E-Infrastructures
H2020- INFRAEDI-2018-2020

INFRAEDI-01-2018: Pan-European High Performance Computing infrastructure and services (PRACE)

PRACE-6IP

PRACE Sixth Implementation Phase Project

Grant Agreement Number: INFRAEDI-823767

D8.2

Interim Progress Report - Staff and Established Project Structure

Final

Version:	1.2
Author(s):	Fabio Affinito, Joost VandeVondele
Date:	18.10.2019

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: INFRAEDI-823767**	
	Project Title: PRACE Sixth Implementation Phase Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: < **D8.2**>	
	Deliverable Nature: <Report>	
	Dissemination Level: PU	**Contractual Date of Delivery:** 31/10/2019
		Actual Date of Delivery: 31/10/2019
	EC Project Officer: Leonardo Flores Añover	

Document Control Sheet

Document	**Title: Interim Progress Report - Staff and Established Project Structure**	
	ID: D8.2	
	Version: 1.2	**Status:** ***Final***
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2016	
	File(s): D8.2	
Authorship	**Written by:**	Fabio Affinito, JoostVandeVondele
	Contributors:	Martti Louhivuori, Mark Bull, Paul Gibbon, Berk Hess, Mark Abraham, Costantia Alexandrou, Mauro Bianco, Raffaele Solcà, Michal Merta, Alex Upton, Fabio Affinito, Joost VandeVondele
	Reviewed by:	Florian Berberich, JUELICH Aris Sotiropoulos, GRNET
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	24/09/2019	1st Draft	Missing conclusions
0.2	30/09/2019	2nd Draft	Introduction and conclusions reviewed and revised by JVV and FA, updated and added references
1.0	01/10/2019	Final version	Final version also reviewed by software project PIs
1.1	17/10/19	Revised version	Version revised after first internal review and comments from Mauro Bianco and Jakob Finkelrath
1.2	18/10/19	Final version	Added items to the abbreviations list

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure, Exascale, Forward-looking software solutions

Table of Contents

List of Figures

List of Tables

References and Applicable Documents

[1] Augonnet, C., Thibault, S., Namyst, R. & Wacrenier, P.-A. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurr. Comput. Pract. Exp.* **23**, 187–198 (2011).

[2] http://www.icl.utk.edu/files/publications/2017/icl-utk-1037-2017.pdf

[3] C. Alexandrou, S. Bacchio, J. Finkenrath, arXiv:1805.09584 [hep-lat]

[4] C. Alexandrou *et al.*, arXiv:1807.00495 [hep-lat]

[5] O. Coulaud, L. Giraud, P. Ramet, X. Vasseur, Developments in Parallel – Distributed – Grid and Cloud Computing for Engineering, Saxe-Coburg Publications, pp.249–275, 2013, 978-1-874672-62-3

[6] V. Fraysse, L. Giraud, S. Gratton, ACM Trans. Math. Softw. pp. 1-12, 35 (2), 2008

[7] L. Giraud, S. Gratton, X. Pinel, X. Vasseur, SIAM Journal on Scientific Computing, Society for Industrial and Applied Mathematics, 2010, 32 (4), pp.1858–1878

[8] http://www.netlib.org/blas/blast-forum/chapter3.pdf

List of Acronyms and Abbreviations

aisbl	Association International Sans But Lucratif (legal form of the PRACE-RI)
AMR	Adaptive-mesh refinement
BETI	Boundary element tearing and interconnecting
CoE	Center of Excellence
CPU	Central Processing Unit
CHASE	Chebyshev Accelerated Subspace iteration eigensolver
CUDA	Compute Unified Device Architecture (NVIDIA)
DCCRG	Distributed Cartesian cell refinable grid
DoA	Description of Action (formerly known as DoW)
EC	European Commission
EuroHPC	European High-Performance Computing Joint Undertaking
FETI	Finite element tearing and interconnecting
FMM	Fast-multipole method
GASPI	Global Address Space Programming Interface
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
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
GPU	Graphic Processing Unit
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
HPL	High Performance LINPACK
KB	Kilo (= 2^{10} ~10^3) Bytes (= 8 bits), also KByte
LINPACK	Software library for Linear Algebra
MB	Management Board (highest decision making body of the project)
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
MoU	Memorandum of Understanding.
MPI	Message Passing Interface

NIH	US National Institutes of Health
PFC	Plasma-facing component
PIC	Particle-in-cell
PM	Person-month
PRACE	Partnership for Advanced Computing in Europe; Project Acronym
QCD	Quantum chromodynamics
RI	Research Infrastructure
SIMD	Single instruction multiple data
SOL	Scrape-off layer
SPMD	Single program multiple data
SSC	Scientific Steering Committee
SVD	Singular value deccomposition
TAMPI	Task-aware MPI
TAGASPI	Task-aware GASPI
TB	Tera (= $2^{40} \sim 10^{12}$) Bytes (= 8 bits), also TByte
TFlop/s	Tera (= 10^{12}) Floating-point operations (usually in 64-bit, i.e. DP) per second, also TF/s
Tier-0	Denotes the apex of a conceptual pyramid of HPC systems. In this context the Supercomputing Research Infrastructure would host the Tier-0 systems; national or topical HPC centres would constitute Tier-1

List of Project Partner Acronyms

BADW-LRZ	Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Germany (3rd Party to GCS)
BILKENT	Bilkent University, Turkey (3rd Party to UHEM)
BSC	Barcelona Supercomputing Center - Centro Nacional de Supercomputacion, Spain
CaSToRC	The Computation-based Science and Technology Research Center (CaSToRC), The Cyprus Institute, Cyprus
CCSAS	Computing Centre of the Slovak Academy of Sciences, Slovakia
CEA	Commissariat à l'Energie Atomique et aux Energies Alternatives, France (3rd Party to GENCI)
CENAERO	Centre de Recherche en Aéronautique ASBL, Belgium (3rd Party to UANTWERPEN)
CESGA	Fundacion Publica Gallega Centro Tecnológico de Supercomputación de Galicia, Spain, (3rd Party to BSC)

CINECA	CINECA Consorzio Interuniversitario, Italy
CINES	Centre Informatique National de l'Enseignement Supérieur, France (3 rd Party to GENCI)
CNRS	Centre National de la Recherche Scientifique, France (3 rd Party to GENCI)
CSC	CSC Scientific Computing Ltd., Finland
CSIC	Spanish Council for Scientific Research (3rd Party to BSC)
CYFRONET	Academic Computing Centre CYFRONET AGH, Poland (3rd Party to PNSC)
DTU	Technical University of Denmark (3rd Party of UCPH)
EPCC	EPCC at The University of Edinburgh, UK
EUDAT	EUDAT OY
ETH Zurich (CSCS)	Eidgenössische Technische Hochschule Zürich – CSCS, Switzerland
GCS	Gauss Centre for Supercomputing e.V., Germany
GÉANT	GÉANT Vereniging
GENCI	Grand Equipement National de Calcul Intensiv, France
GRNET	Greek Research and Technology Network S.A., Greece
ICREA	Catalan Institution for Research and Advanced Studies (3rd Party to BSC)
INRIA	Institut National de Recherche en Informatique et Automatique, France (3rd Party to GENCI)
IST-ID	Instituto Superior Técnico for Research and Development, Portugal (3rd Party to UC-LCA)
IT4I	Vysoka Skola Banska - Technicka Univerzita Ostrava, Czech Republic
IUCC	Machba - Inter University Computation Centre, Israel
JUELICH	Forschungszentrum Juelich GmbH, Germany
KIFÜ (NIIFI)	Governmental Information Technology Development Agency, Hungary
KTH	Royal Institute of Technology, Sweden (3rd Party to SNIC-UU)
KULEUVEN	Katholieke Universiteit Leuven, Belgium (3rd Party to UANTWERPEN)
LiU	Linkoping University, Sweden (3rd Party to SNIC-UU)
MPCDF	Max Planck Gesellschaft zur Förderung der Wissenschaften e.V., Germany (3 rd Party to GCS)
NCSA	NATIONAL CENTRE FOR SUPERCOMPUTING APPLICATIONS, Bulgaria
NTNU	The Norwegian University of Science and Technology, Norway (3rd Party to SIGMA2)
NUI-Galway	National University of Ireland Galway, Ireland

PRACE	Partnership for Advanced Computing in Europe aisbl, Belgium
PSNC	Poznan Supercomputing and Networking Center, Poland
SDU	University of Southern Denmark (3rd Party to UCPH)
SIGMA2	UNINETT Sigma2 AS, Norway
SNIC-UU	Uppsala Universitet, Sweden
STFC	Science and Technology Facilities Council, UK (3rd Party to UEDIN)
SURFsara	Dutch national high-performance computing and e-Science support center, part of the SURF cooperative, Netherlands
TASK	Politechnika Gdańska (3rd Party to PNSC)
TU Wien	Technische Universität Wien, Austria
UANTWERPEN	Universiteit Antwerpen, Belgium
UC-LCA	Universidade de Coimbra, Labotatório de Computação Avançada, Portugal
UCPH	Københavns Universitet, Denmark
UEDIN	The University of Edinburgh
UHEM	Istanbul Technical University, Ayazaga Campus, Turkey
UIBK	Universität Innsbruck, Austria (3rd Party to TU Wien)
UiO	University of Oslo, Norway (3rd Party to SIGMA2)
UL	UNIVERZA V LJUBLJANI, Slovenia
ULIEGE	Université de Liège; Belgium (3rd Party to UANTWERPEN)
U Luxembourg	University of Luxembourg
UM	Universidade do Minho, Portugal, (3rd Party to UC-LCA)
UmU	Umea University, Sweden (3rd Party to SNIC-UU)
UnivEvora	Universidade de Évora, Portugal (3rd Party to UC-LCA)
UnivPorto	Universidade do Porto, Portugal (3rd Party to UC-LCA)
UPC	Universitat Politècnica de Catalunya, Spain (3rd Party to BSC)
USTUTT-HLRS	Universitaet Stuttgart – HLRS, Germany (3rd Party to GCS)
WCSS	Politechnika Wroclawska, Poland (3rd Party to PNSC)

Executive Summary

This document gives a brief overview of the eight projects funded in PRACE-6IP Work Package (WP) 8. The previous deliverable, D8.1, described the selection process. Therefore, in this deliverable we give an overview of how the project teams are starting their work, with a focus on their composition, organisation, and planned work. In order to reach the targets of the proposals, each team has been given freedom with regards to their internal organisation and choice of most appropriate software development methodology for their project. In this document each team will describe their chosen approach and the results achieved in the first stage of the project, with an explicit schedule, or an outlook of the planned work in the next six months.

1 Introduction

The objective of WP8 is to face the challenges of the Exascale transition from the point of view of the software solutions. This means that the fundamental approach consists in delivering, at the end of the PRACE 6IP project, "forward-looking" solutions that will deal with two main issues: 1) the diversity of the incoming new hardware infrastructure and 2) the complexity of the codes in use by the scientific communities. The high difficulties related to those two aspects does not permit to face this problem in other ways than with a disruptive approach consisting in the development of new software (or in some cases of deeply refactored existing software) or in domain-specific libraries. This activity requires a high investment of resources from PRACE and the involved partners. However, the approach "high-risk, high-gain" is necessary because the impact required must be qualitatively high enough to respond to the Exascale challenges. These projects are furthermore timely as it aligns with the EuroHPC investments in pre-Exascale hardware.

In order to ensure a high qualitative standard for this objective, PRACE decided to go through a competitive selection, which started earlier than the PRACE-6IP project. The requirement for this selection was primarily bound to a strong commitment of resources of the partners willing to take part, in terms of manpower and of expertise.

The competitive process started with 14 Letter of Intents and after a selection which deeply involved the PRACE SSC was concluded with 8 approved projects. The selection process has been accurately described in the PRACE Deliverable D8.1. The approved projects (reported in Table 1) span different scientific and technological areas and involve many different PRACE partners. The committed manpower effort varies between approximately 60 and 180 person-month.

The PRACE Kick-off meeting, held in Bratislava, Slovakia in April 2019, gave us the first opportunity to gather and to discuss the organisation of this Work Package once the awarded projects were approved. During this meeting we discussed about the structure and the organisational model of the Work Package, concluding for an approach as light as possible, leaving the possibility to each of the project to decide its internal rules. This includes the communication channels among the participants to the projects, the choice of repositories and the software development methodologies to be adopted. An overall synchronisation of all the projects will be set only on specific cases when this is needed, for example for the submission of deliverables, organisation of workshop and conferences, etc. In addition to such "on-purpose" virtual meetings, we are planning face-to-face meetings and/or conferences focused on themes of interest to many different projects (for example programming models, tasking-based approaches, etc) and also attractive and fostering collaboration with communities external to PRACE. On the other hand, the kick-off participants highlighted the infrastructural need for computational resources and other tools that should be provided by PRACE.

Project title	Partners	Effort (PM)
PiCKeX - Particle kinetic codes for Exascale plasma simulations	JSC, UL	90
Performance portable linear algebra	ETHZ, UL, JSC, CINECA	182
MoPHA - Modernisation of plasma physics simulation codes for heterogeneous Exascale architectures	CSC, MPCDF, UL	135
LyNcs: Linear algebra, Krylov subspace methods and multigrid solvers for the discovery of new physics	CASTORC, INRIA, LRZ	120
LoSync - Synchronization reducing programming techniques and runtime support	EPCC, BSC	90
NB-LIB - Performance portable library for N-body force calculation at Exascale	KTH, ETHZ	105
GHEX - Performance portable communication layer for grid application	ETHZ, UiO	111
FEM-BEM based domain decomposition solver	IT4I	62

Table 1: Overview of WP8 funded projects

In this document we will go through all the approved projects, analysing their individual objectives and relative schedule. All the sections will describe the details of the staff taking part to the project and the role of each partner in the project. Also, each project reports about the choice made about their internal organisation (i.e. communication tools, etc.) and the adopted software development methodologies (for example Agile, Scrum, etc.). Finally, each section contains a short report of the initial results obtained so far and the planned work from the next six months.

2 PiCKeX- Particle Kinetic codes for Exascale plasma simulation

2.1 Introduction and objectives of the project

Numerical modelling of plasmas often demands a kinetic approach to handle extreme nonlinearities, wave-particle interactions and other non-Maxwellian phenomena. Mathematically this requires the *ab initio* solution of the relativistic Vlasov-Boltzmann equation for the plasma constituents together with the appropriate Maxwell equations for the electromagnetic fields. Currently the model of choice is the particle-in-cell (PIC) code (or equivalently gyrokinetics for magnetised plasmas) a highly versatile, robust, finite-difference discretization of the Vlasov equation for the particle distribution function f(x,p). State of the art three dimensional PIC simulations involve up to 10^{12} particles on 10^{6} cores.

