brain region at both the single neuron and neuronal population levels, together with realtime visualization of simulated network activity. The open source program VisIt could be used for in-situ visualization and visualization of synthetic cell mesh activity. The tools developed could potentially be of use for researchers to visualize simulations by providing specific files and parameter settings as needed.

One important remaining issue for future work is to test the scalability of the visualization tools developed. The simulation model currently visualized has a relatively modest number of neurons, around 50,000 – 100,000, though during the course of this project we performed simulations with up to 57 million neurons connected by 7 billion synapses. Since the work started from scratch we developed the applications based on HPC enabled components, but time was not enough for extensive tests of scalability. Larger models will be used in the near future, having on the order of 100,000s neurons. For visualizing output from these models the visualization pipelines developed here can in principle be reused, but the larger scale will negatively influence the 3D rendering and data processing capabilities of ParaView. The VisIt package already allows visualization of large scale system. Its parallel scalability is excellent, especially in the case of multiprocessor/multicore usage and GPU Tesla.

3.8 High resolution ocean simulations with NEMO modeling system

Project leader:	Jean-Marc Molines, CNRS, LEGI, Grenoble, France
PRACE staff:	Nicole Audiffren, CINES, France
PRACE facility:	CURIE
Research field:	Earth Sciences and Environment
Application code:	NEMO

3.8.1 *Project objectives*

This project aims at preparing the high-resolution ocean/sea-ice realistic modeling environment implemented by the European DRAKKAR consortium for use on PRACE Tier-0 computers. Two original realistic model configurations (ORCA12 and PERIANT8_BIO) based on NEMO modeling framework are considered. They are designated to ensure the study of the role of multiple-scale interactions contributing to the ocean variability and the ocean carbon cycle and/or marine ecosystems exchanges.

ORCA12 configuration is a 1/12 degree realistic global configuration while PERIANT8 is a realistic regional 1/8 degree circumpolar Southern Ocean configuration including biogeochemistry. The former is designated with higher number of vertical levels than before (75 vs 46) while the second one has got 23 passive additional tracers which dramatically increase the computational burden. The folding at the north pole in the OCA12 configuration is a major difference between these two cases. The north pole folding requires an additional MPI communicator and dramatically complexes the communication scheme.

This project serves many purposes:

- porting the existed configurations from Jade (Tier-1) to CURIE
- fine tuning and assessment of the performances on CURIE
- checking the scalability of the configurations
- sensitivity experiments to the north pole folding.

3.8.2 *Optimization work and results*

For the ORCA12 configuration, the default distribution of tasks through the nodes (done by slurm and the scheduler) demonstrates very bad performance as shown in Figure 6. Binding of tasks was then done avoiding default placement and were based on openMPI options (round-robin distribution of MPI tasks by node and binding memory to socket). Additional use of the openIB RDMA protocols significantly ameliorates the overall performances leading to 60 steps per minute on 2560 cores. The overall gain is of 33% compared to the default placement.

For PERIANT8-BIO configuration extensive scalability tests were performed from 512 to 6144 cores. We use the standard ccc_mprun command from TGCC on CURIE. In Figure 7 we represent the performance rate as a function of the number of used cores. Blue squares correspond to the measured performance, red diamond indicates what can be expected with a perfect scalability. The first 2 points (512 ,1024 cores) almost fit the 'perfect' line. Then up to 3000 cores, the lost of scalability is less than 20%. At 3500 cores, we denote very different results depending on the choice of the domain decomposition: for about the same size of the domain, the‘shape’ of the domain seems to have a dramatic influence. Above 3500, there is a clear stagnation of performances, together with a great sensitivity to the domain decomposition. On the other hand, a test performed with 3072 cores on 192 fat nodes (unpopulated core computation) show a tremendous increase of performance (green triangle in Figure 7).

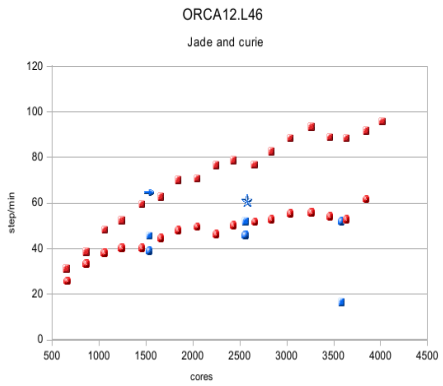


Figure 6: Scalability of ORCA12 configuration : comparison of results on Jade and CURIE. Red symbols correspond to JADE, blue symbols correspond to CURIE'. Circles are obtained with standard placement (provided by the ccc_mprun command). Squares are obtained with optimal placement (see above). The arrow show the performance without north pole folding condition. The star indicates the performance obtained with extra optimisation tuning (RDMA).

