Grant agreement no. 823712

CompBioMed2

Research and Innovation Action
H2020-INFRAEDI-2018-1
Topic: Centres of Excellence in computing applications

D4.2 Report on Maintenance and Development of CompBioMed Computational Services

Work Package:	WP4	
Due date of deliverable:	Month 25	
Actual submission date:	28 April 2022	
Start date of project:	01 October 2019	Duration: 48 months

Lead beneficiary for this deliverable: *SARA*
Contributors: *UEDIN, LRZ, UCL, UvA, USFD, UNIGE, UNIBO, BSC, USFD, UOXF, ACE, UPF*

Disclaimer

Project co-funded by the European Commission within the H2020 Programme (2014-2020)		
Dissemination Level		
PU	Public	YES
CO	Confidential, only for members of the consortium (including the Commission Services)	
CI	Classified, as referred to in Commission Decision 2001/844/EC	

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

Table of Contents

1 Version Log 3
2 Contributors 3
3 Definition and Acronyms 5
4 Public Summary 7
5 Introduction 7
6 Applications Services 9
6.1 HemeLB 9
6.2 Alya 12
6.3 Palabos 15
6.4 HemoCell 16
6.5 Binding Affinity Calculator 17
6.6 CT2S, ARF10 20
6.7 openBF 23
6.8 PlayMolecule 23
6.9 TorchMD 25
6.10 Virtual Assay 25
7 HPC and Data Services Infrastructures 26
7.1 Compute and Data Systems 26
7.1.1 CompBioMed HPC allocation programme 26
7.1.2 CompBioMed codesign systems 26
7.1.3 CompBioMed data storage infrastructures 27
7.2 Support Service for User Applications 27
7.2.1 CompBioMed applications benchmark and performance analysis tools 27
7.2.2 Middleware and workflows support 28
8 Support Services 30
8.1 Scalability Support Service 30
8.2 Internal Helpdesk 31
9 Risk Management 32
10 Conclusions 33
11 Annexes 35
11.1 Applications Services Supplemental Material 35
11.1.1 HemeLB 35
11.1.2 Palabos 36
11.1.3 PlayMolecule 37
11.2 QCG Performance on CompBioMed HPC Systems 37

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

1 Version Log

Version	Date	Released by	Nature of Change
V0.1	15/08/2021	SARA	First Draft
V0.5	04/10/2021	SARA	Draft submitted for internal review
V0.9	20/10/2021	SARA	Reviewers' comments integrated
V1.0	29/10/2021	UCL	Final Draft, submitted to the EC
V1.1	28/04/2022	UCL	Revised version with additional information on co-design activities for the applications – requested by the Commission

2 Contributors

Name	Institution	Role
Marco Verdicchio	SARA	Principal Author
Gavin Pringle	UEDIN	Contributor
David Wifling	LRZ	Contributor
Jon McCullough	UCL	Contributor
Max van der Kolk	UvA	Contributor
Ivan Benemerito	USFD	Contributor
Jonas Latt	UNIGE	Contributor
Antonino Amedeo La Mattina	UNIBO	Contributor
Adrià Qunitanas Corominas	BSC	Contributor
Xinshan Li	USFD	Contributor
Elisa Passini	UOXF	Contributor
Shunzhou Wan	UCL	Contributor
Alexander Wade	UCL	Contributor
Raimondas Galvelis	ACE	Contributor
Gianni De Fabritiis	UPF	Contributor
Okba Hamitou	BULL	Reviewer
Alfonso Santiago	BSC	Reviewer
Peter Coveney	UCL	Reviewer