Even when performed on modern supercomputers, PIC simulation still has its limitations: apart from the computational expense of solving the particle and field equations for the full 3D electromagnetic system, the necessity of transferring information to and from the spatial grid makes it inherently noisy, collisional regimes are only accessible with the help of ad-hoc extensions, and some form of adaptive mesh refinement is often required to handle geometrically complex problems. Various simplifications are possible by exploiting geometrical symmetry, enabling 1D and 2D codes to be used for preliminary studies or parameter scans. If the phenomena of interest evolve slowing in time, implicit algorithms can be used to overcome timestep limitations of explicit codes due to the Courant condition.

While these various classes of kinetic plasma model - from fully electromagnetic to simplified electrostatic or magnetoinductive - are sufficient for isolated physical contexts, the complexity of plasma device simulation and laser-plasma applications increasingly demands hybrid or multi-scale modelling involving two or more separate but closely coupled codes. Here it would clearly be advantageous to offer generic algorithmic components for kinetic plasma simulation on a single platform or within an easily portable library which are capable of exploiting modern supercomputing architectures. Clearly this goal is way beyond the scope of the present call, so to ensure that results are achieved with some impact, the present project will concentrate on two important kinetic codes, **EPOCH** and **BIT1**, which are heavily used in the laser-plasma and magnetic fusion (edge physics) communities respectively. The general aim here is to transfer HPC best practices and software technology known to work well in algorithmically related, but physically much simpler applications, in order to enable these two codes for Exascale-class simulations.

Major community codes involved

The **EPOCH** code already has a large user-base of over 800 registered users and has become central to many plasma physics research projects worldwide for studies of laser-plasma interactions, QED-plasmas, tokamak kinetic instabilities, space physics and particle accelerator design. It solves the full Maxwell equations for the electromagnetic field (offering several different numerical implementations of the field solver), with fully relativistic charge dynamics (supporting various integrator schemes). EPOCH utilises computers ranging from workstations in 1D, up to 100,000's of cores on national Tier-1/Tier-0 supercomputers in 3D, with a clear potential for performance optimisation to bring it to the Exascale HPC level needed for 3D simulations of complex plasma processes over disparate length-scales. To achieve this while maintaining the code's versatility, EPOCH's core parallelism, vectorization, I/O and other algorithm kernels will need to be adapted to make best use of contemporary and future Exascale machine architectures.

BIT1: Kinetic effects in the Scrape-off Layer (SOL) play an important role for the future confinement fusion devices: they affect strongly plasma and power loads to the plasma facing components (PFC). As a result kinetic effects in the SOL influence on the lifetime of the PFC. Therefore, kinetic study of the SOL has become one of the most challenging topics in fusion plasma research. The systematic kinetic study of SOL can be made using the unique PIC/MC code BIT1. The BIT1 is electrostatic parallel massively PIC code with MC (Monte Carlo) routines. The general PIC algorithm in BIT1 uses the motion of each plasma particle to calculate all macro-quantities (like density, current density and so on) from their positions and velocities. The macro-force acting on the particles is calculated from the field equations i.e. set of Maxwell's equations. Since BIT1 is electrostatic it solves only the Poisson equation in one spatial dimension, but retains 3 velocity components (1D3V). Collisions are reintroduced via a Monte-Carlo (MC) based collision operator. The BIT1 code algorithm includes both the PIC and MC routines, with a workflow comprising an integrator to solve the equation of motion, a Poisson field-solver including the effects of the vessel walls, followed by the MC part to account for particle collisions and the characteristics of the plasma source and boundary effects.

2.2 Project schedule

The following Gantt chart shows the project timeline with the main tasks and subtasks:

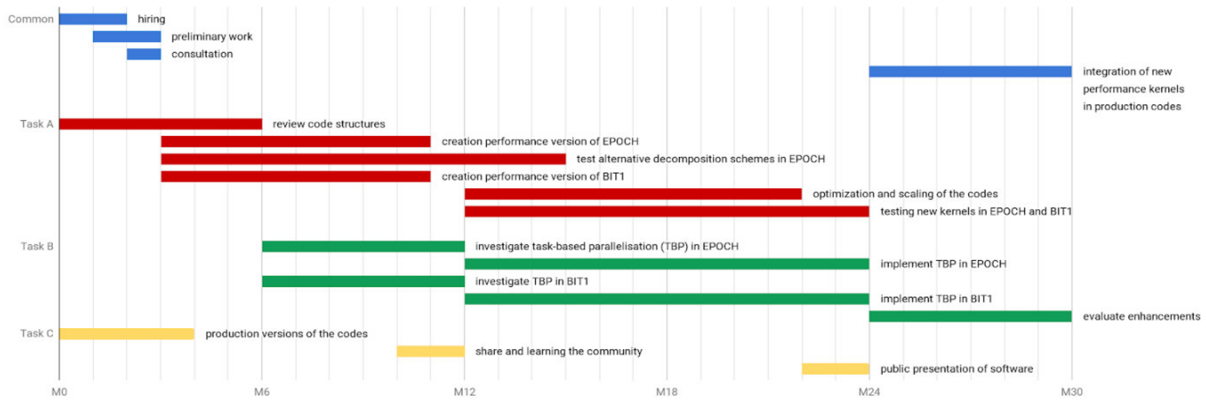


Figure 1: PiCKeX timeline (months) showing estimated start and end of tasks and sub-tasks

The project will basically comprise 4 phases with associated checkpoints:

M6: An initial consultation with the associated developer groups in Warwick and Vienna to obtain up-to-date information on the current and near-term development plans and benchmarking.

M12: Creation of 'performance' versions of each code, perhaps with reduced physics/model complexity, followed by verification and first optimisation actions.

M24: Completion of major refactoring including application of new programming models, with benchmarking of new kernels and algorithms. Incorporation of the enhanced code versions back into the production versions.

M30: Final report with performance enhancement results.

2.3 Team and collaboration

The team consists of two sites at the Juelich Supercomputing Centre (Juelich, Lead) and the University of Ljubljana (UL, Contributor). The staff members and their roles, together with the original code developers, are described below.

Team member	Position	Project role	Funding type
Paul Gibbon	Division head, JSC	PI	matching
Dirk Broemmel	Staff member, JSC	EPOCH analysis & redesign	matching
Philipp Otte	PRACE HLST, JSC	Performance analysis	matching
Ujjwal Sinha	Postdoc, JSC	EPOCH refactoring	PRACE-6IP (100%)
Zahra Chitgar	PhD student, JSC	EPOCH physics use cases	matching
Junxian Chew	PhD student, JSC	PEPC electrostatic solver	matching
Leon Kos	Head, LECAD, UL	co-PI	matching
Ivona Vasileska	PhD student, LECAD	BIT1 user and test case	matching
Gregor Simic	Technical colleague, LECAD	testing and deploying the software	matching
Matic Brank	PhD student, LECAD	BIT1 code refactoring schemes	matching
Tony Arber, Keith Bennett	University of Warwick	Developer team, EPOCH code	External in-kind
David Tskhakaya	Prague University	Lead developer, BIT1 code	External in-kind

Table 2: PiCKeX Staff members and roles

For collaboration the team uses, skype and video conference room for regular and ad-hoc meetings. The project has also a Bitbucket organisation that will be used to realise the test codes and their prototypes developed in the projects. Each of the codes have also their own code development repositories:

- **EPOCH** https://gitlab.version.fz-juelich.de/SLPP/epoch/epoch
- **BIT1** https://bitbucket.org/lecadpeg/simpic/

For the EPOCH code, the gitlab repository at JSC is effectively a clone of the main developer hub hosted at the University of Warwick, and will be synchronised such that both sides have access to the new performance version.

Meetings

- 28/29 May 2019: general discussions at PRACE-6IP F2F meeting, Bratislava
- 19 June 2019: project team VC between JSC and U Ljubljana to discuss platform sharing and initial code activities
- 4 September 2019: VC between JSC and EPOCH developer team at University of Warwick
- October/November 2019: F2F in Juelich

2.4 Status and outlook

The project has had a successful start, with a first performance assessment of the 2D version of EPOCH and preliminary refactoring work on BIT1. For the latter code, the team from the University of Ljubljana already start to work on the prototype code SIMPIC, and successfully got the compute time on EUROfusion A4 partition (D.A.V.I.D.E cluster at CINECA). The team now can work truly heterogeneous GPU/OpenMP/MPI machine (4 NVIDIA GPUs per node with Power 8 processors that have 16 cores with 8 threads per core totalling to 128 threaded cores per node), they can submit up to 32 nodes in one job (128 GPUs+4096 CPU threads).

The EPOCH code has been instrumented using Score-P for proper profiling and tracing. Three representative test cases were chosen: i) a Weibel instability as an example of an ideally balanced configuration; ii) laser-particle acceleration utilizing the moving window feature where only the last 50 iterations are measured and iii) an interaction of a laser striking a solid foil target where load balancing is difficult to maintain over long run-times. Results shown below consider the time marching loop only with I/O disabled. The main problems of EPOCH identified in the initial tests are: lack of adaptive load balancing (c.f. Figure 1), a naive and costly implementation of the moving window (approx. 30% of the runtime is spent on this feature) and a high ratio of CPU cycles stalled on resources.

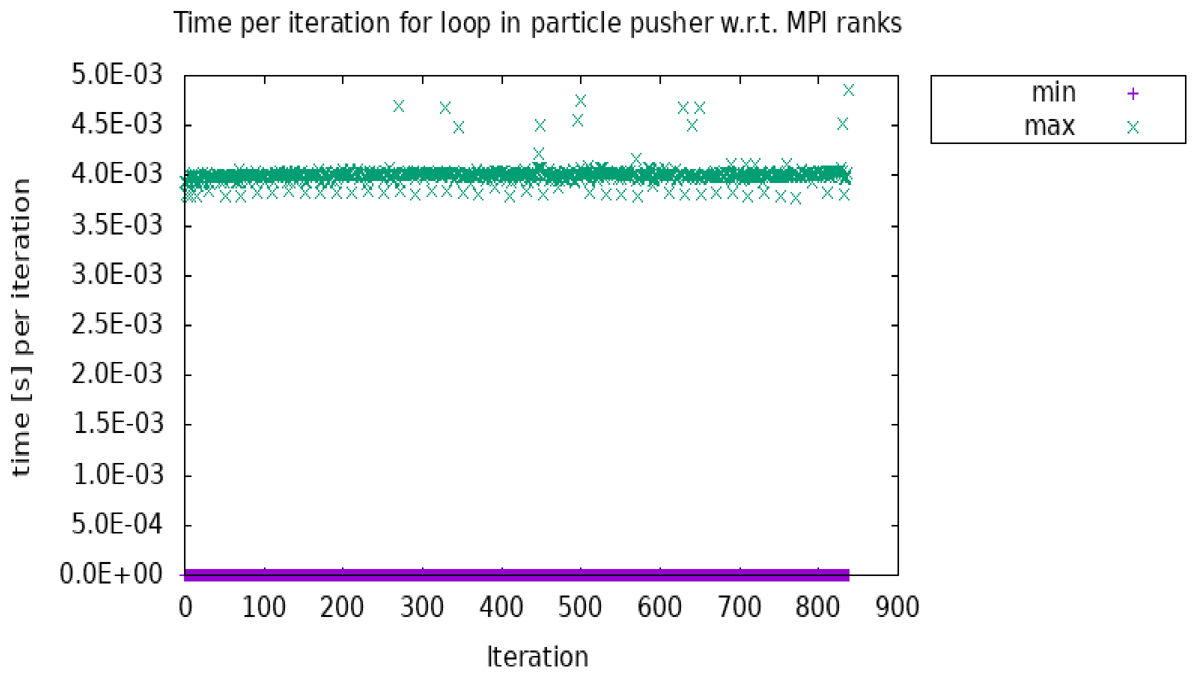


Figure 2: Evolution of time spent in particle pusher per iteration across MPI ranks, showing the emergence of load imbalance as the plasma system is disturbed by a powerful laser pulse

Based on this preliminary analysis, the following modification are considered: 1) improving load balancing by hybridization using OpenMP and implementation of an improved dynamic load balancer; 2) reimplementation of the moving window feature in order to remove the high cost of transporting data; 3) improvement of data reuse in order to drive down the ratio of stalled CPU cycles; 4) hybridization using MPI and OpenMP to further mitigate load imbalance.

Over the next 6 months first experimental versions of both codes will be established: BIT1->SIMPIC; EPOCH -> EPOCH-X along the lines indicated above.

3 MoPHA – Modernisation of Plasma Physics Simulation Codes for Heterogeneous Exascale Architectures

3.1 Introduction and objectives of the project

Numerical simulations are absolutely essential to address central open questions in plasma physics, from fusion energy to space weather. Of key importance is the understanding of the fundamental physical processes involved in plasma turbulence. Insight-providing simulations require enormous computational efforts. Prerequisite for adequate and efficient usability of next generation supercomputers in the pre-Exascale and Exascale era is to push respective codes in this field to the next structural level with respect to scalability and portability to heterogeneous HPC architectures.

In order to prepare for the technical and scientific challenges in Exascale computing, the project aims to improve the scalability and adaptability of three widely used plasma physics codes - ELMFIRE, GENE, and Vlasiator - which share the feature that they use higher-dimensional (3D to 6D) grids. The project consists of three main tasks: 1) *refactoring* of the codes to make them more accessible and adaptable, 2) implementing *task-based parallelisation* to achieve performance portability and scalability, and 3) *knowledge transfer* to encourage code reuse in the plasma physics community (as well as beyond) and to share lessons learned in the project.

Task A: Refactoring of the existing codes to have a clearer separation of concern is the key to achieving the adaptability and portability needed to fully utilise extreme-scale HPC systems. Code modernisation and modularisation will allow for more accessible and adaptable codes with a clear separation between different levels of the code. This separation of concern will make it easier to use external libraries and to fine-tune computational kernels to different architectures in order to achieve maximal performance. In light of the current trend of increasing diversity of architectures within a single high-end HPC system (CPUs with large vector registers, GPUs connected via PCIe or NVLINK, FPGAs, vector processors as accelerators etc.), it is clear that a modular design with clear separations is needed more than ever. In order to cater for specific programming models, special treatment of data or other architecture-dependent tweaks that are required to fully utilise the hardware, it is crucial to either limit the code changes needed to a small part of the code or to offload the burden to an external library.