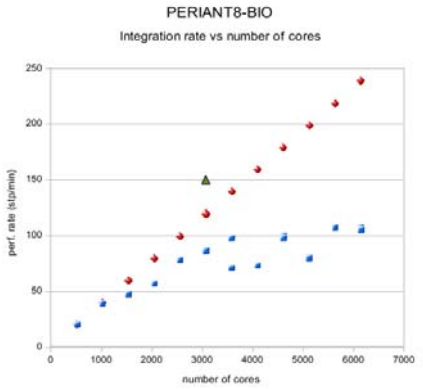


Figure 7: Scalability of PERIANT8-BIO with 23 passive tracers. The performance of the model is evaluated by the number of time-steps performed per elapsed minute. Blue squares correspond to the measured performance of the model when running in full core computation, red diamonds to the 'perfect' scalability with respect to 512 core experiment. Green triangle is a single test performed with unpopulated core computation (16 core/node).

3.8.3 *Conclusions*

Tests on PERIANT8_BIO indicates that high-resolution modeling coupled with biogeochemistry is feasible with almost the targeted performances. Tests on ORCA12 show a somewhat different behaviour from JADE current runs. The impact of the north pole folding condition has been demonstrated, and another group (headed by John Donners in PRACE) implements a new technique for this north pole folding condition. Tests on ORCA12 also demonstrated that the code will be more efficient on CURIE thin nodes for global simulation.

NEMO, in idealized configuration scales very well. For realistic configurations the scalability is degraded: For PERIANT8-BIO, a regular scalability is observed up to 2500- 3000 cores. For ORCA12, the scalability is poor in the tested range. We can infer reasons explaining these poor results (Majo and Gross 2011) and we might expect much better results on CURIE thin node. We need access to CURIE thin nodes in order to confirm the expected behaviour, with Donners's north pole folding condition and new I/O server.

3.9 Direct numerical simulation and turbulence modelling for fluid-structure interaction in aerodynamics

Project leader:	Marianna Braza, IMFT, Toulouse (France)
PRACE staff:	Eric Boyer, CINES, Montpellier (France)
	Hilde Ouvrard, CINES, Montpellier (France)
	Tyra Van Olmen, CINES, Montpellier (France)
PRACE facility:	CURIE
Research field:	Engineering and Energy (CFD)
Application code:	NSMB

3.9.1 *Project objectives*

The present proposal focuses on code development with support of experts concerning multi-level optimization of an efficient and robust CFD solver in aeronautics: NSMB, Navier-Stokes MultiBlock. This solver, used during the decade of 1990's and 2000's by principal European research institutes and European industry in aeronautics, has been significantly upgraded in the last ten years in terms of implementing new flow physics modelling, able to capture complex phenomena arising in aeronautics and aeroelasticity, by the collaborative efforts of CFS-Engineering, IMFT, IMF-Strasbourg, EPFL. A specific advanced version of the code containing Turbulence Modelling Approached developed by IMFT will be the object of the present optimization. The present project aims at improving the performances of this version of the code in order to use efficiently the largest number of processors currently available. This will allow investigating new physics in the turbulence processes arising in aeronautics, thanks to an increase by orders of magnitude the number of degrees of freedom and by refining the time-steps. Fully implicit numerical schemes will be privileged, dealing properly with the non-linearity of Turbulence in the flow equations. Furthermore, new methods of turbulence modelling and of coupling with deformable solid structures will be implemented in an optimal basis of MPI architecture, to increase the predictive capability of the solver comparing to other currently existing methods, concerning the unsteady aerodynamic loads, the nuisance instabilities as buffet and flutter, as well as strong flow detachments. These constitute currently the stiffest problems in aeronautics, fluid-structure interaction and flow control.