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

Emily Lumley	UCL	Reviewer

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

3 Definition and Acronyms

Acronyms	Definitions
0/1/3D	0/1/3 Dimensional
AGPL	Affero General Public License
ALCF	Argonne Leadership Computing Facility
ARF0/10	Absolute Risk of Fracture after 0/10 years
BAC	Binding Affinity Calculator
BSC	Barcelona Supercomputing Centre
CoE	Centre of Excellence
CPU	Central Processing Unit
CSCS	Swiss National Supercomputing Centre
CT(2S)	Compute Tomography (2 Strength)
DICE	Data Infrastructure for Capacity for EOSC
DLB	Dynamic Load Balancing
DXA	Dual-energy X-ray Absorptiometry
EOSC	European Open Science Cloud
ESMACS	Enhanced Sampling of Molecular dynamics with Approximation of Continuum Solvent
EU	European Union
FAIR	Findable, Accessible, Interoperable, Reusable
FE	Finite Element
FPGA	Field-programmable Gate Array
GNN	Graph Neural Network
GPFS	General Parallel File System
GPGPU	General-Purpose Graphics Processing Unit
GPU	Graphical Processing Unit
HDF5	Hierarchical Data Format
HPC	High Performance Computing
HPC	High Performance Computing
I/O	Input/Output
IT4I	Information Technology for Innovations
ITIL	Information Technology Infrastructure Library

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

LRZ	Leibniz Supercomputing Centre
MD	Molecular Dynamics
MPI	Message Passing Interface
MTF	Mean Time to Failure
N-S	Navier-Stokes
NFS	Network File System
npFEM	Np-Finite Element Method
OLA	Operation Level Agreement
OLCF	Oak Ridge Leadership Computing Facility
PRACE	Partnership for Advanced Computing in Europe
PSNC	Poznan Supercomputing Centre
R&D	Research & Development
RAM	Random Access Memory
RDFaaS	Research Data Facility as a Service
SaaS	Software as a Service
SDK	Software Development Kit
ShARC	Sheffield Advanced Research Computer
SLA	Service Level Agreement
SM	streaming multiprocessor
SME	Small and medium-sized enterprise
TACC	Texas Advanced Computing Center
TIES	Thermodynamic Integration with Enhanced Sampling
Torch-CG	Torch-Coarse Grained
UCL	University College London
UEABS	Unified European Applications Benchmark Suite
UNIBO	University of Bologna
USA	United States of American
USFD	University of Sheffield
VM	Virtual Machine
VMD	Visual Molecular Dynamics
WP	Work Package

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

4 Public Summary

CompBioMed is a user-driven Centre of Excellence which serves users from academia, industry, and clinical practice in promoting and enhancing computational methods and simulation tools in the biomedical domain. The Centre of Excellence (CoE) is also working as contact point for our users' community by acting as network and incubator centre to promote application usage and integration, through specialised training and support activities. The impact of the CoE within the Computational Biomedicine user community is critically dependent on the ability of the CoE to translate the work done within the consortium partners into tangible and usable services which can help to bring value to the end users.

Since the inception of CompBioMed2, leveraging the simulation community that has grown out of CompBioMed1, we have built several core services which are maintained and offered to our userbase. These services have been designed to support the needs of the biomedical community in delivering state of the art solutions, access to training material and efficient use of high-performance and high-throughput infrastructures.

This deliverable describes the compute and data services developed and delivered within the project. The first part of the document focuses on the applications services available through the project partners, providing an overview of technical characteristics and infrastructure usage. In the second part of the document, we describe the tools and support activities we have been working on in these first 24 months of the project. The goal is to help our partners to deliver mature biomedical solutions and performance improvement, future proofing of biomedical applications, preparing strategies and best practice to exploit current multi-petaflop and emerging exascale resources.

5 Introduction

WP4 "Operation and Service" main objective is to coordinate and support the delivery of the computational and data-oriented services developed within the project. The WP coordinates the development of the project service portfolio and the service management system, which will serve to enrich the service offering with more community-driven biomedical solutions. The WP has also been designed to support the computational and data intensive applications and workflows within the project, by analysing the technical requirement of our user community and providing flexible and secure access to the appropriate compute and data infrastructures. In WP4 we optimised existing applications and Workflows, including provision of performance improvements to current applications which are not able to scale up to large HPC systems, parallelizing currently serial codes or optimise the resource usage for large scale workflows. In collaboration with WP2, WP3 and WP5 we are working to identify suitable applications that can be extended and ported to novel architectures to improve performances and user experience with focus in particular on exascale systems and hardware/software co-design principles.

In WP4 we work mainly in three areas:

- Service management: To support the development and delivery of compute and data services focused on the requirements of the biomedical community.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

- Applications support and performance enhancement: To provide support in both porting of application to new architectures and hardware (T4.6) as well as supporting the development of efficient algorithms to fully exploit HPC systems (T4.3, T4.2). The WP has provided support in adapting biomedical workflows to run at full scale on large systems (T4.4) and the production of benchmark and scalability results for the flagship applications developed within the project.