One of the aims of this project is to modernise, refactor, and modularise three representative plasma physics codes to achieve a high level of separation between different levels of code. In addition, using an external library (MFEM, https://mfem.org/) for finite element discretisation will be investigated and new mesh implementations will be explored to address scalability issues due to the increased complexity of mesh metadata in the adaptive mesh refinement (AMR).

ELMFIRE

Full-torus simulations of electrostatic turbulence in tokamaks are extremely computationally intensive, and to take full advantage of the increasing computing resources available, the ELMFIRE code needs to be modernised and modularised. A more versatile version of ELMFIRE can be achieved by in-depth refactoring of the code, using modern programming models and modular writing with consistent conventions, testing and documentation.

The development of the refactored ELMFIRE will be tiered in order to deliver usable successively less-reduced versions of the final code during the course of the project. Using an external library (MFEM) for scalable and portable finite element discretisation will also be investigated.

GENE

The diversity of architectures in the HPC world, especially in the top-end systems, has made it increasingly difficult to run and maintain a single code base and still reach maximal performance with GENE on nearly all major HPC systems around the world. To address this, a refactoring of the GENE code is necessary to achieve a clear separation between different levels in the code.

On the highest level, an abstraction layer that describes the physical problem and differential equations, in the middle level a numerical representation of the equations and algorithms to solve them, and on the lowest level a number of computationally intensive kernels. The computationally intensive kernels, such as the right-hand terms in the Vlasov equations, can then be ported with the best possible approach to different architectures.

Vlasiator

The current implementation is based on the DCCRG (Distributed Cartesian Cell-Refinable Grid) library, which suffers from a rather centralised mesh metadata handling and related memory overheads. With the introduction of adaptive mesh refinement (AMR), the complexity of mesh metadata has grown considerably, and mesh implementations with less global communication and shared state will be essential. The aim is to refactor Vlasiator to become independent from its underlying grid framework, and portable to be based on existing state-of-the art mesh implementations such as AMReX.

In the process, the actual solver implementation and mesh structure will be logically separated, to allow for easier switching of the mesh back-end. This will enable and improve code reuse, portability and aid Vlasiator in utilizing the latest developments in this area.

Task B: Task-based parallelisation offers a natural approach for heterogeneous parallel computing, both in terms of algorithms and computing tasks but also in terms of diverse hardware environments. By splitting the computation into reasonably small tasks and by defining the dependency graph among the tasks, it is possible to utilise efficiently the full width of a single computational node, even in a heterogeneous architecture that may contain e.g. GPUs or other accelerators. Task-based parallelisation in combination with message-passing communication (MPI) for multi-node parallelisation is well suited to achieve good performance and scalability on extreme-scale computing resources containing a diversity of architectures.

One of the aims of this project is to explore and implement task-based parallelisation in the context of plasma physics codes. To this aim, task graphs will be designed for the computationally intensive solvers (e.g. for Vlasov and field equations). In addition to task-based parallelisation schemes, the runtime task scheduler StarPU [1] will be investigated. Using an external runtime task scheduler may make it easier to port functionality and performance between architectures without a need to have in-built support for new architectures in the plasma simulation codes.

In order to facilitate the exploration of different approaches to task-based parallelism and/or utilisation of GPUs, the project will implement simplified prototype codes (miniapps) that address a particular aspect of plasma simulations. These miniapps will then be used to test different approaches allowing for a more rapid exploration for potentially effective strategies. All insights gained with these tests will be valuable for finally introducing those approaches with the most potential into the target codes.

Task C: Knowledge transfer is an integral part of fostering good development practices, including code reuse and sharing of experience, in the plasma physics community. In general, the pooling of effort to develop external libraries dedicated to providing good solutions to common problems that are then used in multiple scientific codes is by far the best trend in recent years.

Any lessons learned in the project will be shared with the community with the aim that by encouraging the use of external libraries and code reuse, the overall standard of plasma physics codes will be improved. All miniapps developed in the project will also be made freely available in a common git repository.

3.2 Project schedule

The Gantt chart in Figures shows the project timeline with the main tasks and subtasks:

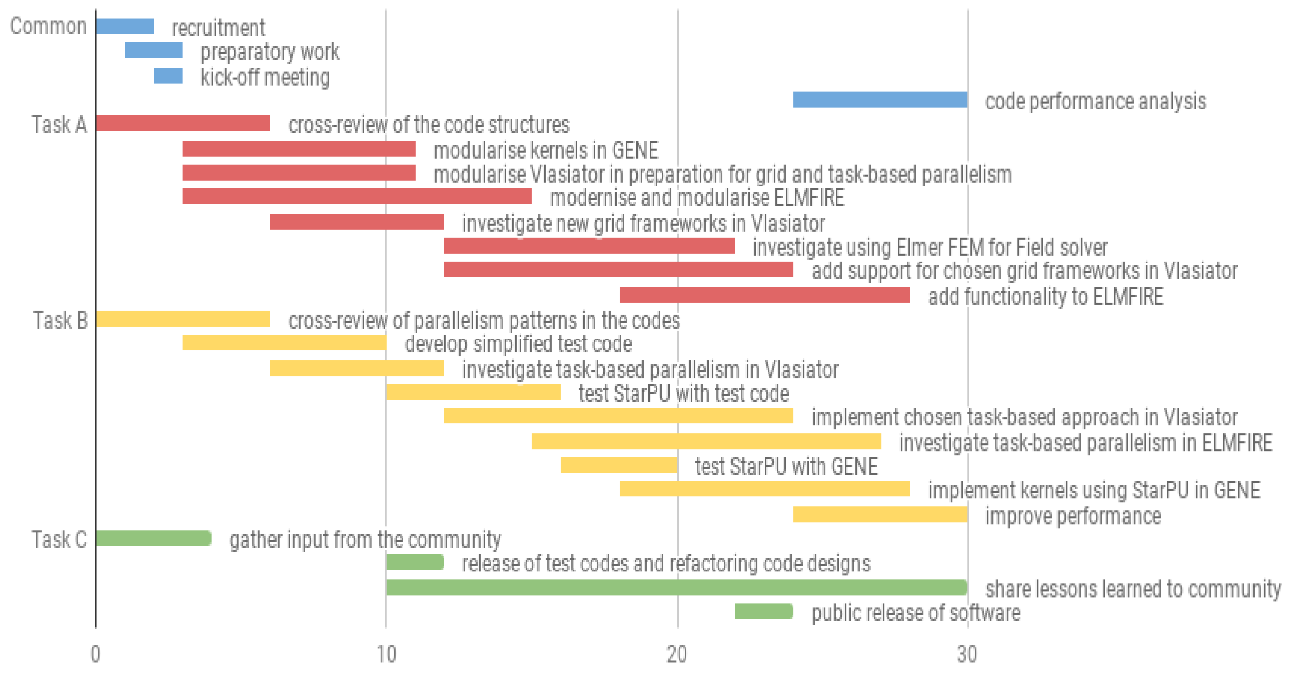


Figure 3: MoPHA timeline (months) showing estimated start and end of each sub-task

Milestone / Deliverable	Month	Description
D1	6	Contribution to WP8 1st interim progress report (staffing, project structure and initiation)
D2	12	Contribution to WP8 2nd interim progress report (pre-release of software)
D3	24	Contribution to WP8 3rd interim progress report (public release of software)

Milestone / Deliverable	Month	Description
D4	30	Contribution to WP8 final report
MS1	3	Kick-off meeting held and recruitment finished
MS2	6	Cross-review of the codes (ELMFIRE, GENE, Vlasiator) discussed
MS3	10	Task-based work-sharing and task graphs designed for the evaluation test code
MS4	12	Release of evaluation test codes and refactoring code designs
MS5	22	Separation of concern realised through refactoring of the codes
MS6	24	Public release of software
MS7	28	Task-based parallelisation implemented in a plasma physics code
MS8	30	Presentation of results in a suitable event (ISC, plasma physics conference etc.) including performance assessment

Table 3: MoPHA milestones and deliverables

3.3 Team and collaboration

The project has three main contributors: CSC, MPCDF, and University of Ljubljana, with additional in-kind contributions from the code development teams of ELMFIRE, GENE, and Vlasiator. The team consists of HPC experts and scientific code developers familiar with state-of-the-art HPC programming techniques and the codes targeted in the project.

CSC

Martti Louhivuori (*PI*), Sebastian von Alfthan (*co-PI*), Laurent Chôné (*lead contributor for ELMFIRE work*), Henrik Nortamo, Fredrik Robertsén

MPCDF

Tilman Dannert (*lead contributor for GENE work*), Carlos Lopez Cruz

University of Ljubljana

Leon Kos, Matic Brank, Dejan Penko, Gregor Simič, Ivona Vasileska

University of Helsinki

Urs Ganse (*lead contributor for Vlasiator work*)

The project is also supported by scientific advisors for each of the codes: Frank Jenko (Max Planck Institute of Plasma Physics, GENE), Timo Kiviniemi (Aalto University, ELMFIRE), and Minna Palmroth (University of Helsinki, Vlasiator).

For collaboration the team uses a Slack workspace, an always open Zoom video conference room (for regular and ad-hoc meetings), and a mailing list. The project has also a Github organisation that will be used to release the test codes and miniapp prototypes developed in the project. Each of the codes have also their own code repositories.

• MoPHA on Github:	github.com/MoPHA
• ELMFIRE:	elmfire.eu
• GENE:	genecode.org
• Vlasiator:	helsinki.fi/vlasiator github.com/fmihpc/vlasiator

Meetings

- May 27-28, Kick-off meeting, in Bratislava prior to the PRACE-6IP all-hands meeting
- Sep 9, 1st Video conference
- Oct 1, 2nd Video conference

3.4 Status and outlook

The project has had a successful start and the work is progressing as planned. A one-and-half day kick-off meeting was organised in conjunction with the PRACE-6IP all-hands meetings to discuss the technical details of the project and to make concrete plans for the work. In addition, the recruitment of project personnel has also been finished.

Started at the kick-off meeting and continued over the summer period, a cross review of the three codes was done to identify similarities and common structures in the codes. Based on some of the similarities identified, an initial set of prototype miniapps has been planned and work on implementing them has started.

Refactoring work on all three codes has also started. Additionally, a continuous integration (CI) workflow has been implemented for Vlasiator that will greatly improve productivity for the refactoring work. Work on designing the task graphs has also started with initial ideas on how to organise the code into tasks.

In the next six months, the project aims to continue the refactoring work by improving the modularisation of GENE and Vlasiator and by integrating the MFEM library with ELMFIRE. The planned prototype miniapps will also be implemented and task graphs will be designed for the evaluation test code. Preparations for the pre-release of the miniapps and refactoring code designs will be done as well.

4 NB-Lib: Performance portable library for N-body force calculations at the Exascale

4.1 Introduction and objectives of the project

A large number of scientific applications use particle interactions (e.g. Molecular Dynamics, Monte Carlo or multiscale simulations in life sciences or materials), and several smaller codes or combinations of codes have unique features. However, as computers have become more specialised, many codes have not been accelerated e.g. for GPUs and it is increasingly hard to maintain parallelisation – which will make them increasingly difficult to use on next-generation PRACE systems.

The goal of the NB-LIB project is to address this with a library of cutting-edge-performance nonbonded interactions as well as a parallelization framework that can be used by all these applications. This way, future acceleration, porting or library features will benefit all applications making it easier to convince vendors to contribute such efforts. We will achieve this by separating the lower-level parts of the GROMACS code into an API of non-bonded force routines to enable re-use of widely used, highly portable, and performant HPC code. This will benefit applications in multiple domains and leverage Exascale optimization underway in co-funded and collaborative projects. The library will be available via APIs in industry standard languages, including both python3 and C++14. This will permit domain scientists to use the API to prototype and deploy solutions for new N-body simulations rapidly, leveraging existing knowledge and best practices, rather than learning how to modify existing code. This will enable innovation across disciplines as we approach the Exascale era, while still providing familiar tools to HPC developers, rather than require knowledge of niche runtimes or languages. The project will work alongside co-funded PRACE-6IP and Swedish HPC projects, as well as distinct API efforts funded in the USA. It will provide efforts that will advantage both users of the NB-LIB library and API, and the underlying GROMACS simulation back end.

The disruptive innovation within GROMACS will stimulate existing modernization and optimization efforts. It will evolve testable software seams to permit non-bonded force-calculation, which will be tailored by GROMACS for correctness and performance for particular Exascale node structures and/or user workloads. GROMACS is a widely used and highly targeted code, within PRACE and around the world, which will ensure sustainability of the library and its API. Once deployed, the API will provide the leverage for future innovation, for example to deploy Exascale-suitable FMM implementations as drop-in replacements, so that user-level Python workflows will require no changes.

The resulting library and API will be available under a highly permissive license such as the LGPL v2 used currently for GROMACS (no restrictions whatsoever on linking to the library). Sustainability will be assured through integration with existing GROMACS development workflows and roadmaps, and tests of the API functionality will be developed and deployed the same way. Core GROMACS developers are closely involved in the project to facilitate this.

Task 1. Refactoring of various aspects of GROMACS simulation setup and execution code to minimise the need for large amounts of adapter code. Currently simulation setup procedures in GROMACS are targeted towards biomolecular simulations, which has historically been the primary usage of GROMACS. These setup procedures are also implemented as a pipeline which is targeted to biomolecular simulations and the procedures have an implicit assumption that a biomolecular force-field will be used. Further, the actual simulations that GROMACS runs are currently exposed as command line binaries.

The aim of this task is to focus on refactoring GROMACS data structures so that they can accept objects built outside of any GROMACS software pipeline, such as will be the case for the NB-Lib API. This will align nicely with ongoing work in BioExcel to refactor user-facing aspects of the GROMACS codebase.

In addition to simulation system setup refactoring, some work will be spent on refactoring GROMACS simulation execution to move towards simulations being built by external callers, such as would be the case with the NB-Lib API. This will align well with ongoing work by various NIH (National Institutes of Health) projects to make GROMACS simulation execution more modular.

Task 2. Implementation of API prototype that will be possible to extend to facilitate different use cases. Initial efforts will focus on the simplest use case that could have scientific value, namely, a simulation of Lennard-Jones particles in a cubic periodic box on a single node. This use case will still be able to take advantage of GROMACS excellent parallelism using multiple threads and/or GPUs on a single node, while leaving aside the question of how to allocate resources in a multiple-node, MPI context. This focus will mean that emphasis can be placed on user experience and user interface (UX/UI) in the initial development phase. This early emphasis on UX/UI should help drive uptake of the API, resulting in early feedback from early adopters that can drive further API development.