3.9.2 *Optimization work and results*

The work focused on establishing a state of efficiency of the parallelization of NSMB for a medium size problem. The goal is to determine the performance and to locate the steps blocking or penalizing the parallelization. NSMB is a solver of the Navier-Stokes equations (structured volumes, multi-blocks). The size of the problem is fixed (10 million finite volumes) and the study is focused on the influence of the number of blocks, the number of processors but also the distribution of blocks on the available nodes. Indeed, the sizes of blocks and the connections should be considered for an optimal distribution of works. Three techniques were used to define this distribution: a distribution based on the block size (for uniform distribution of the total size of finite volume per node), a distribution based on the SCOTCH algorithm taking into account the communication between blocks and the size (it minimizes the connections while keeping a uniform distribution of finite volumes per node) and the METIS tree on the same principle.

The problem chosen is the simulation of the flow around two cylinders in tandem configuration for a high Reynolds number flow (Re = 166 000), see figure below, as well as the transonic buffet over a supercritical airfoil, the OAT15A having 8,5 M points. The

speedup and efficiency of NSMB on the cylinders test case have been studied up to 4096 cores. The performanceis satisfactory (efficiency of 88% for 512 cores and 76% for 1024 cores). The computational domain has been decomposed into a significant number of sub-domains (called blocks) and computations on these blocks (up to 4096 blocks) have been performed using different loadbalancing strategies. Validating the different strategies on this test case for a significant number of blocks represents a crucial step to enable simulations on several thousand of cores on higher size problem. The METIS tree algorithm has proven to be a very efficient strategy. We noticed that the use of SCOTCH has to be modified to better share the work between processes and to fully take advantage of this strategy.

In order to benefit from the fat nodes part of CURIE: considerable number of cores per node (32 cores per node) and available memory (128 GB per node), it has been considered to work on a hybrid version of NSMB (OpenMP/MPI). This hybridization task asks for a huge cost in term of human resources and could not be fully accomplished within the 6 month period of access to CURIE. The preparatory access enabled us to better examine the chosen resolution algorithm and the parallelization strategy of NSMB. This is essential to help us to determine judicious locations into the resolution algorithm where OpenMP directives have to be added. It enabled us to realize some important tests regarding this specific fat node architecture. We actually did not have access to architecture with so many cores per node.

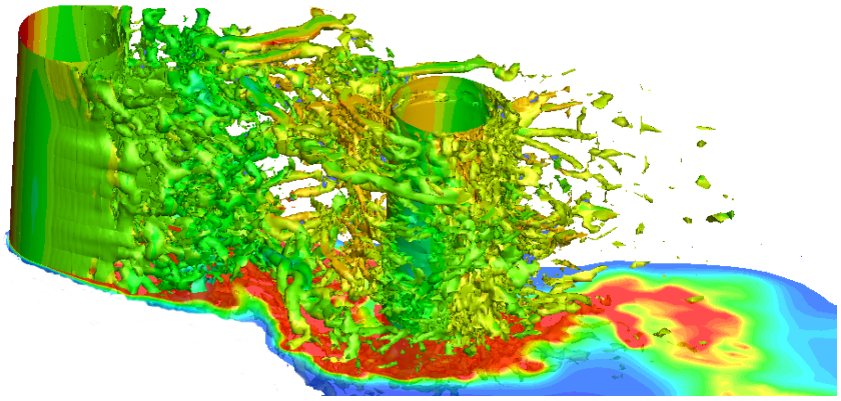


Figure 8: Numerical simulation of the tandem cylinders configuration of a landing gear at Reynolds number 160,000. Illustration of the complex eddies structure responsible for the acoustic noise.

3.9.3 *Conclusions*

The test of the parallelization of NSMB on a real medium problem has allowed determining its scalability (relative to a moderate mesh size, given the number of CPU hours available: 250,000). The present work shows the importance of the distribution of tasks per node. The choices based on METIS or SCOTCH show better performance when opportunities permit (number of blocks exceeds the number of cores). These tests have shown also the importance of multi-blocks cutting. It may be considered the involvement of the METIS/SCOTCH algorithms in cutting himself. The second track to increase the parallelization of the code is the use of direct solver library (like PSBLAS, MUMPS or SuperLU) on the whole problem.