- Relation with EU initiatives coordination: To strengthen the connections with existing European e-infrastructures (e.g: PRACE, EOSC, etc.) and maintain the collaboration with other CoE and international projects.

We have endeavoured to bring together HPC experts and core developers to work on the optimisation of existing applications and workflows (evaluated in collaboration with WP2 and WP5), providing technical support on parallel architectures and algorithm design. This includes parallelizing current serial codes and enhancing the e-infrastructure usage of already parallel applications and the tools used to run large computational campaigns (e.g.: middleware, data management tools).

Within the scope of the project, we define a service as “a way to provide value to customers through bringing about results that they want to achieve.”

Already from phase 1, CompBioMed has continuously worked to provide value to end-users by offering different types of services to the biomedical community. Acting mainly as a federated service provider the Centre has supported computational biomedicine users with the expertise and resources coming from the consortium’s Core and Associate Partners.

The consortium provide access to several services developed and maintained within the project:

1. CompBioMed Training Portal
2. CompBioMed Scalability Support
3. CompBioMed Software Hub
4. CompBioMed Visitor Programme
5. CompBioMed HPC Allocations
6. CompBioMed Incubator Registry

These are available through the “Services” section of the CompBioMed website (www.compbiomed.eu). Since they have been extensively described in D4.1, this section will focus on the update we made within the CompBioMed Scalability support service, and on the newly created internal helpdesk, which was anticipated in the previous deliverable.

In this deliverable we focus on the CompBioMed compute and data services at the halfway point of the project’s lifetime. In section 6 we provide a technical description of the CompBioMed applications with details of their components, access mechanism used and e-infrastructure usage. These are the main components of the CompBioMed Software Hub, through which users can get to know and access the software developed and maintained by the consortium partners. Section 7 presents an overview of the services developed to support the applications service portfolio. The section focuses on the infrastructures access services provided within the project, the work done towards testing and benchmarking community codes and on new tools and working environments (i.e., middleware, resources provisioning self-service tools, virtual analytics workspaces and secure access management) to support our user community needs in

*This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No **823712**“*

terms of security, compute and data access. Section 8 describes the scope and structure of the scalability support service, through which we aim to provide technical support on parallel architectures and algorithm design, as well as help users in scaling sequential applications or improve the performances of existing parallel codes.

This deliverable complements the information on the CompBioMed compute services provided with WP2 deliverable D2.2 “Second Report on Fast Track Application Readiness”, where we presented the advancements in scientific results obtained using CompBioMed compute services, and the collection of resources developed to facilitate the adoption of the codes by external users.

A further view on the external user experience for the different compute services, as well as on codes usability, robustness and efficiency, will be presented in D5.2 “Advanced Report on Biomedical Applications” as part of the work done within WP5.

These three deliverables provide an all-around view on the different services and the plans we have in the project to continue supporting and delivering the computational solutions developed within the project.

6 Applications Services

The CompBioMed Software Hub addresses the needs of the computational biomedicine research community, which can use the Hub to access the applications services developed, aggregated and coordinated by CompBioMed. In this section, we describe the applications developed and maintained by the consortium partners at 24 months since the start of the project. Each section provides a description of the applications, the technical specifications for each of the codes, and an analysis of the main developments achieved so far in both infrastructure usage, performance optimisation and where feasible co-design efforts around these codes.

6.1 HemeLB

Application description

HemeLB is a 3D macroscopic blood flow simulation tool that has been specifically optimized to efficiently solve the large and sparse geometries characteristic of vascular geometries. It has been used to study flow in aneurysms, retinal networks, and drug delivery among many other cases. Target users will be trying to understand blood flow in complex vascular domains where 3D knowledge of the geometry is critical to solving their problem. HemeLB is open-source and available for free download from Github under the LGPL-3.0 License [1]. Build instructions are provided either from the repository itself (via the ReadMe file or using a provided build script) or via the HemeLB website. External users will need to compile the code themselves on their target machine. HemeLB has been successfully built on a wide variety of HPC systems including ARCHER2, SuperMUC-NG, Summit, and Blue Waters.