Task 3. Collection of use cases that can guide future development goals. While there are many possible simulation systems that could be targeted by NB-Lib, in order to focus resources, specific targets will be chosen based on on-going discussions with potential users. Eventually this may also expand to include incorporation of feedback from early adopters into development targets. However, for the coming period, most focus here will be on gathering user stories and use case scenarios.

4.2 Project schedule

Milestones

- **MS1 (M6)** Project staffing in place and project successfully initiated.
- **MS2 (M12)** Release of public API prototype
- **MS3 (M24)** Release of public API (v1.0)
- **MS4 (M30)** Performance testing on pre-Exascale systems

Deliverables

The deliverables will be in accordance to those planned in the WP8 schedule:

- **D1 (M6)** Initial report on project staffing and project initiation
- **D2 (M12)** Report of public API prototype release
- **D3 (M24)** Report on public API (v1.0) release
- **D4 (M30)** Final report including performance testing on pre-Exascale machines

4.3 Team and collaboration

The team consists of two partners, The Royal Institute of Technology (KTH) in Sweden, and The Swiss National Supercomputing Center (CSCS). KTH is leading GROMACS development and

API planning/integration efforts while CSCS is leading API implementation and performance testing as well as use case collection.

KTH

Berk Hess (PI): Provide guidance on and refactoring of neighbour search code in GROMACS. Neighbor searching is a key aspect of GROMACS parallelization and will be necessary for a performant API.

E. Joseph Jordan: Main liaison between GROMACS development team at KTH and CSCS. Plan API design and refactor GROMACS internal data structures to minimise the need for boilerplate and adapter code in API.

Paul Bauer (external contributor): GROMACS product manager. Provide guidance on and refactoring of GROMACS data structures pertaining to atomic coordinates and forces parameters.

CSCS

Victor Holanda Rusu: Main liaison between CSCS and GROMACS development team at KTH. Provide some benchmarking of GROMACS and API deliverables.

Prashanth Kanduri: Expose GROMACS seams that will allow API calls to non-bonded force calculation. Write API adapters and user-facing elements.

Sebastian Keller (external contributor): Expose GROMACS seams that will allow API calls to non-bonded force calculation. Write API adapters and user-facing elements.

Channels

Weekly teleconferences are held between KTH and CSCS participants. In fall or winter, KTH participants will have an in person planning meeting/hackathon at CSCS. In winter or spring CSCS participants will attend annual GROMACS developer planning meeting (location TBD) where the wider GROMACS community comes together to decide on development goals for GROMACS 2021 release. Further in person meetings will be planned as needed. Code is shared and reviewed in a variety of places, including GROMACS gerrit, gitlab, and redmine.

4.4 Status and outlook

Project staffing has been completed. Refactoring work is underway to expose necessary seams for NB-Lib API. An in person meeting will be held at CSCS before the end of 2019 and another meeting will be held next year location TBD. The prospects for having an initial prototype implementation in line with deliverable **D2** is good. Overall, the NB-Lib project is off to a promising start.

5 LoSync - Synchronisation reducing programming techniques and runtime support

5.1 Introduction and objectives of the project

The LoSync project will focus on improving the scalability of applications by removing unnecessary synchronisation and serialisation, and full realising opportunities to overlap calculation and communication. To do this, we will make use of modern features of well-standardised APIs, to ensure portability and relevance. These techniques will include:

- Using OpenMP/OmpSs tasks with data dependency clauses. This will include not only expressing computation as tasks, but also communication, by wrapping MPI or GASPI library calls inside tasks. We will utilise the Task-Aware MPI (TAMPI) and Task-Aware GASPI (TAGASPI) interoperability libraries developed by the INTERTWinE project to make this as efficient as possible.
- MPI single-sided communication. Recent developments in MPI libraries have significantly improved the performance of single-sided communication to the point where its benefits can be realised in real applications.
- GASPI single-sided (put-notify) communication. This is a lightweight alternative to MPI single-sided communication which interoperates well with MPI, and offers different synchronisation semantics which can help remove serialisation constraints.

A significant part of the work of this task will consist of implementing these techniques in key kernels of important applications such as Gysela, Flucs, IFS, Quantum Espresso, CP2K, iPIC3D, CASTEP, LULESH, NTCHEM, as well as, some of the applications involved in the DEEP-EST project. The project will also make use of smaller kernels and mini-apps, sourced from the PRACE CodeVault, the INTERTWinE project resource packs, and elsewhere as appropriate.

In addition, work will be carried out on runtime library implementation to support this work, building on development in the INTERTWinE FET-HPC and POP CoE projects. This will include:

- Continuing the development of the INTERTWinE TAMPI and TAGASPI interoperability libraries to support interaction of MPI and GASPI with OpenMP/OmpSs tasks with dependencies.
- Continuing the development of the INTERTWinE Resource Management interface and its implementation.
- Exploring extensions to the OpenMP tasking model to support task dependencies on external events, task-nesting, fine-grained dependencies, weak dependencies and early release of dependencies, to avoid artificial synchronization and serialization effects.
- Making use of performance analysis tools and techniques developed in POP to identify optimisation targets in real applications where these techniques can be most beneficially applied.

The project will engage with the relevant standards bodies for MPI, OpenMP and GASPI, to track relevant upcoming features, propose new features where necessary, and to produce prototype implementation to test these features. The project will engage with the application developer community through training courses, hands-on developer workshops, and targeted website materials.

Task 1: Runtime enhancements

This task will extend the Task-Aware MPI (TAMPI) and the Task-Aware GASPI (TAGASPI) libraries to further improve the interoperability of tasking systems and message passing APIs. The TAMPI library currently supports synchronous two-sided MPI primitives inside tasks. In this task we will extend TAMPI to support both asynchronous and single-sided MPI primitives inside tasks. These extensions will widen the applicability and performance of the TAMPI library. Moreover, the TAGASPI library, which is currently under development, will be completed and evaluated on various systems. The OmpSs-2 runtime system will be extended with the required interfaces to effectively support both libraries, which rely on the Pause/Resume API, the external Events API and the polling services offered by the OmpSs-2 runtime. New tasking features such as advanced task-nesting, fine-grained dependencies, weak dependencies and early release of dependencies, will be investigated to further improve the integration of tasking systems and message passing APIs based on the feedback from Task 2.

Task 2: Kernels and Applications

This task will implement synchronisation minimising programming techniques in kernels, proxy or mini apps and in real application codes. Initially the focus will be on simpler examples, in order to gain experience, and understand in greater depth the software engineering challenges involved in converting “conventional”, two-sided MPI (or MPI + OpenMP loops) codes to use dependent tasks for both computation and communication. The project will analyse a selection of full application codes for their asynchrony potential (building on techniques developed by the POP CoE) and identify candidates where there are potential gains to be made and where the software engineering challenge is tractable. Emphasis will be given to applications where the developers are willing and able to collaborate on the refactoring work. Careful performance analysis experiments will be carried out, so that the work in Task 1 can be properly evaluated, and an in-depth understanding of not just whether, but precisely why, performance can be gained with these methods. The task will feed back performance data and additional feature requests to Task 1 to inform the development of the task-aware communication libraries and task-based runtime system.

Task 3: Dissemination and exploitation

This task will ensure the effective dissemination and exploitation of the work in Tasks 1 and 2, via the following mechanisms:

- Training courses aimed at application developers and delivered via the PTC Training Programme
- Developer workshops, including hands-on coding sessions
- Online material, including software releases, demo kernels and best practice guides
- Ad-hoc consultancy for interested development teams
- Publications in conference proceedings and journals
- Promotion of the project on social media

Task 4: Management

This task will ensure the effective management and oversight of the project, including:

- Organising and chairing meetings (online and face-to-face)
- Monitoring progress against milestones
- Producing timely, high-quality deliverables
- Reporting to PRACE 6IP WP8 management

5.2 Project schedule

Milestones

- **MS1 (M6)** Project staffing in place, project structure established, and project successfully initiated.
- **MS2 (M12)** Prototype releases of TAMPI and TAGASPI libraries, and initial versions of kernels and mini-apps. Plans for full application development work completed.
- **MS3 (M24)** Updated releases of TAMPI and TAGASPI libraries, updated versions of kernels and mini-apps. Use of libraries demonstrated in real application codes.
- **MS4 (M30)** Performance results of kernels, mini-apps and full applications on large scale systems.

Deliverables

- The deliverables will be in accordance to those planned in the WP8 schedule: **D1 (M6)** Initial progress report on staffing, established project structure and project initiation.
- **D2 (M12)** First interim progress report, including descriptions of prototype releases of TAMPI and TAGASPI libraries, initial versions of kernels and mini-apps and plans for full application development work.
- **D3 (M24)** Second interim progress report, including descriptions of updated releases of TAMPI and TAGASPI libraries, updated versions of kernels and mini-apps and results of full application development work.
- **D4 (M30)** Final report, including performance results on large scale systems.

5.3 Team and collaboration

The team consists of two partners, UEDIN (represented by EPCC) and BSC. UEDIN leads the project and is focusing on the implementation of synchronisation reducing techniques in kernels and applications. BSC is focusing on providing enhanced support for these techniques in the OmpSs-2 programming model and TAMPI and TAGASPI libraries. UEDIN acts as co-ordinator and performs project management tasks. The project is fully staffed as follows:

UEDIN

Mark Bull (PI), Caoimhín Laoide-Kemp, Dominic Sloan-Murphy

BSC

Vicenç Beltran Querol (Co-PI), Raúl Peñacoba Veigas, Kevin Sala, Marcos Maroñas

A face-to-face kick-off meeting was held in Edinburgh on 2nd July 2019. This has been followed by a number of Skype meetings. Marcos Maroñas commenced a 3-month visit to Edinburgh under the HPC-Europa3 programme on 2 September 2019. The next face-to-face meeting is planned to be in Barcelona in December.

- The OmpSs-2 programming model is publicly available at GitHub (https://github.com/bsc-pm/ompss-2-releases).
- The Task-Aware MPI (TAMPI) prototype library is publicly available at GitHub (https://github.com/bsc-pm/tampi).
- The Task-Aware GASPI (TAGASPI) library will be publicly available at GitHub.

We will integrate the modifications done to kernels, proxy-apps and applications into their official distributions when possible. Otherwise, we will include them in some of the software repositories maintained (https://pm.bsc.es/gitlab/ompss-2/examples) or hosted (https://pm.bsc.es/gitlab/benchmarks/ompss-2) by BSC.

5.4 Status and outlook

Initial work at UEDIN has focused on analysing the Mantevo mini-apps collection and assessing their suitability as text cases for the MPI + tasks programming model. Of these, Co-MD and miniMD have been selected for porting: although they are both molecular dynamics applications, they uses different algorithms (cell based vs neighbour-list based) which require different tasking strategies, and will make an interesting comparison. Work will continue on these, and then move on to other mini-apps in the near future.

At BSC, other kernels and mini-apps have been ported to evaluate the OmpSs/OpenMP tasks + TAMPI implementation: this evaluation work is ongoing in the project. On the runtime, side, work will now concentrate on support single-sided message passing inside tasks, both with MPI (in the TAMPI library) and GASPI (in the TAGASPI) library. The TAGASPI and TAMPI (with single-sided support) libraries will be published as a pre-release of WP8 software output (Milestone MS2). Recent work on exploiting loop parallelism within tasks, in order to reduce tasking overheads, will be evaluated in the context of MPI + tasks, with the intention of helping the user navigate the complex trade-off space between numbers of MPI processes, number of threads and task granularity.

6 FEM/BEM based domain decomposition solvers

6.1 Introduction and objectives of the project

Solution of large scale multiphysics engineering problems requires a significant amount of computational resources. However, most of the existing commercial or open source solvers are not capable of utilizing full potential of current or upcoming supercomputers equipped with large number of heterogeneous nodes, processors with wide SIMD (Single Instruction Multiple Data) registers, accelerator cards, etc.

The aim of this project is to extend the domain decomposition library ESPRESO to support highly scalable solution of problems in complex domain using finite/boundary element tearing and interconnecting (FETI/BETI) non overlapping domain decomposition method, thus enabling solution of large scale sound scattering and harmonic analysis problems. The current implementation in ESPRESO (similarly as in most other solvers) focus on the elasticity or the heat transfer problems for which it features excellent parallel and numerical scalabilities. The implementation for complex problems (such as solution of the Helmholtz equation) will be based on the FETI-H/FETI-DPH (FETI-Helmholtz/FETI-Helmholtz dual-primal) approach where the regularization is done using the complex interface mass matrix and preconditioning is based on the plane wave deflation.

The goal is to provide a modern, modular and portable code written in C++, parallelised on all possible levels, and capable of utilizing the most powerful supercomputers. Basic refactoring of the original ESPRESO code providing wrappers to external libraries such as BLAS or parallel direct solvers, which should enable faster development, will be part of the project. The development will take into account the heterogeneous nature of current and future supercomputers. The code will be parallelised in a hybrid manner in shared and distributed memory and will benefit from the SIMD vector capabilities of modern CPUs. The parallelization in shared memory and vectorization will be done using the OpenMP pragmas. The computationally most demanding parts of the code will be accelerated using state of the art GP-GPUs. MPI will be used for a distributed memory parallelization. The implementation will be made available to developers of external libraries using a public API with extensive documentation and, moreover, the solver will be incorporated into a “Solver as a service” platform at IT4Innovations, thus enabling scientists and engineers with zero experience in HPC to leverage the power of supercomputers. This effort will also provide a highly scalable easy-to-use open-source alternative to commercial packages such as ANSYS or COMSOL.

Work on the project is split into several phases.

Refactoring. The ESPRESO library now provides users with unified interface and input files, it has an extensive help with examples and a GUI is under development. However, the project will start with refactoring its internal structure to more reflect the modern object-oriented and modular design. The main purpose of this refactoring will be to make the internal structure more easily modifiable and extendable, to provide developers with easy-to-use interface to external libraries such as BLAS for CPUs or accelerators, and parallel direct solvers. During refactoring the espresso code, a prototype FETI-H code in Matlab will be developed for testing purposes, fast prototyping, etc.