3.10 Parallel Uniform Mesh Subdivision in Alya

Project leader:	Guillaume Houzeaux (BSC)
PRACE staff:	G. Houzeaux (BSC), R. de la Cruz (BSC), M. Vazquez (BSC)
PRACE facility:	JUGENE
Research field:	Engineering and Energy (CFD)
Application code:	Alya

3.10.1 *Project objectives*

Mesh generation in engineering applications is often the bottleneck of the complete simulation process. The mesh is the basis of the discretization algorithm and the first ”lego” of a simulation. A mesh should approximate well the necessary geometrical elements of the computational domain. In addition, it should be fine enough to capture the relevant physical scales of the engineering problem. Usually, commercial mesh generators (like ANSYS, etc.) do well with the first task. They include refinement tools like boundary layer elements and local adaptivity. However, it is quite difficult to generate very large meshes (say of the order of thousands of millions of elements) with the available tools. The problem is two fold. On the one hand, sufficient memory and time are needed on the computer where the mesh is generated. On the other hand, the size of the mesh can render very difficult the manipulation of the mesh files and the partition of it. Indeed, the only solution available today for partitioning the mesh is to have at least one node with sufficient RAM to read it.

The idea of this project is to implement a recursive and parallel uniform mesh multiplication in a HPC code developed at Barcelona Supercomputing Center named Alya.

- We assume that a first mesh (referred to as 0-level mesh) has been obtained with a mesh generator and that this mesh has been generated with an a-priori knowledge of the problem: for example with boundary layer elements near the wall (for CFD). For typical engineering applications, this mesh has between 10M to 100M of elements.
- Alya is based on a master-slave strategy. The master reads the 0-level mesh, partition it, and sends the subdomain meshes to the slaves. After this task, the original mesh does no longer exist, neither any array defined on it. The slaves now subdivide automatically their meshes and reconstruct their boundaries with neighboring subdomains. This mesh multiplication can be used recursively to obtain any number of level refinement.

The advantage of this strategy is that the mesh multiplication (MM), also refered to as mesh subdivision, is carried out in parallel. For example, if we start with a 30M tetrahedra element mesh, we obtain after 2 consecutive subdivisions a global mesh of 30Mx8x8 = 1.92 Billion element mesh in few seconds. Another advantage is that, if the 0-level is conserved, then the postprocess can be carried out on this mesh.

3.10.2 *Optimization work and results*

Some speedup tests have been carried out on two different architectures, namely MareNostrum and JUGENE supercomputers. The mesh multiplication has been used recursively on two different geometries. We denote here MM(i) the i-th mesh multiplication level. Results are shown in Figure 9. Near the CPU time table we added a reference time for the sake of comparisons. It is the time taken by METIS to partition the mesh on MareNostrum (denoted MN). In the case of Marenostrum,

we started from the mesh of a brain composed of 19M P1/P1 elements and could multiply the mesh up to 9.7B elements. The CPU time is comparable to that of METIS and the speedup is more than linear. From the detailed elpased time spent during the different tasks of the MM (not shown here) we observe that the task responsuble for the superlinear speedup is the node