Technical specifications

HemeLB is written in C++ and uses MPI to distribute work amongst multiple processors. It has been designed to conduct a single run of a flow problem in a monolithic format. Aside from a compiler and MPI library, HemeLB comes packaged with the dependencies necessary for building and running the code. One dependency (CTemplate) currently requires a version of

*This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No **823712**“*

Python2 for successful compilation, however we are in the process of transferring this to Python3 as it becomes essential on different HPC platforms. Versions of HemeLB that are accelerated by GPUs are currently being developed. Those currently/soon-to-become available utilise CUDA to enable acceleration on NVIDIA GPUs. A port, using HIP to enable execution on AMD GPUs, has been conducted and is currently being tested. As a rule of thumb, input files and simulation data scale linearly with the number of sites being simulated. Execution may require available memory of around 2kB/lattice site.

HemeLB geometries can be developed using the tools available at https://github.com/UCL-CCS/HemePure_tools, whilst post-processing of results generally makes use of HemeXtract (https://github.com/UCL-CCS/hemeXtract) to convert the binary output files to a human readable format. These can then be viewed with scientific visualization software such as ParaView.

CompBioMed partners UCL and LRZ have worked on video [3] visualising the blood flow in the human forearm and how this was created with the HemeLB application. This work will be shown as part of the "Scientific Visualization & Data Analytics Showcase" [4] at this year's supercomputing conference, SC21, and has been shortlisted as one of five best scientific visualisations in the conference programme.

HPC usage and parallel performance

Simulation of blood flow in the full human vasculature is a challenging task for any computational approach. Although 1D solvers can do this with less computational resources than a 3D model it does not provide the same level of fundamental flow details and personalization of the flow domain. For the development of virtual human simulations, high fidelity 3D simulations – such as those generated by HemeLB – will be crucial to both medical and scientific understanding. The efficient generation of such simulations will make the use of HPC essential, this is the only way that flow of physiologically relevant timeframes can be created within a reasonable simulation time. At the full-human scale, where a high-resolution decomposition of the vasculature could require more than 50 billion lattice sites, such simulations can demand the resources of an exascale machine.

HemeLB manifests excellent strong scaling results on a number of HPC platforms. Here we highlight the performance of the CPU version of the code on SuperMUC-NG (LRZ, Germany) and the NVIDIA GPU version on Summit (USA) – machines currently at #17 and #2, respectively, on the Top500 list [2]. To compare GPU and CPU results, we convert the number of GPUs to CPU equivalent by using the number of streaming multiprocessors available on the chip – this is the methodology used by the Top500 list. In Annex 11.1.1 we present our current observed strong scaling behaviour with HemeLB. These have been obtained using a 10 billion site circle of Willis domain that represents the complexity of domain used in production jobs. The data in Annex 11.1.1 has been scaled to cores for the CPU code and SMs (i.e., CPU core equivalent) for the GPU data. From these plots, it can be seen that HemeLB demonstrates very good strong scaling behaviour to the full production partition of SuperMUC-NG (approx. 310,000 CPU cores, the fastest CPU only machine in Europe). On Summit, the use of up to 20,000 GPUs allows this to be extended up to approx. 1.5 million SMs. In further testing, we have been able to extend execution with acceptable performance up to 2 million SMs. These last two figures demonstrate that HemeLB is already able to effectively run at around the maximum day-to-day usage levels expected of an exascale machine.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

Within a collaboration with FocusCOE (https://www.hpccoe.eu/) UCL has also developed a dedicated training website [5] for users to get to support with the deployment and benchmarking of HemeLB. The training is designed with the purpose of acquiring performance data and scaling information on emerging exascale architectures.

HemeLB uses MPI in a master-worker format to manage communications across the assigned resources. Here a single rank is dedicated to managing the simulation whilst the remainder are devoted to conducting the computation throughout the domain. Communication between ranks occurs when a lattice site's neighbours are being computed on a separate rank. MPI communications have been organized to mask this as much as possible. Data input all occurs within the initialization phase of the simulation, this can correspond to a significant quantity of data – approximately 1.3kB/site. Data output occurs at a frequency indicated by the user in the input file, this output will typically require 64 bytes/site (though variable dependent on actual output requested) and this is stored in a compressed binary format. Checkpoint files can also be saved in this manner if required. For the single component version of HemeLB discussed here, no temporary files are written to aid simulation completion.