Shared memory implementation and optimization. After the refactoring, we will implement the initial non-accelerated version of the solver for the Helmholtz equation which will feature a hybrid parallelization by OpenMP in the shared memory. Within the shared memory, a special attention will be paid among other things to the SIMD vectorization of the system matrix assembly. Since modern CPUs are equipped with the SIMD registers with of up to eight double operands (using instruction set extensions such as AVX, AVX2, AVX512), neglecting this feature would lead to a significantly less efficient code. Vectorization and shared memory parallelization will be based mainly on the OpenMP 4.5 pragmas.

GPU acceleration. Since the architecture of current supercomputers is becoming increasingly heterogeneous and this trend is expected to continue in the upcoming years, it is necessary to reflect it when designing a solver capable of fully employing state-of-the-art HPC clusters. We will leverage our experience with accelerating the FETI method by converting operations with sparse data structures to operations with dense matrices using the Schur complement method on both Intel Xeon Phi coprocessors and NVIDIA graphic cards. Dense data structures are more suitable for accelerators since they allow for coalesced memory access pattern, which leads to a better usage of accelerator's high bandwidth memory. However, since the Intel Xeon Phi architecture will no longer be developed by Intel, we will now mainly focus on the acceleration by GPUs. The most time-consuming parts of the code will be accelerated using the CUDA platform.

MPI parallelisation. Simultaneously with the intra-node optimization, parallelization, and acceleration, attention will be paid to the distributed memory parallelization using MPI. We will leverage our previous experience with parallelization of FETI and its variants Total-FETI (TFETI), Hybrid FETI (HFETI), and Hybrid Total FETI (HTFETI) for the Laplace and Lamé equations. The main bottleneck of the FETI method is the solution of the coarse problem; size of this problem grows proportionally to the number of subdomains. The hybrid variants of FETI (HFETI, HTFETI) combine FETI and FETI-DP to group subdomains into larger clusters thus reducing the size of the coarse problem. If needed to ensure scalability of the solution, we will provide hybrid versions of the FETI for complex problems. Moreover, if it will be necessary to reduce overhead by communication, communication hiding and avoiding algorithm may be employed.

Other tasks include development of the external API to make the methods easily available for third-party solvers, continuous testing and development of documentation, establishment of a webpage for dissemination of the project's results. When the code development is finished and properly tested for performance, scalability and correctness of solution, it will be integrated into the "Solver as a service" online platform simultaneously developed at IT4Innovations. The purpose of this platform is to make the high-performance computing easily available to scientists and engineers without proper training in parallel algorithms and parallel programming. In the long run the aim is to provide a highly scalable platform for engineering computing and alternative to current commercial engineering software.

6.2 Project schedule

Tasks

- **Task 1** *Refactoring of the existing code, development of the Matlab prototyping application*. Including object-oriented redesign considering modularity and portability; implementation of core classes and wrappers to external software; in the final phase development of external API.
- **Task 2** *Shared memory implementation, optimization, and parallelization of FETI for complex problems and simultaneous development of the BEM API.* Includes system matrix assembly vectorised and parallelised in shared memory, implementation of local subdomain solves.

- **Task 3** *Acceleration of the computationally intensive kernels using GPUs*. This includes system matrix assembly acceleration; local subdomain solver acceleration using Schur complement method.
- **Task 4** *Distributed parallelization using MPI*. Including implementation of FETI for complex problems and its hybrid total variants.
- **Task 5** Testing and documentation. Includes initial creation of testing infrastructure; testing performance and correctness; creation of documentation and examples; and, in the last phase, testing of external API.

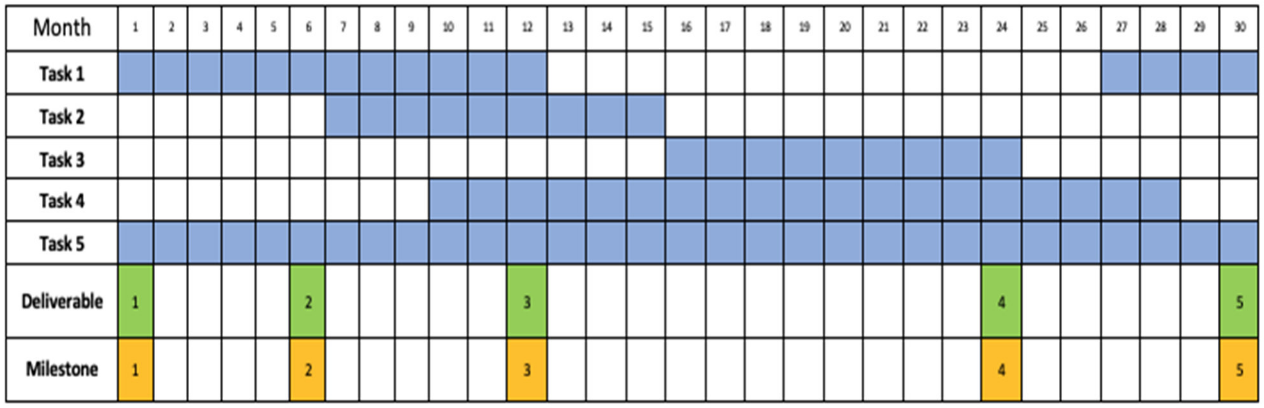
Deliverables

The deliverables will be in accordance to those planned in the WP8 schedule:

- **D1** *Project kick-off (M01).*
- **D2** *Interim progress report: staff and established project structure (M06).*
- **D3** *Interim progress report: public prototype software release (M12).*
- **D4** *Interim progress report: Public Software release and integration in external codes (M24).*
- **D5** *Final report: including performance results on (pre)Exascale system (M30).*

Milestones

- **MS 1 (M1)** *Kick-off meeting*
- **MS 2 (M6)** *Initial redesign and refactoring of the original code finished.*
- **MS 3 (M12)** *Initial shared and distributed memory parallelised version of FETI for complex problems available.*
- **MS 4 (M24)** *Software is publicly released supporting hybrid variant of FETI for complex problems, GPU acceleration.*
- **MS 5 (M30)** *The external API is finished and tested, final report is delivered.*



Month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Task 1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															🟦	🟦	🟦	🟦
Task 2							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
Task 3																🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦						
Task 4										🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Task 5	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Deliverable	1					2						3												4						5
Milestone	1					2						3												4						5

Table 4: Timeline of FEM/BEM milestones and deliverables

6.3 Team and collaboration

The project team consists of researchers employed at IT4Innovations National Supercomputing Center in Ostrava, Czech Republic. The researchers have experience with development of parallel scientific software, its optimization and testing on state-of-the-art HPC platform. They participated or participate, e.g., in the H2020 projects (e.g., READEX, EXA2CT, POP2), national grants, or Intel supported IPCC (Intel Parallel Computing Center) project. The development is supported by 62 person-months split among the team:

- **Michal Merta (PI)** – involved in Tasks 1, 2, 3, 5.
- **Jan Zapletal** – involved in Tasks 2, 3, 4, 5.
- **Lukáš Malý** – involved in Tasks 3, 4, 5.
- **Ondřej Meca** – involved in Task 1, 4, 5.

The researchers will be supported by the ESPRESO development team, namely by Tomáš Brzobohatý, one of the founders of the ESPRESO library.

Since the employees come from a single institute, they meet in person at regular basis; regular meetings have been established. Git-based repositories are used for code development and a webpage was created to disseminate the project's results:

- http://numbox.it4i.cz - the main communication channel, which will contain all the important information and links to external repositories of the project and relevant resources
- https://github.com/It4innovations/espreso - publicly available repository
- https://code.it4i.cz – developer (protected) repository
- http://espreso.it4i.cz/doc/index.html - documentation.

6.4 Status and outlook

The project was successfully initiated after the PRACE-6IP kick-off meeting in May 2019. Recruitment of the staff is now completed and tasks are assigned to the involved researchers. Besides the establishment of the git repositories and webpage, the current work has been split into two subtasks – refactoring of the ESPRESO code and development of the Matlab code for fast prototyping.

The code refactoring consisted largely of the development of interfaces to external direct solvers. Espreso now supports solution of large sparse linear systems by the MPI version of Pardiso available in the Intel MKL (Math Kernel Library), as well as the original Pardiso 6, SuperLU, and Watson Sparse Matrix Package (WSMP) developed by IBM. Moreover, domain/graph decomposition by multiple graph partitioning software (Metis, ParMetis, Scotch, PT-Scotch, or KaHIP) is now available which enables combination of different partitioners for multilevel domain decomposition method (ParMetis/PT-Scotch on the MPI level, Metis/Scotch/KaHIP on the node level). Parallel solution of the harmonic analysis problems now supports automatic decomposition of the spectrum band across computational resources. However, the code currently only supports decomposition in the frequency domain. Finally, to improve users' experience, the structure of the ESPRESO configuration file which serves to set parameters of the solver, has been redesigned.

Furthermore, a testing FETI-H code within the MatSol library written in Matlab was implemented. The purpose of the code is to have a tool for fast prototyping of various domain decomposition methods suitable for problems in the complex domain. Regularization of stiffness matrices by interface mass matrices has been implemented and tested on academic examples (cubical domains with point acoustic source).

Within the next six months, the Matlab code will be extended to support preconditioning by projection to an artificial coarse space. After testing the functionality of the prototype code, the implementation in the ESPRESO C++ library will start. We will start with implementation and optimization in the shared memory, using OpenMP pragmas. Next, distributed version will be developed using MPI.

7 Performance portable linear algebra

7.1 Introduction and objectives of the project

In general, linear algebra algorithms have a central role in scientific applications: (i) In the particular case of Materials Science many applications rely heavily on linear algebra to solve complex tasks. A subset of these applications crucially depends (also in terms of performance) on the solution of distributed dense eigenvalue problems; (ii) in non-linear optimization the interior point methods must solve in each iteration a system of linear equations involving the so-called Schur Complement Matrix which is dense, has to be generated in each iteration separately, and is the standard bottleneck of these methods; (iii) in big data analysis dealing with high dimensional data several classical methods like discriminant analysis or linear regression amount to computing inverse or spectral factorization of a large dense matrix. Overall, the diversity of linear algebra operations together with the large size of the operands motivates the necessity of high-performance implementations of distributed algorithms.

For example, modern electronic structure methods rely on the Density Functional Theory (DFT) method, in which the exponential complexity of the many-body Schrödinger equation is reduced to a computationally tractable problem with polynomial complexity which requires the solution of the decoupled Kohn-Sham equations. After discretization, such equations translate into either algebraic eigenvalue problems or linear systems. Depending on the discretization method used, the eigenvalue problems may be dense or sparse. Dense eigenvalue problems are currently solved mainly using the ScaLAPACK (http://www.netlib.org/scalapack) or ELPA (https://elpa.mpcdf.mpg.de/software) libraries.

ScaLAPACK has been developed and is maintained by the Innovative Computing Laboratory (ICL) at the University of Tennessee. During the process of planning the roadmap to provide a successor of ScaLAPACK, ICL performed a study [2] to identify the current status of linear algebra libraries, hardware architectures and the structure and features needed by a modern library.

The study analyses the processing units, memory types, and communication links, programming models, matrix layout and algorithms and clearly identifies the ScaLAPACK limitations. The study also identifies a problem for the current approach of offloading high parallel work to the GPU and execute the rest of the operations on the CPU, since most of the raw performance of the supercomputers is provided by the GPUs. The study also highlights the improvement trend in the performance of nodes and bandwidth of the interconnect of the computing systems. Bandwidth is becoming scarce and therefore techniques of dynamically rebalancing work should not be considered and statically partitioned matrices have to be considered, leaving dynamic schedule only at the node level.

An alternative strategy in the development of an eigensolver is to leverage on well-known and well-established iterative algorithms such as subspace iteration. A modern example of such algorithm has recently being implemented in the Chebyshev Accelerated Subspace iteration Eigensolver (ChASE) library. When tackling sequences of Hermitian eigenproblems, as they often appear in electronics structure codes, ChASE takes advantage of the distinctive features connecting adjacent problems in a sequence. At the core of the library, the polynomial degree of the Chebyshev filter is optimised so as to reduce up to 20% the number of FLOPs necessary to declare each of the eigenpairs converged.

The goal of this project is to provide a modern and efficient distributed linear algebra package based on HPX (https://github.com/STEllAR-GROUP/hpx), that can replace ScaLAPACK in scientific applications, and help the developers of scientific applications in the process of adopting modern, performance portable, and distributed linear algebra libraries. Since independent tasks can be executed easily in parallel, libraries built on task-based runtime can improve significantly the parallel efficiency of single function calls, and solve the fork-join mechanism issue of ScaLAPACK. In general task-based runtimes simplifies the scheduling of small non-parallelizable independent tasks on different cores, that in general are executed sequentially.

An API based on HPX future (https://en.cppreference.com/w/cpp/thread/future) will be developed and will allow tasks of different routines to run concurrently creating a single dependency graph for the full application. The dependency tracking of the tasks of the full application can have a large impact on the performances, since the hardware resources can be used more efficiently reducing the idle time.

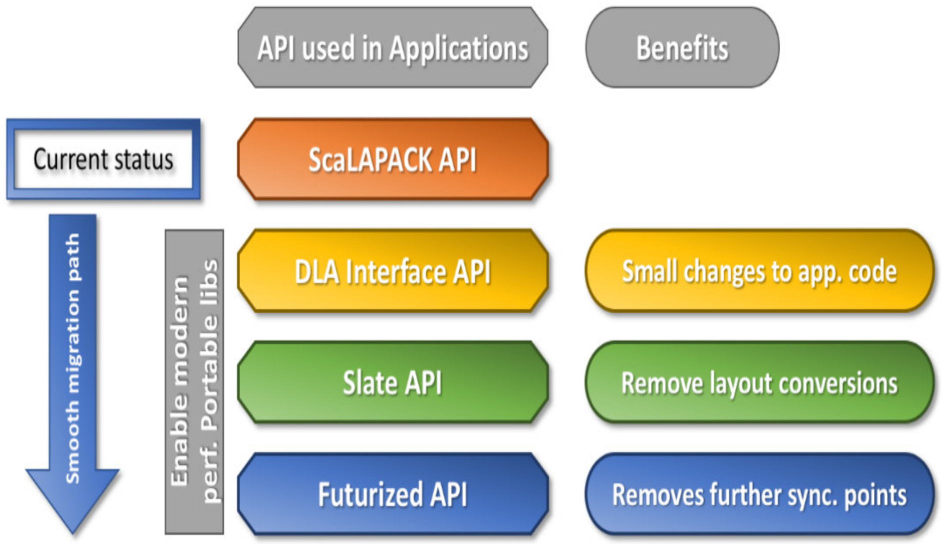


Figure 4: Smooth migration from ScaLAPACK to modern libraries

To simplify the adoption of new libraries we will provide different levels of APIs presented in Figure 1. The first and easiest step is to replace direct ScaLAPACK calls with DLA Interface calls. This method requires little changes to the application code, but some overhead for the matrix layout conversions has to be taken into account. To fully exploit the benefits of the new libraries the native interfaces has to be used but it requires more changes of the source code.