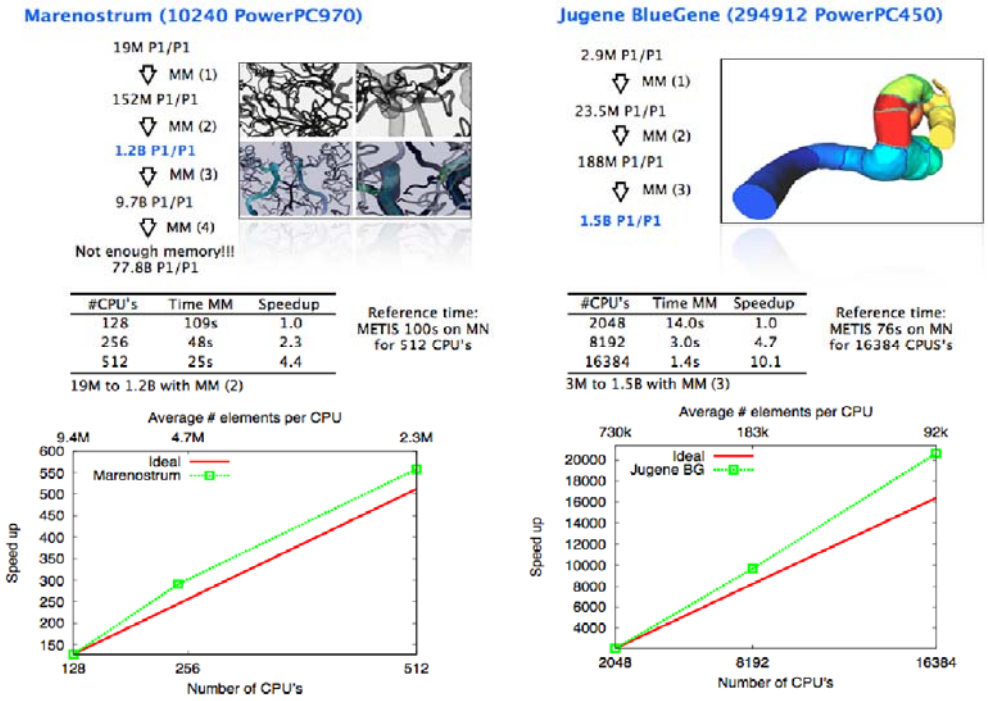


Figure 9: Time and speedup for the mesh mulpiplication

renumbering (METIS function METIS NodeND) and the associated nodal array permutations.

We also treated the geometry of an aneurysm composed of 2.9M elements. We multiplied three times the mesh in order to obtain 1.5B elements. The simulation was carried out on JUGENE using up to 16384 CPUs. When using 16384 CPUs, the time for the MM(3) is only 1s, compared to a METIS time of 76s. The speedup is also more than linear. The superlinear behavior is attributed to the node renumbering startegy applied at each mesh multiplicaiton level.

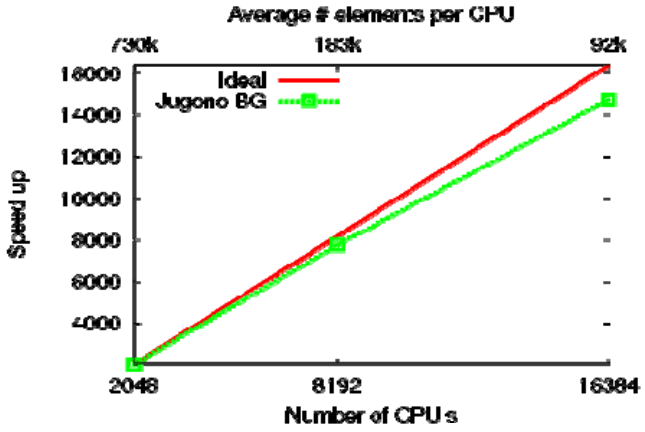


Figure 10: Speedup of the Navier-Stokes equations

Another interesting feature is the speedup of the code when solving the Navier-Stokes equations after using the MM. To to this we used the aneurysm mesh with MM(3). The difficulty is that when partitioning this mesh into 16384 subdomains, we obtain very few elements on each slave, 177 in average. Figure 10 hows that the resulting speedup is 90%.

3.10.3 *Conclusions*

We have developed a recursive, parallel and uniform mesh multiplication algorithm. The technique has been tested successfully using up to 16384 CPUs and to generate up to 10 Billion element meshes. In the near future, some improvements are going to be considered. For example, in the present work we do not correct the surface as interpolation is linear from one level to another. Some techniques like the one presented in will be implemented. Another issue is to take advantage of the mesh hierarchy to generate multigrid solvers.

3.11 Shocks: Understanding Relativistic Plasmas Acceleration Systems

Project leader:	Dr Gareth Murphy, Dublin Institute for Advanced Studies
PRACE staff:	Dr Michael Browne, Dr Gilles Civario, ICHEC
PRACE facility:	JUGENE and CURIE
Research field:	Astrophysics
Application code:	PSC code

3.11.1 *Project objectives*

Collision-less shock waves underlie many of the most spectacular and energetic events in the universe, such as supernova remnants, plasma jets, and gamma ray bursts. Although many plasma shocks have been observed in the universe, collision-less shock waves still hold many secrets such as how they are formed, how they amplify magnetic fields and how they accelerate particles.

The aim of this project, as part of PRACE- 1IP WP7.1, was to carry out test simulations to examine the scaling of the Particle In Cell (PIC)-based 'PSC' code for large numbers of CPU processors and to carry out development work to improve the codebase based on the outcomes.