Within CompBioMed2, we have achieved a number of technical developments for the HemeLB code and are continuing to work towards several more. The port to NVIDIA GPUs is the primary exemplar of this, resulting in the strong scaling results presented above. Efforts within CompBioMed2 have also developed a port for AMD GPUs and the performance of this is currently being evaluated. Whilst platform-agnostic programming is an ideal goal, it will take time before a consensus is reached on how this is to be best achieved for GPUs (if possible). Further optimization work (current and ongoing) with LRZ has been focused on improving the performance of HemeLB on SuperMUC-NG. This has included an update of the intrinsics used to accelerate key computational kernels within the code – this effort has demonstrated improvements to the parallel efficiency of the code and we are currently working to evaluate the large-scale performance of these. Ongoing work will further look at how memory management could be optimized for this machine.

As part of the work in CompBioMed, EPCC have developed a service called the HemeLB High Performance Offload service (Hoff, https://github.com/EPCCed/hemelb-hoff) which allows to offloads HemeLB simulations to a remote HPC system, using a set of REST endpoints for submitting and managing computational jobs on remote resources. With the Hoff, existing Portals or existing SaaS offerings are able to upload its input files to a server connected to the HPC system where the simulation will be executed. After copying the input data sets onto the target HPC system, the Hoff schedule and execute the simulations, and, once completed, notify the user and allow for retrieve of the output data. The tool is currently interfaced between a portal, PolNet, which extents the functionality of HemeLB, with the Cirrus HPC system, and EPCC is currently working to extend its functionalities to work with the SLURM scheduling system.

Co-design activities

The pending arrival of AMD GPUs on the pre- exascale LUMI (Finland) and exascale Frontier (USA) herald the importance of different hardware manufacturers entering the HPC GPU market. Looking further ahead, the SuperMUC-NG Phase 2 expansion will deploy a set of Intel Ponte Vecchio GPUs. To take advantage of these new hardware platforms will require the conversion of existing CUDA codes to a more platform agnostic setting. We have an ongoing partnership with AMD through Atos and SURF to work on the co-design of HemeLB which led, so far, to the successful porting of this application on multiple (up to 16) AMD MI50 and MI100 GPU accelerators. Regular meetings with Atos experts, AMD GPU experts and UCL code

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

developers enabled the initial porting of HemeLB. Future work will consist of analysing the comparative performance of this implementation and identifying its limitations. Through this partnership we aim to work with AMD on the test cluster deployed at SURF to analyse and optimise the current performance on both MI50 and MI100 as well as extend the tests to the new MI200 AMD GPU cards (currently not available in production HPC systems).

HemeLB is also one of the target applications which we will use to evaluate the use of Intel's OneAPI toolkit for the heterogenous development of codes across CPUs and GPUs. This will be of particular interest as CompBioMed applications may be flagged for performance evaluation on LRZ's SuperMUC-NG Phase 2 that is planned to deploy Intel's Ponte Vecchio GPUs.

In collaboration with ARM, UCL is also looking at the performance and optimisation of HemeLB on future ARM hardware. This blood flow simulation tool has attracted attention because of its impressive scaling performance on existing CPU architectures. Whilst the landscape for HPC simulation workflows is diversifying, monolithic jobs conducted on large quantities of hardware will remain the main mechanism for many users. We plan to look at whether future ARM hardware can achieve improvements in the speed and scalability of the HemeLB code. It is foreseen that this work will be conducted in combination with EPI and may serve as an exemplar for large-scale monolithic simulations on these new platforms.

Further co-design efforts are currently in the planning stages for the possible use of HemeLB as benchmark code for testing, evaluating and commissioning new hardware such as NVIDIA's GPU hardware development kits (at UCL), SuperMUC-NG Phase 2 (at LRZ) and the upcoming European Processor (with ATOS).