To address the different project goals, we define 4 tasks:

- Task 1 (T1): Preparation of a migration guide,
- Task 2 (T2): Improvement of the DLA interface library supporting new libraries and routines,
- Task 3 (T3): Collaboration for the development of eigensolvers in SLATE, and implementation of a HPX based version of SLATE,
- Task 4 (T4): Integration in applications.

The first two tasks prepare the tools needed for the smooth migration to modern linear algebra routines. The first task (T1) will focus on the development of the migration guide for the DLA interface library which will facilitate the migration of current scientific application.

7.2 Project schedule

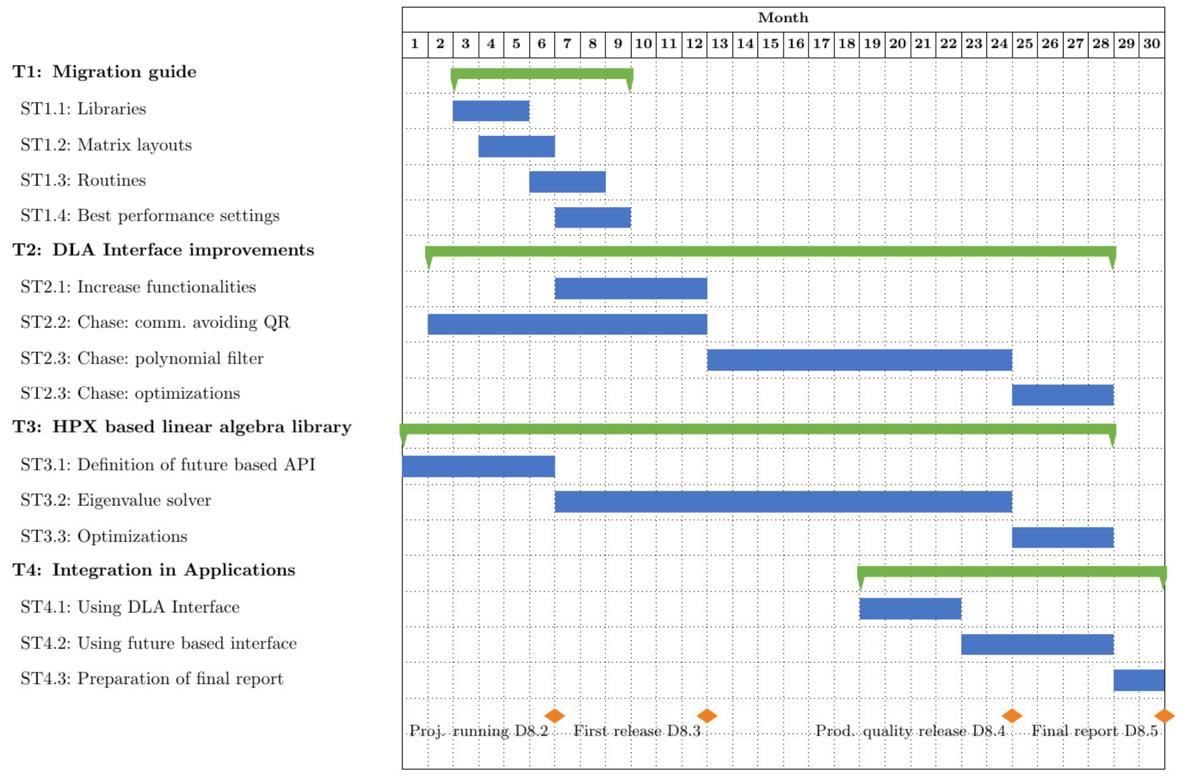


Table 5: Performance portable linear algebra timeline of subtasks and deliverables

T1 is divided in four subtasks (ST). The description of the different libraries included in DLA interface will be added to the guide by ST1.1. The content of the guide regarding the different matrix layouts, the matrix interface and the conversion methods will be created in ST1.2. ST1.3 will document the routines available in the DLA interface library and their usage. It will also document the limitations of the routines of the different libraries compared to ScaLAPACK (e.g. constraints on the choice of the block sizes for the 2D block cyclic distribution). ST1.4 will contribute to the description of the performance of the different libraries and of the selection procedure of the different parameters to achieve the best performance.

T2 will contribute to the improvement of the DLA interface routines which compute the eigenvalue and eigenvectors. ST2.1 will contribute adding the ChASE eigenvalue solver to the library to give access to eigenvalue solvers optimised for distributed multi-GPUs clusters. In addition, ST2.2 and ST2.3 will improve some of the key kernels of the ChASE library (e.g. skinny QR decomposition), and extend its support to generalised Hermitian eigenproblems without the necessity of Cholesky factorizations, and include partial SVD computations.

The major contribution of this project is however included in T3 in which we propose to provide an HPX based distributed linear algebra library (called DLA-Future). The library will provide the standard SLATE API and an API based on C++ std::future.

The fourth task of the project (T4) will contribute with the integration of the libraries developed in the previous tasks in the scientific applications, such that they can benefit of the better efficiency and performance portability of the new implementations. The focus will be on Quantum ESPRESSO, SIRIUS and CP2K, applications used by the material science community in scientific publications. Additionally, we will assist the optimization community to improve the solvers for linear conic programming by integrating the new libraries.

7.3 Team and collaboration

The project has four contributors: ETH Zurich, CINECA, University of Ljubljana and Jülich Supercomputing Centre.

ETHZ:
Raffaele Solcà (PI), Alberto Invernizzi, Alo Roosing

CINECA:
Fabio Affinito (coPI), Alessandro Colombo (until September 30), Lara Querciagrossa

University of Ljubljana:
Janez Povh (coPI), Aleksander Grm, Gregor Simič, Timotej Hrga

JSC:
Edoardo Di Napoli (coPI), Xinzhe Wu

ETHZ and CINECA will contribute to the development of the Eigensolver based on HPX, University of Ljubljana will contribute to the improvement of the documentation and to the migration guide. JSC will contribute to the further development of the ChASE library and its documentation.

For collaboration the team set up a Slack channel (linalg6ip.slack.com) and two mailing lists (one for PI and coPIs, the second one for the full team).

Each of the libraries is hosted on his own git repository:

- **DLA-Interface** https://github.com/eth-cscs/DLA-interface
- **DLA-Future** https://github.com/eth-cscs/DLA-Future
- **ChASE** https://github.com/SimLabQuantumMaterials/ChASE

Meetings:

On 11 June 2019 the kick-off video conference with all members of the project took place.

7.4 Status and outlook

The project started successfully and the work proceed. A kick-off videoconference has been organised with all the project collaborators to discuss about the project details. The work on the Cholesky prototype continued with two experiments. We performed a comparison between column-major and tile layout which showed no performance difference. On the other hand we performed some experiments on communication which showed an overall improvementof the performance of the prototype.

Based on the results of the experiments we started the design of the API based on HPX future and the refactoring of the Cholesky prototype. An open repository has been set up for the distributed linear algebra library based on HPX futures (DLA-Future) and a continuous integration service has been setup to run tests for each Pull Request made on the GitHub repository.

The work on the ChASE library started as well. The user documentation has been improved and it can be found online at https://simlabquantummaterials.github.io/ChASE/index.html. In addition the parallel I/O of the library has been modernised and a HPX based Tiled QR decomposition has been developed and the work to integrate it inside ChASE to replace the LAPACK QR has started.

In the next six months we aim to complete the design of the HPX based linear algebra API, finish the refactoring of the Cholesky decomposition and develop the conversion from generalised to standard eigenvalue problem. Moreover we plan to improve the documentation of DLA-Interface and start writing the migration guide. For the ChASE library we plan to continue the benchmark and implement a partial SVD. In addition we aim to improve the developer documentation and to simplify validation and further development of the ChASE eigensolver we plan to implement and test a Julia-based prototype.

Scotch, PT-Scotch, or KaHIP is now available which enables combination of different partitioners for multilevel domain decomposition method (ParMetis/PT-Scotch on the MPI level, Metis/Scotch/KaHIP on the node level). Parallel solution of the harmonic analysis problems now supports automatic decomposition of the spectrum band across computational resources. However, the code currently only supports decomposition in the frequency domain. Finally, to improve users' experience, the structure of the ESPRESO configuration file which serves to set parameters of the solver, has been redesigned.

8 GHEX: Generic Halo-Exchange for Exascale

8.1 Introduction and objectives of the project

Applications in different scientific domains use various types of computational grids. Cartesian topologies with and without regular spacing or curvilinear, they can represent true 2D and 3D geometries; block-structured grids in which each block exhibits regularities but their connection are general; Fully unstructured meshes in which the topology of the connections arbitrary.

Because of the topological differences between the individual grids, and in how the grid data structures are organised in individual applications, the halo zones are defined and stored in memory differently for each of them. GHEX will strive to provide a clean and performance-portable implementation of halo exchange for the most important grid types independently of the particular applications characteristics. Priority will be given to grids used by the strategic scientific collaborators, among which MeteoSwiss and the COSMO weather and climate model, and the Rosseland Centre for Solar Physics (RoCS) and the DISPATCH modelling framework.

GHEX will be used as a communication layer in GridTools - a set of open-source libraries to develop next generation weather and climate simulation applications in a common infrastructure. One ambitious objective is to run high-resolution whole-globe simulation on a grid with less than 1 km spatial resolution. Several other ongoing collaborations are also employing GridTools at different stages, e.g., NICAM model developed at Riken Center for Computational Science, ICON at the Max Planck Institute, the Atlas library at the European Centre for Medium-Range Weather Forecasts (ECMWF), and in the context of the European project ESCAPE-2.

GHEX is developed in a collaboration between the *Swiss National Supercomputing Centre (CSCS)*, and the *University Center for Information Technology* (*USIT*, University of Oslo, Norway). At USIT the aim is to integrate the communication primitives library into DISPATCH - a task-based numerical framework for the solution of partial differential equations on Exascale systems, and BIFROST stellar atmospheric model. DISPATCH currently supports Cartesian and orthogonal curvilinear meshes and implements its own MPI-based halo exchange collective, which is not designed to work on accelerators. With GHEX integration our goal is to make DISPATCH more portable, and provide it with a larger choice of communication backends optimised for present day and future HPC architectures. Since DISPATCH is a modern task-based framework that has a demonstrated scalability on tens of thousands of CPU cores, including on PRACE Tier-0 systems, it is a perfect target to integrate GHEX, but also to provide invaluable feedback during the design and the development of the library. BIFROST is a more traditional application that will benefit from GHEX for portability to other platforms.

To target this wide range of applications GHEX will support the following grid types

- Regional regular latitude-longitude topologies with Cartesian topology
- Block-structured grids on the sphere, such as inflated cubes, global latitude-longitude, icosahedral
- Fully general, unstructured meshes

Traditionally, scientific applications are designed to work on one grid type. The communication primitives are implemented directly by the developers using low-level interfaces (usually MPI). This approach has two drawbacks: first each group ends up developing similar software for the halo-exchange; second, the performance portability is often limited since the different computing platforms require distinct strategies for optimising performance. Understandably, scientists and modellers specialise within their field and need not be experts in performance optimization at the same time. Unless they are not explicitly interested in optimizing their software for various types of HPC hardware, they often implement for only a single machine and don't update/refactor their software for new architectures. This often causes the effort to be short lived or become inefficient over the years.

A solution to these problems is to use high-level interfaces that hides the architectural details and allow the HPC experts to provide optimised implementations. In this way only one refactor is needed to modernise the code and then the code can run on multiple platforms. Although the general approach is not new, a successful implementation is challenging, especially to support multiple parallel programming models (e.g., MPI, multithreading, or task-based). To become widely adopted and future-proof, such interface should:

- Define application-level semantics that can be supported efficiently by a wide range of present and future platforms
- Include hooks and customization points for hardware-specific, or problem-specific, performance optimizations, without semantic implications
- Support a wide range of grids and meshes, while guaranteeing good performance

The first point allows users to reason about the behaviour of their massively parallel applications, which is often not intuitive. The second point can help to improve performance by introducing a few lines of codes, without impacting code correctness when moving to other platforms, or when changing the grid. The third point essentially means that the software addresses the needs of the target communities. To achieve those goals, technical expertise is necessary, but not enough. The developers of GHEX collaborates closely with the scientific partners in an interdisciplinary style to make sure that what they deliver is useful and usable.

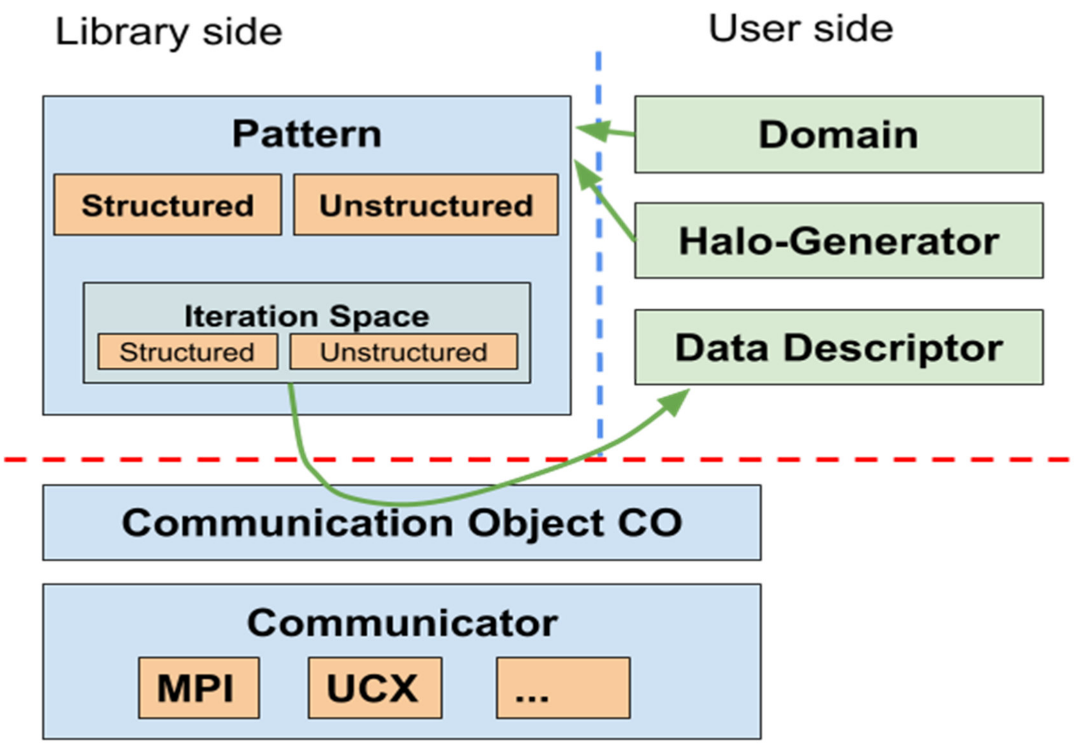


Figure 5: Overview of GHEX

The library will be developed in C++ using the technique of *generative programming.* In a nutshell, this technique is based on providing high-order functions to which to pass the functions (in jargon *functors*) to be executed to perform the actual operations. This technique is a proven technology for producing high-quality and overhead-free general software abstractions. Our group has a long standing experience with this technology. On the left side of Figure 5 there is a depiction of the software architecture that highlights the main library components, in blue, and the concepts from the applications needed for GHEX to function correctly.