The overall aim of the project is to allow the Project Leader to probe the complex 3D kinetic physics of shocks self-consistently, using simulations on Tier-0 architectures in Europe with the expectation that such simulations will also allow for a 'plasma shock database' to be produced in a later production centric project.

3.11.2 *Optimization work and results*

Several parallel scaling tests, for both strong and weak scaling, were carried out on the JUGENE and CURIE systems. Software analysis tools were used to analyze time spent in MPI communication versus time spent in computation. Using the *Scalasca* tool available on CURIE and JUGENE, it was found that optimising and rearranging parallel calls led to a speed up without affecting scientific output. It was found that proportionally more time was spent in global MPI communication during timing routines in the code. Investigations made by PRACE staff found that global communication could be reduced in frequency resulting in a speedup of 14.9% in the most computationally intensive loop of the code.

As part of this work the PSC code was shown to scale to 32,000 CPU cores on the JUGENE system. Experience in testing and identifying bottlenecks in the parallel algorithms was gained by the Project Leader throughout the project. For example, it was found that the 'virtual node mode' on JUGENE resulted in the fastest and largest scale simulations. This was in line with expectations, as the Particle-In-Cell (PIC) algorithm is CPU-bound and has good scaling properties. The 14.9% speedup, gained as a result of reducing the frequency of global communication, was maintained for high core counts, resulting in near-linear strong scaling (see Figure 11). This work demonstrated that the PSC code is comparable in performance on JUGENE with other leading PIC codes (e.g the OSIRIS consortium).

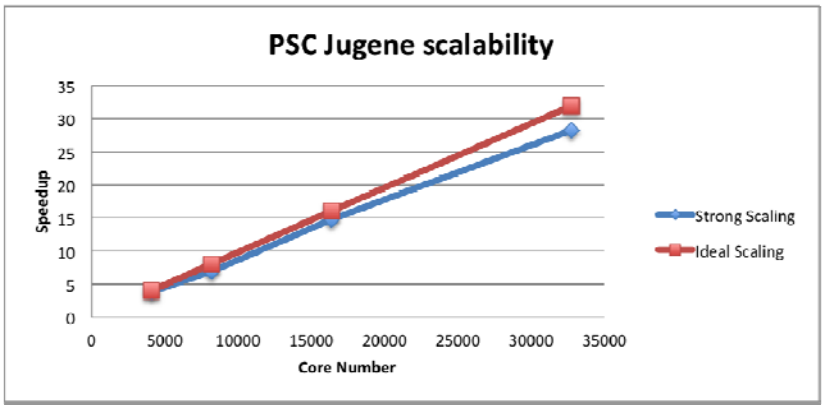


Figure 11 Strong scaling of the PSC code on JUGENE

Testing of MPI-I/O on petascale systems, which replaced the previous system of individual files per core, was carried out.

3.11.3 *Conclusions*

We have carried out a performance analysis of the PSC code on European Tier-0 architectures for the first time. We have been able to carry out this analysis with the aid of the Scalasca tool available on CURIE. We have also performed a scaling investigation of the PSC code on the JUGENE system where tests on up to 32,000 cores on JUGENE have shown linear (strong) scaling in agreement with results found for other PIC codes with similar requirements (OSIRIS).

The Project Leader would like to point out that he found the Irish national HPC infrastructure available through ICHEC very useful during the project, to use as a back up and for testing purposes before deploying to the large-scale systems.

3.12 Computational seismic wave propagation in highly heterogeneous volcanoes

Project leader:	Dr Gareth O'Brien, University College Dublin
PRACE staff:	Mr Ivan Girotto, Irish Centre for High End Computing (ICHEC)
PRACE facility:	JUGENE and CURIE
Research field:	Earth Sciences and Environment
Application code:	3D elastic lattice method

3.12.1 *Project objectives*

The overall aim of this project is to harness the power of Tier-0 systems available in Europe in order to simulate seismic wave propagation in structurally realistic volcano models. The code used in this project is written in the C language and makes use of MPI for inter-process communication. Forces, positions and velocities are distributed over a 3D grid of MPI processes based on a MPI Cartesian group distribution. The method is ideally suited to Massively Parallel Processor (MPP) architectures. The aim of this preparatory access project, as part of PRACE-1IP WP7.1 was to carry out test simulations to examine the scaling of the code for large numbers of CPU processors and to carry out development work to improve the codebase based on the outcomes. The project represented the first time that the project leader was able to test the code on a large number of CPU cores.