Cited references:

[1] https://github.com/hemelb-codes/hemelb/blob/main/LICENSE
[2] https://www.top500.org/lists/top500/list/2021/06/
[3] https://youtu.be/Sd8_cujMP_4
[4] https://sc21.supercomputing.org/program/posters/scientific-visualization-data-analytics-showcase/
[5] https://hemelb-dev.github.io/HemeLB-Carpentries/

6.2 Alya

Application description

Alya aims to simulate complex multi-physics / multi-scale coupled problems at tissue, organ, and system level. The idea is that of a "Virtual Patient", in which we can model the patient in health, in disease and under treatment. In the context of CompBioMed2, we are focused on the cardiovascular system.

So far, the code is being used by colleagues from academia. In its raw version and without any kind of graphical user interface, the code is not friendly and due to the complexity of the problems we are dealing with, only an expert user can setup, run, and analyse the simulations. In 2018 we created the spinoff ELEM Biotech (an Associate Partner of CompBioMed) to speed up the technology transfer to biomedical stakeholders and one of its missions is making the code more usable by non-expert users, including a cloud deployment and an ad-hoc biomedical user interface. Thanks to ELEM action, we are extending the access to Alya to users with biomedical knowledge, such as engineers from device manufacturing or pharmaceutical companies.

The code has different types of licences, depending on the case:

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

- Non-commercial available source
- Non-commercial cloud SaaS
- Open source for a limited version (only fluid mechanics)

Technical specifications

The code is written in modern Fortran language, with some user tools in Python and Perl. The memory is dynamically allocated and strongly depends on the problem solved. The code has no external third-party high-level library dependency, the only dependencies are of low-level type, such as MPI. The code has been compiled in several platforms such as ARM, Intel, NVIDIA or IBM Power processors.

The parallelization strategy is a hybrid one, MPI+OpenMP. Part of the code has been ported to GPUs to offload solvers or matrix assembly. The code has examples of use as monolithic, coupled and in ensembles and for replica computing. Being portable and efficient in both large and small architectures, it has records on different running scenarios.

The code relies mostly on internal developed tools. There is no need for containers or workflow managers. Several pre-processing tools have been used in combination with Alya: Ansa, IcemCFD, GMesh or GiD. As post-processing tools, typical used applications include ParaView, VisIt, Ensight or GiD.

HPC usage and parallel performance

The code is used in complex very large, coupled problems and it is designed and continuously improved to run efficiently in parallel for coupled problems. If not carefully done, coupling can strongly punish performance: a code can be very efficient solving a fluid or a solid mechanics problem, but this does not preclude that a fluid-solid interaction case could become extremely inefficient. We use a multiple instance point-to-point MPI communication strategy for coupled problems, allowing us to run the individual instances in different architectures with a heavy communication pattern. We also combine this with load balancing schemes by linking Alya to the BSC's DLB tool for dynamic load balance. For all this, the use of HPC combined with an efficient yet very advanced programming and running strategy becomes important.

The strong scalability of Alya has been established in PRACE, Alya being part of the Unified European Applications Benchmark Suite (UEABS), as well as through continuous testing on the main supercomputers in Europe and the US. The continuous development and improvement of our HPC capabilities have been carried on since the early days of the Alya Development Team, in 2005, through several collaboration projects. Since then, we have focused on all the aspects of the problem, and lately with the exascale horizon in sight. Our current strategy is a *tick-tock model*, in which we dedicate alternative efforts, on one hand, efficiency at system level in large core counts (weak/strong scalability, communication bottlenecks, partition algorithms, system level load balance, etc.), and, on the other hand, efficiency at node level (vectorization, hybrid models, porting to new architectures, co-execution, node level load balance, etc.). Of course, on the road to exascale all levels are accountable. All phases, when completed, contribute to a general strategy. On top of that, for a few years, we have been working on combining Machine Learning with modelling and simulation, all in the context of High Performance Computing and exascale. At this moment, according to our long-term development plan, we are finishing one of our node level efficiency phases.

As mentioned above, the CompBioMed2 applications which Alya is tackling are all large and tightly coupled. Alya consists of many different modules, each accounting for different physics

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

and solving for a particular set of equations. Some of them, including the one of interest in CoE RAISE (https://www.coe-raise.eu/), have been ported to GPGPUs using OpenACC pragmas. In addition, a co-execution model has been developed to enhance the resources usage. The idea of co-execution is to have the same computational kernels running at the same time on both CPUs and GPGPUs and therefore to fully exploit hardware heterogeneity [1,2].