A large part of existing scientific software, including weather, climate and astrophysics codes, are implemented in FORTRAN. This includes not only old legacy software, but also modern applications (e.g., DISPATCH). The GridTools project provides general facilities to bind the C++ components to FORTRAN codes, from which GHEX will directly benefit. In addition, Python has become the language of choice of many scientific communities, and there is a current effort, dubbed GT4Py, to provide a Python frontend to the GridTools main engine. GHEX will be a fundamental component to enable GridTools4Py to not just enable python programs to run natively on a node, but to be also deployed in a big parallel system. This is part of a larger effort in which many supercomputing centres around the world are currently engaged in to ease access to computing resources.

The GHEX interfaces will be designed to support the two major parallel programming paradigms: S*ingle Program Multiple Data (SPMD)* (multithreaded) and the Exascale oriented task-based parallelism. The collective halo exchange communication primitives will be asynchronous, and will avoid global synchronizations. GHEX functions will return future-like handlers to allow checking for completion in a later stage. This model follows the direction that the ISO C++ language is taking, and is employed by popular HPC libraries.

8.2 Project schedule

After the initial phase of the project we revised our schedule to reflect on the results of the initial design phase and the priority shifts in the participating institutions. We organise our project in 6 Tasks:

- T1: Requirements collection from users and designing/prototyping of solutions
- T2: Implementing halo-exchanges for regular grids
- T3: Implementing halo-exchanges for unstructured grids for different applications
- T4: Development of a thin point-to-point exchange layer with different transport layers
- T5: Benchmarking with tests and applications on different architectures
- T6: Integrating into production applications and frameworks and benchmark on early Exascale systems

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
T1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
T2				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨													
T3						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦						
T4						🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
T5					🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥				
T6																⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛

Table 6: Timeline of GHEX tasks

T1 is spanning the whole duration of the project since our development methodology is based on continuous integration and delivery, so collecting requirements, prototyping and refactoring is an integral part of the process. In this work-package the building system and the continuous integration and testing system will be developed and improved.

T2 will focus on regular grids, for which plenty of optimizations are possible, due to the fact that the halo regions can be described in very synthetic ways. We assume the work on regular grids will be completed around mid-project. T3 focuses on unstructured grids. Our first target is to interface with the Atlas library developed at the *European Centre for Medium-Range Weather Forecasts (ECMWF)* with which we are collaborating. Other meshing and libraries will be studied subsequently.

T4 aims at developing a common transport layer that is used to move data. This layer can be substituted without impacting the user code, and will be implemented on top of lower level communication libraries such as MPI, UCX, libfabric, or shared memory abstractions. This work-package will last until the end of the project to address problems in applications due to performance issues or correctness problems in the transport layer.

T5 is dedicated to benchmarking our solutions in both the low-level transport layer and the application layers for halo-exchanges. Keeping the implementation under pressure for performance is key to ensure a good quality of implementation.

Finally, T6 will focus on integrating GHEX into existing production applications and framework. The basic candidates are the COSMO model, the DISPATCH task based application, and GridTools4Py, a Python frontend to develop weather and climate models for the next generation computers. While prototype integration and benchmarking will be conducted before, in this work-package we will focus on production quality implementations and benchmark the full applications in order to guarantee the best performance.

8.3 Team and collaboration

The staff in the project has been hired. Two developers have been hired at the Swiss National Supercomputing Centre and working 100% on the project. One position is dedicated to this project at the University of Oslo. The team is now composed of

- Mauro Bianco, PI, CSCS
- Marcin Krotkiewski, proponent and main contributor, USIT
- Fabian Bösch, main contributor, CSCS
- Marco Bettiol, main contributor, CSCS

The collaboration uses a Slack channel in the GridTools organization, with access to all other members of it. The development is organised according to agile methodologies based on SCRUM, with adaptations for our particular needs. At the moment we organise the development in sprints of three weeks and have daily stand-up meetings on the slack channel. The work is divided into tasks, that are maintained in a backlog and scheduled before each sprint starts during a *sprint planning meeting*. We use Jira software to maintain and organise the backlog and track the progress of the tasks.

The code is shared on GitHub. Each member of the team has a fork of the main repository in order to avoid polluting it with many branches. In this way, individual team members are responsible for their own working copy and can manage them as they prefer. When updated can be merged into the main repository a *pull request* is made and the code is reviewed by another team member. To facilitate the code review process a document describing the code conventions has been developed and shared in a repository's wiki page. GitHub is also used to track issues in order to have a central place where we collect and control them.

Frequent video calls are needed to discuss requirements and results from benchmarks, and for this we typically use Skype or Zoom. We had two face-to-face meetings, one before the project started and one before the kick-off meeting in Bratislava. We organised another face-to-face meeting in Oslo in October 2019 to discuss the interfaces to the low-level transport layer and the current status of the design of the other components.

8.4 Status and outlook

The project started by dedicating the first three months to collecting requirements and test designs. Two main designs were developed and one was selected for the halo update interfaces. A regular grid implementation based on MPI is working on CPUs and GPUs and is currently being tested in the COSMO weather prediction model dynamical core, showing benefits in performance (9% overall gain) with respect to the previous implementation (Figure 6, which refers to a synthetic test case). This is due to the different scheduling of serializations and exchanges that take advantage of finer grain synchronizations and reduction in the number of pack/unpack kernels needed. Similarly, the integration with Atlas, the library for meshing, domain decomposition and halo-updates developed at ECMWF has started and show better scalability on CPUs for meshes on the sphere (Figure 7).

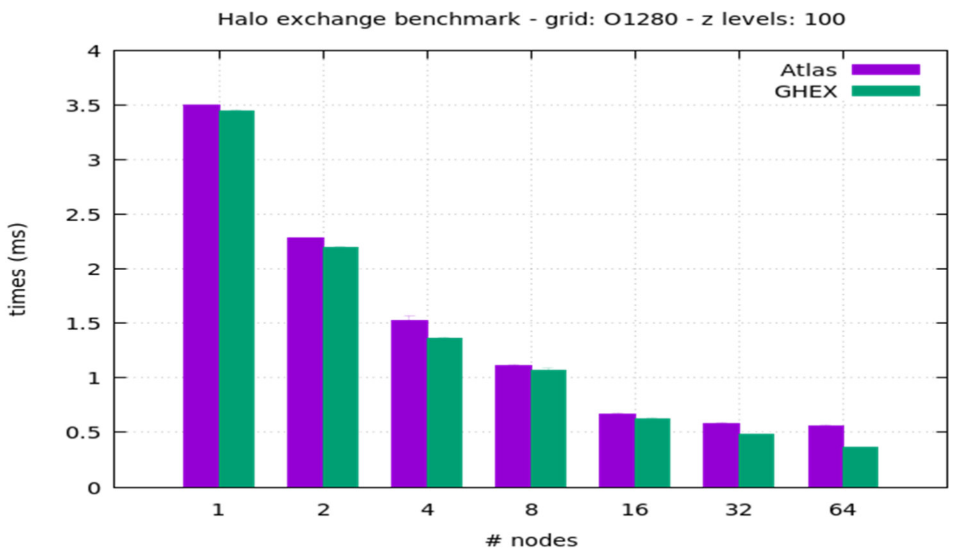


Figure 6: GHEX performance with Atlas library

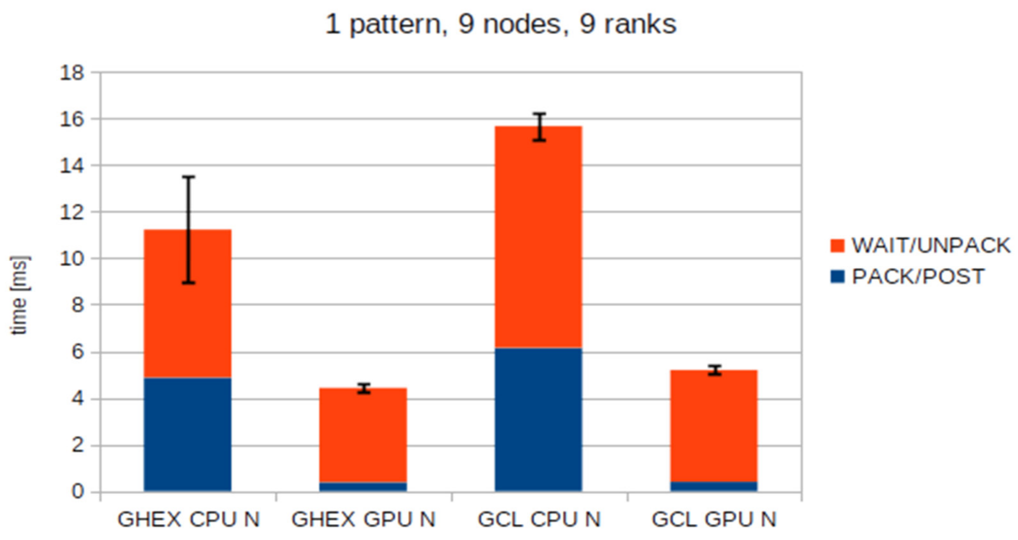


Figure 7: GHEX performance on standalone benchmarks for structured grids

Substantial work has been carried out to gather the requirements from DISPATCH and they are driving the development of the low-level transport layer to make sure it will suit both target application types: standard SPMD and the more loosely structured task-based codes, where the communication and computation performed by different tasks can be scheduled independently of each other. We prototyped an MPI backend, and a version of the transport layer that uses the *Unified Communication X (UCX)* communication framework. Initial experiences are encouraging, and in some cases we observe good improvement over the standard MPI approach. We are also integrating the low-level transport layer into the existing code that uses MPI directly.

In the next six months we will finalise the regular grids halo updates. We will discuss with our collaborators the opportunity to integrate GHEX into COSMO regional weather model in their production systems. We will also finalise the interface of GHEX to Atlas to run also on GPUs. We will have multiple low-level transport layers with a finalised API, and have a prototype implementation of DISPATCH to benchmark.

9 LyNcs: Linear Algebra, Krylov methods, and multi-grid API and library support for the discovery of new physics

9.1 Introduction and objectives of the project

The project **L**inear Algebra, Kr**y**lov methods, and multi-grid API and library support for the discovery of **n**ew physi**cs** (LyNcs), will pool together software development efforts across Europe to provide communities with the next generation of parallel libraries for solving sparse linear systems at the Exascale. LyNcs is led by the Computation-based Science and Technology Research Centre (CaSToRC) of The Cyprus Institute which will join forces with partners from the French Institute for Research in Computer Science and Automation (Inria) and the Leibniz Supercomputing center (LRZ). Within LyNcs, we will implement cutting-edge sparse linear solver algorithms, develop and prototype new Krylov and block Krylov solvers, and optimise existing parallel codes that implement a range of preconditioners for these solvers.

The improvements proposed span all levels of the scientific application software stack, from the basic Sparse BLAS library to fully-fledged simulation codes. In particular, we target the Fast Accurate Block Linear krylOv Solver (Fabulous) and its dependencies Starpu, Chameleon, Maphys, and Pastix, which we will enable for lattice Quantum Chromodynamics (QCD), for Computational Chemistry, and Computational Electromagnetics. Lattice QCD community solver libraries QUDA and DDalphaAMG will be further developed to implement and optimise Krylov and block Krylov solvers and new preconditioners. At the lowest level, we will further develop the efficient sparse matrix support software librsb, its APIs and adapter libraries, pursuing a tighter integration with the aforementioned packages. This library development and optimization will be accompanied by a coherent effort in designing, implementing, documenting, and maintaining an API, enabling various scientific user communities to build full-fledged scientific applications on top of these libraries.

LyNcs has two overarching objectives. The first is to consolidate and extend low-level advanced libraries that implement basic kernels and algorithms for the parallel solution of large numerical linear algebra problems, in particular by employing Krylov subspace methods suitable for Exascale. LyNcs recognises the urgent need to improve and further develop these libraries while adopting new frameworks to enhance their scalability. The second is to support user communities to use sparse linear solvers as building blocks of larger applications, such that they can address the next generation of challenges in their scientific fields, namely in lattice QCD, computational chemistry, and engineering applications in electromagnetism. These domains rely on access to leadership supercomputers such as via PRACE Tier-0 access, requiring the solution of huge sparse linear systems often involving poorly conditioned matrices. Adoption by user communities and maintainability relies on well thought-out, documented, and portable APIs with demonstrated examples of their use in established community codes. A recent example is the adoption of DDalphaAMG within tmLQCD, allowing the first simulation using a true multi-grid, crucial for allowing the simulation of QCD with quarks tuned to their physical masses on the PRACE Tier-0 system SuperMUC [3, 4]. LyNcs will generalise such development efforts, contributing to the development of multiple "low-level" solver libraries and targeting their adoption in several community codes enhancing synergies between leading European scientific groups.

Task 1 - Sparse linear solvers suitable for Exascale: This task will focus on optimizing existing, broadly used libraries to incorporate novel approaches either available in the literature or with demonstrated improvements in proof-of-concept applications. In particular, for Krylov subspace techniques, we will implement several methods that have exhibited improved scalability and performance. The core Krylov methods can be designed and implemented independently of two main kernels they intensively used, namely the linear operator application (usually a matrix-vector product) and the preconditioner application. Those two numerical kernels have to be specialised for the targeted applications and must be provided by the user of the library. For the Krylov library we will consider block solver variants that recycle spectral information either at restart or between solution phases. In particular, we will consider augmentation techniques, deflation techniques for Krylov block solvers [5]. The Krylov block solvers will also be implemented in their flexible counterparts [6,7], allowing for mixed arithmetic calculation, reducing communication volume, and consequently both time and energy-to-solution. These advanced linear system solvers share common kernels to those necessary for implementing eigensolvers. This synergy will be leveraged to investigate the robustness of these methods for eigensolvers. The outcome of this activity will be a new release of the Fabulous library.