3.12.2 *Optimization work and results*

Several steps were made with regard to optimization of the code as part of this project which are listed below:

Introduction of an OpenMP layer (hybrid MPI+OpenMP implementation):

Initial attempts at a light-touch implemention of an OpenMP layer within the MPI-based code resulted in no further improvement in terms of speedup or efficiency, due mainly to the requirement of atomic operations within OpenMP loops which resulted in significant performance degradation. Avoiding atomic operations would have complicated the existing code considerably and would also have gone beyond the remit of the current project. For these reasons the hybrid model was decided against at an early stage of the project.

I/O profiling:

The code was instrumented to deliver a detailed analysis of where time was being spent. The bottleneck in speedup occurs for 8192 cores (see table 1 below) due to an increase in time spent in I/O operations. This is mainly due to the surface dump when increasing the number of MPI processes. For large core counts, parallel I/O manages a small amount of data and so it was analyzed how this could be easily managed using buffered I/O to implement a non-blocking strategy.

Results:

Two large problem sizes were analyzed during this project, characterized by data sets of sizes 2048^3 and 4096^3, respectively. All the tests were considered for calculations that included 1000 time iterations.

Table 1 shows the strong scaling performance of the code for the 2048^3 problem size. For a count of 8192 CPU cores we found that the code achieves 0.77% of efficiency based on the performance results for 512 CPU cores.

Table 2 shows the weak scaling performance of the code for two separate grid layouts. For both data layouts good weak-scaling of the code is evident. A larger number of CPU cores was not available at the time of the project due to queue restrictions while running on a smaller number of CPU cores was not possible due to the demands on memory for the problem sizes investigated.

Core count	Wall clock time	Speed-up (based on 512 cores)	Node count	Process count
512	12600	1.0	16	512
1024	6410	2.0	32	1024
2048	3320	3.8	64	2048
4096	1770	7.1	128	4096
8192	1020	12.4	256	8192

Table 1: Strong scaling results (grid size = 2048^3)

Problem size	Core count	Wall clock time	Speed-up (based on 512 cores)	Node count	Process count
Block grid size for process 256x256x128					
2048^3	512	12500	1	16	512
4096^3	4096	14500	0.86	128	4096
Block grid size for process 128x256x128					
2048^3	1024	6320	1	32	1024
4096^3	8192	7310	0.86	256	8192

Table 2: Weak scaling results

3.12.3 *Conclusions*

A performance analysis of the code on Tier-0 systems has been obtained by the Project Leader and PRACE staff for the first time. The code was shown to scale excellently on up to 8192 CPU cores. Investigations into how the code might be further optimized were also made by PRACE staff.

4 Summary

The task 7.1 has worked towards its objectives in enabling applications for capability science. The task has participated in preparing the Preparatory Access calls as well as performed reviews for type C Preparatory Access proposals. The evaluation process as well as the process for assigning optimization work to PRACE partners has been defined.

The task has worked on 14 optimization projects, partly in collaboration with tasks 7.5 and 7.6. The results of 12 optimization projects are reported here, and selected projects have also produced white papers which are published in PRACE RI www-site [4]. The results of optimization work on different application codes are summarized in Table 3. Performance of the has been improved in all projects, and most application codes worked on in task 7.1 are now ready for production usage in PRACE Tier-0 systems.

Application code	Result of optimization work
TurboRVB	Ready for Tier-0
Ludvig	Ready for Tier-0
PLUTO	Improved performance
Vlasiator	Ready for Tier-0
Pencil	Ready for Tier-0
Octopus	Ready for Tier-0
BrainCore	Improved performance
NEMO	Ready for Tier-0
NSMB	Improved performance
Alya	Ready for Tier-0
PSC	Ready for Tier-0
3D elastic lattice method	Ready for Tier-0

Table 3: Summary of results of optimization projects

In summary, task 7.1 achieved its objectives. In particular, it has been demonstrated that the approach of focussed short-term support by PRACE experts is successful. It had the expected effect of enabling new applications for Tier-0 usage or of significantly enhancing their scalability. In future, support for type C Preparatory Access projects will be continued in the task 7.1 of PRACE 2IP project, utilizing the foundations laid out by the current work.