Alya is general purpose engineering application used and developed within several CoEs working in different scientific domains. Since its conception in 2004, Alya has being designed and extended using advanced High Performance Computing programming techniques to solve coupled problems on supercomputers efficiently, making it one of the most promising European scientific applications which is expected to be taken to the exascale.

Co-design activities

Several co-design activities have been focusing around Alya in dedicated EU projects such as the European Processor Initiative (EPI [3]), and the Excellerat CoE [4] where the code is one of the main supported applications. From a co-design perspective, Alya can indeed efficiently exploit both latency-oriented CPUs and throughput-oriented GPUs. Those are the two parallelization models required on the upcoming pre-Exascale EU supercomputers. The portability to GPUs has implications both in the programming language and on the data structuring. It is being carried progressively in the distinct modules of Alya following a common strategy. In particular, the Navier-Stokes (N-S) solver, critically used in many biomedical applications, was completely ported to GPUs and run on a POWER9+NVIDIA system [1] and we aim to port all the modules involved in multiphase combustion simulations to GPUs within The Center of Excellence in Combustion (CoEC, [5]). Some of the kernels of Alya have also been ported to FPGAs in the context of the LEGaTO European project [6].

Additionally, it is worth remarking that partner BSC is leading the MareNostrum Exascale Experimental Project (MEEP, [7]) which is a flexible FPGA-based emulation platform that will explore hardware/software co-design for Exascale Supercomputers and other hardware targets, based on European-developed IP (Intellectual Property). Within this project, BSC is working to foresee the behaviour of Alya's computational kernels in FPGA-based emulators of future exascale computers, especially with those architectures proposed by the EPI initiative.

The BSC Alya Dev Team is also collaborating with the PoP Centre of Excellence and in the context of the EPI at all levels. Notably, we are creating mini apps from different parallel parts of Alya to study code optimization for the new vectorial RISC5 architecture in a BSC emulator running in FPGAs. We analyse code-reengineering to expose vectorization with long vectors (>256). At this moment we are waiting for the Fortran compiler developed by FLANG, of the LLVM Compiler Infrastructure [8], being due the first quarter of 2022.

Cited references:

[1] R. Borrell, D. Dosimont, M. Garcia-Gasulla, G. Houzeaux, O. Lehmkuhl, V. Mehta, H. Owen, M. Vázquez, and G. Oyarzun. Heterogeneous CPU/GPU co-execution of CFD simulations on the POWER9 architecture: Application to airplane aerodynamics. Future Generation Computer Systems, 107:31-48, 2020.
[2] G. Oyarzun, M. Avila, R. Borrell, G. Houzeaux, O. Lehmkuhl, H. Owen. Explicit/Implicit Navier-Stokes Solver on CPU/GPU heterogeneous Supercomputers, PARCFD2020, Paris, May 2021.
[3] https://www.european-processor-initiative.eu/

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

[4] https://www.excellerat.eu/
[5] https://coec-project.eu/
[6] https://legato-project.eu/
[7] https://meep-project.eu/
[8] https://releases.llvm.org/11.0.0/tools/flang/docs/ReleaseNotes.html

6.3 Palabos

Application description

Palabos is a general software library for computational fluid dynamics using the lattice Boltzmann method. In biomedical research, the main domains of applications are cardiovascular, including the simulation of blood flow in arteries, the investigation of the effect of medical devices such as stents, and cell-level blood simulations to investigate fundamental blood properties. Principal target users are researchers in the biomedical domain and R&D developing new medical devices.

The code is open-source and is distributed under the terms of the software license AGPLv3. It can be downloaded from the Web site www.palabos.unige.ch and is straightforward to install, as it depends on no external library (except for an implementation of the MPI communication layer).

Technical specifications

The Palabos library is written fully in C++ and runs on CPU clusters. It is parallelized with MPI for multi-core CPUs and for CPU clusters, using one MPI task per core (single-run parallelism). Typical memory requirements are 2 GB per core. Pre- and post-processing capabilities are built-in to the Palabos software library. A recommended data visualization tool for Palabos output is the ParaView software.