As preconditioner for lattice QCD, multigrid implementations are available in both QUDA for GPUs and DDalphaAMG for CPUs. These methods have brought dramatic improvements in time-to-solution for solving sparse linear systems arising in lattice QCD with large condition numbers, by solving the system on an hierarchy of coarser grids as preconditioners. However, floating point performance on the coarse grids is much lower than what is achievable. We will implement block methods for bundling solutions of multiple right-hand sides to improve efficiency at the coarse level. In addition, we will implement a framework allowing distributing the coarse-level operator applications over sub-partitions of the process pool, to optimise the load-balancing of the various multi-grid levels. Providing such solutions is crucial if these methods are to scale beyond the state-of-the-art, which relies on a symmetric domain decomposition between coarse and fine grids.

For the solution of sparse linear systems, where sparse matrix-matrix product are involved as well as for some inner numerical kernels of the above mentioned methods, this task targets the fundamental building block of our solvers, that are the sparse matrix support routines. In LyNcs we will adhere to the Sparse BLAS API [8], which is vendor-independent. We will improve our existing Sparse BLAS implementation and provide a Sparse BLAS API-compliant layer to support routine libraries simplifying the validation of performance, correctness, and scalability properties. For the performance critical kernels, or for novel heterogeneous architectures, we will develop specialised support kernels accessible through the generic Sparse BLAS API. In this way we can develop, compare and experiment with novel data layouts, without the need of modifying our numerical algorithm specification.

Task 2 - Prototyping new methods for Exascale: This task will investigate cutting-edge, disruptive techniques that have not been considered so far but that are expected to be beneficial towards Exascale. Sparse linear solvers rely on an efficient implementation of the stencil operator that is a discrete representation of a linear operator. In parallel applications, each iteration requires communication of the stencil boundaries between processes, as well as global reductions to implement vector dot-products. Several techniques have been proposed to mitigate this scalability issue. One class involves relaxing the frequent communication and synchronization requirements. Another relies on parallel solutions on different chunks of right-hand sides running in parallel over smaller MPI partitions, mostly independently, but with infrequent exchange of information, such as the Krylov deflation spaces. Techniques for asynchronous composition of deflated spaces and block pipelined variants of Krylov solver techniques will be investigated within this task first with proof-of-concept implementations to assess their scalability features and improvements. Depending on the outcomes, we will selectively incorporate these methods in the libraries of Task 1. At the lowest-level, we will investigate new approaches to optimise our Sparse BLAS implementation, namely dense substructures support, refined cache blocking techniques, and adaptation of the precision on a block-specific basis.

Task 3 - Portable APIs: Within this task we will design, implement, and document APIs for calling the libraries developed in Tasks 1 and 2. We will design these APIs to be called for either GPU or CPU variants of the underlying libraries, so that scientific application developers can transition from GPU systems to CPU systems with minimal changes in their higher-level codes. Documentation and example use applications will be provided with the API library for demonstrating its use, how the library initialises, data structures used, and conventions.

Task 4 - Community code enabling: This task will provide direct support to the developers of main European community codes for interfacing their codes with our developed libraries through the API of Task 3. We have identified major European codes employing linear solvers, including HORSE for Electromagnetism, A-VCI for computational chemistry and for lattice QCD openQCD, which is used to produce the Coordinated Lattice Simulations datasets used by several collaborations including Mainz, RQCD, and Alpha, and the code tmLQCD, which is used by the European Twisted Mass Collaboration that spans 10 European countries. Developers and users of these codes are major PRACE allocations holders on SuperMUC, Marconi, Piz Daint, and JUWELS and have already expressed support for LyNcs.

9.2 Project schedule

In the Gantt chart of Table 7, we show the scheduling of the tasks including their subtasks, in intervals of three-month quarters. The timing of the Milestones is also shown, with details for each Milestone presented in Table 8. We successfully teamed up at the PRACE-6IP kick-off meeting in Bratislava. In the right part of Table 7 we also show the distribution of partner effort Person Months (PMs) towards the four tasks.

For the ongoing project we adapt the work schedule as follows:

- The work effort of T 1.3 is extended to Q1
- The end of Task 2 is shifted from Q6 to Q7 in order to soften possible delays
- Task 3/4 we plan to include test on the upcoming EuroHPC systems for (Q9-Q10)
- We plan to schedule F2F meeting within this year if all positions are filled (Q3)

	1	2	3	4	5	6	7	8	9	10
Task 1. Sparse linear solvers suitable for Exascale - leader CYI	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
1.1: Block solvers using augmentation/deflation techniques	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		
1.2: Specialised matrix-vector and multigrid for QCD		🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	
1.3: Sparse BLAS	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Task 2. Prototyping new methods for Exascale- leader Inria	🟥	🟥	🟥	🟥	🟥	🟥				
2.1: Asynchronous composition of deflated spaces		🟥	🟥	🟥	🟥	🟥	🟥			
2.2: Design of block pipelined variants			🟥	🟥	🟥	🟥	🟥			
2.3: Asynchronous communication in iterative solvers			🟥	🟥	🟥	🟥	🟥			
Task 3. Portable APIs - leader LRZ	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
3.1: API definition, consultation with community	🟩	🟩	🟩	🟩						
3.2: API implementation				🟩	🟩	🟩	🟩	🟩	🟩	
3.3: API documentation							🟩	🟩	🟩	🟩
Task 4. Community code enabling - leader CYI					🟧	🟧	🟧	🟧	🟧	🟧
4.1: Enabling lattice QCD codes (OpenQCD, tmLQCD)					🟧	🟧	🟧	🟧	🟧	🟧
4.2: Enabling computational electromagnetics codes (HORSE)					🟧	🟧	🟧	🟧	🟧	🟧
4.3: Enabling computational chemistry codes (A-VCI)					🟧	🟧	🟧	🟧	🟧	🟧
Milestones				MS1	MS2	MS3		MS4		
Face-to-face meetings	KO		F2F		F2F			F2F		

	CYI	INRIA	LRZ	Total
Task 1	16	11	12	**39**
Task 2	12	10	10	**32**
Task 3	12	3	6	**21**
Task 4	20	6	2	**28**

Table 7: LyNcs Timeline. Left: scheduling of tasks, subtasks, in units of three-month quarters (Q). Milestones and meetings are also shown, namely the Kick-Off meeting (KO) and face-to-face meetings (F2F). Right: PM distribution of partner effort to each task.

Milestone number, title, and month of completion	**Description, associated tasks, and means of verification**
MS1: First version of APIs available (M12)	A first version of the APIs will be made available, incorporating algorithms developed in **Tasks 1** and **2** and after consultation with the community within **Task 3**.
MS2: Prototype methods for Exascale evaluated (M15)	The evaluation of algorithms prototyped in **Task 2** will be finalised. The improved algorithms will be selected for implementing and optimizing within **Task 1**.
MS3: First reported improvements on Tier-0 system (M18)	First benchmark results on a PRACE Tier-0 showing improved scalability of the libraries developed in **Task 1** using the APIs developed in **Task 2**.
MS4: Community codes using LyNcs APIs (M24)	Community codes using the methods developed and implemented in **Tasks 1** and **2**, via the APIs of **Task 3**, after support from this project within **Task 4**.

Table 8: LyNcs Milestones (MS)

In Table 9 we list the PRACE 6IP deliverables as they appear in the proposal and detail the contribution to these deliverables by LyNcs.

PRACE 6IP deliverable	**Contribution by LyNcs**
D8.2: Interim progress report: staff and established project structure (M6)	Finalised set of tasks and milestones, PM distribution, and team, including any new staff.
D8.3: Interim progress report: Public Prototype software release and development infrastructure (SSC feedback) (M12)	First version of APIs from **Task 3**. Progress of library optimization (**Task 1**) and evaluation of prototypes new algorithms (**Task 2**)
D8.4: Interim progress report: Public Software release (docs, testing, issue tracker) and integration in external codes (SSC feedback) (M24)	First release of software, including documentation, benchmark results and issue tracking statistics (**Tasks 1**, and **3**). Report on evaluation of new methods of **Task 2**, and candidates selected for implementation within **Task 1**. Progress of integration into community codes (**Task 4**).
D8.5: Final report: including performance results on (pre)Exascale systems (M30)	Performance results on community software linked to APIs (**Tasks 1**, **2**, **3**, and **4**). Report on results presented in conferences. Publication of results of **Task 2** (as peer-review publication or PRACE white paper)

Table 9: LyNcs contributions to WP8 deliverables

9.3 Team and collaboration

CaSToRC filled two positions opened for the tasks related to LyNcs. Namely, Dr. Simone Bacchio was recruited and started in July 2019, while Dr. Shuhei Yamamoto started in September 2019. This enables CaSToRC to fully engage with the upcoming tasks of LyNcs and WP8 in general. CaSToRC is leading LyNcs, organizing the meetings, coordinating work, and scheduling telcons. Furthermore, CaSToRC members worked during the first month of the project on Task 1, Task 2, and Task 3. These efforts include an investigation of novel algorithms related to Task 1 and 2, and an investigating of different hybrid software approaches, like a python-based API, with respect to Task 3. For the upcoming months, CaSToRC will enable and optimise multiple right-hand-sides in DDalphaAMG. This will enable a close collaboration with Inria by investigate new algorithmic approaches using Inria's solver software Fabulous.

The open position at Inria is vacant due to a last minute cancelation of the recruited postdoc. Inria is currently advertising and searching to fill the open position as soon as possible to minimise delays in the work plan. The work is in the meantime progressing as planned, with Inria leading Task 2, currently working together with CaSToRC on investigations of communication strategies to efficiently share information accumulated in Block-Krylov procedures running on different partitions of the HPC system. Furthermore, Fabulous is currently further optimised to meet the challenges of novel computing architectures.

The position at LRZ is held by Dr. Michele Martone. LRZ is engaged with Task 1 and Task 3, and is leading the effort related to libsrb. After the successful release of an update of librsb, the next step will be the investigation of the potential of the sparse matrix library in user kernels, such as in lattice QCD, in collaboration with CaSToRC. A prototype version linked to a lattice QCD kernel is available and we expect to be able to report about its findings within the next reporting period.

During this initial period we initialised a common communication structure between the three partners. This includes an email-list hosted by LRZ where project partners are exchanging information and scheduling meetings. For software development, Inria provided a gitlab repository, which will be used to publish first and new releases of the developed API. Moreover efforts related to Task 2 are currently hosted by private Github accounts and will be published when results are available. We have regular monthly telcons, with the next upcoming call to be held during the first week of October. Furthermore, we plan to host a face2face meeting to coordinate work effort with the new recruited postdocs. We will schedule this meeting as soon as possible and once all positions are filled.

9.4 Status and outlook

LyNcs was successfully initiated during a meeting of all three partners during the PRACE-6IP kickoff meeting that took place from the 28 to 29 May 2019 in Bratislava. We determined the common communication channels and gave an update of the project timeline. Promising methods to explore were discussed and a plan was drafted on which key algorithms to focus on, which are essential to reach the project goals.

In detail, we determined that a multigrid procedure with multiple right hand sides, accelerated by using Block-Krylov methods, enabling deflation methods on the fly will be explored. During the past month, we collected benchmark results using QUDA running on NVIDIA GPUs in order to understand the potential of exact deflation in multigrid procedures. The results are very promising, although at the cost of an increased overhead – the setup time of the multigrid, which has increased. We plan to mitigate this additional time by using Block-Krylov methods to calculate the low lying spectrum directly during the iterations. This work is related to Task 1 and 2 and is currently in development by CaSToRC in collaboration with Inria. Implementation of communication avoiding Krylov-subspace methods remains open for investigation, to push further the limited strong scaling bounds of our Krylov solver.

Currently we intensified efforts in developing communication schemes to partition more flexibly the solver tasks, which will enable us to use efficiently large partitions of HPC systems without facing the strong scaling boundaries of our Krylov solvers. This will be directly included in the first API version of LyNcs which is targeted for MS12.

The LRZ effort of the first months of LyNcs was focused on a new release for the Sparse Matrix library librsb. The new release is available since August 2019 and is a maintenance release, easing build on different platform and with bug fixes. A first prototype is available linking librsb to DD-HMC, a predecessor software to openQCD with a common base. This prototype will help to understand the potential of librsb with respect to lattice QCD kernels. We expect to have benchmark results for Q2 which will help us to coordinate the Sparse Matrix library effort with respect to Task 3.

10 Conclusions

In this Deliverable we report about the status, the organisation and the structure of the eight projects approved in the process of selection described in the D8.1 deliverable. These projects cover a wide area of computational sciences ranging from numerical libraries to programming models and scientific domain applications. The main focus of this task is to deliver high quality software able to deal with the Exascale challenges. All the projects described in this document discuss how they intend to develop their software and achieve their targets. The organization reflects the 'agile team of teams' concept. For this reason, we are limiting the global communications only in case of necessity, for example when we need to establish a common milestone or delivering a document. To share experience and expertise, we are planning face-to-face events when we can globally discuss topics that can be of common interest to (almost) all the projects (for example task-based approaches, programming paradigms, etc.). For the remaining part, the management of each project is individually in charge to their PI.

In this document, each project reports about their internal organisation. In particular, after the context and the objectives of each project are briefly recalled, the tasks and the planning are shown. The information about the staff composing each project, their distribution among the partners and their expertise are reported. Also, the choice of the tools used in the projects to coordinate the work is discussed in each section. In many cases, regular video conferences have been chosen to check the progress of the work, while in other cases other tools, such as Slack have been preferred. Where present, the opportunity of collaboration with other projects external to PRACE was reported. In conclusion, all the projects described the activities conducted from the PRACE-6IP Kick-off Meeting until this moment and the planning of the activities for the next few months.

This Deliverable shows that all the selected projects in the Work Package 8 have set up their internal organisation with an accurate planning of the activities. This is an additional benefit of having required detailed proposals for the selection process. None of these projects is having staffing issues at the moment and the collaboration between the PRACE partners looks working satisfactorily well. All the projects have their activity on track and no criticalities appear at present.