HPC usage and parallel performance

The need for HPC is particularly motivated in the field of cell-level blood simulations. Here, an exascale or pre-exascale system is required to simulate a macroscopically significant number of blood cells (of the order of 1 billion cells) while maintaining a detailed description of the physics of each cell. Cell-level simulations are achieved with the Palabos-npFEM coupling. A multi-scale description, i.e., simulation of a large blood volume (a section of an artery) while maintaining the cell-level description, is achieved in the limit of weak scaling, as the blood volume (and the total number of blood cells) scales with the number of available nodes.

The figure in Annex 11.1.2 reports the scaling behaviour of the Palabos-npFEM code, as it simulates a coupled fluid – Red Blood Cell system on a heterogeneous platform (fluid on CPU, blood cells on GPU). The graph shows weak scaling, and the size of the simulated blood volume (in micro-meters) is indicated for each data point.

Current research focuses on scaling the Palabos-npFEM code to a larger number of nodes, and on porting the full Palabos library to GPU for more efficient large-scale simulations of arterial flows. Porting of the Palabos library to single GPU has been achieved. Current and future efforts focus on multi-GPU performance of the full Palabos library.

Co-design activities

The GPU port of Palabos was achieved through co-design exchanges with NVIDIA technology teams through the participation at a GPU hackathon (Eurohack21 in September 2021, organized

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

by the Swiss supercomputing centre CSCS) and subsequent, ongoing communications which have led, on one hand, to subsequent performance improvements of Palabos on NVIDIA GPU clusters, and to feedback of UNIGE to NVIDIA on the usability and performance metrics of the NVIDIA implementation of C++ standard language parallelism. This allowed the Palabos team to make suggestions for extensions to the NVIDIA HPC SDK that would allow HPC codes like Palabos to be ported to an exascale capable machine more efficiently. Furthermore, in collaboration with NVIDIA technology teams, a performance model for the multi-GPU execution of the Palabos software library is set up, taking into account multiple aspects of the parallel multi-GPU platform, allowing extrapolation of currently measured performance to future systems.

6.4 HemoCell

Application description

HemoCell is a parallel computing framework for simulation of dense deformable capsule suspensions, with special emphasis on blood flows and blood related vesicles (cells). The library implements validated mechanical models for red blood cells and can reproduce emergent transport characteristics of such complex cellular systems. HemoCell can handle large simulation domains and high shear-rate flows providing a virtual environment to evaluate a wide palette of microfluidic scenarios.

The code targets academic and industrial users with an interest in simulating dense cellular suspensions, where the users typically have a special interest in blood-related flows. The users' applications can include a broad range of applications, e.g., understanding experimental observations using numerical simulation, validation based on experimental data, or investigating numerical aspects of HemoCell, such as performance analysis, scalability, and load-balancing.

The source code of HemoCell is provided as an open-source library (https://github.com/UvaCsl/HemoCell) under the AGPL license. The source code comes with a wide range of illustrative examples and corresponding documentation. HemoCell is easy to compile and runs on a variety of HPC systems (e.g., Cartesius (SURF), SuperMUC (LRZ), MareNostrum (BSC), etc.), where helper scripts are included to define the needed compilation environments. So far, HemoCell is not yet provided by default on any HPC system.

Technical specifications

HemoCell is implemented in C/C++ with parallelism achieved through MPI. The library provides the cellular mechanics and cell transport on top of the fluid simulations using the lattice Boltzmann methods implemented by the underlying Palabos library (also written in C/C++ and publicly available). Currently, the implementation only exploits CPU nodes on HPC systems, i.e., no accelerated hardware is (yet) used. Typically, simulations run with a single MPI thread per core available in the allocated job on the HPC system.

Parallelism is then managed through a decomposition of the simulated domain. The underlying fluid field is decomposed in blocks of (nearly) identical size. Each of these so-called *atomic blocks* are distributed across the available MPI threads, where typical communication patterns are implemented to exchange data at the interfaces. The suspended cells are distributed based on their location to the corresponding atomic blocks. As the simulation progresses, cells move through the domain and need to be redistributed to different MPI threads, potentially causing a load imbalance if cells accumulate in parts of the domain. To counteract such imbalance,

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

HemoCell provides methods to dynamically resize the atomic blocks in addition to the original static domain decomposition.

HemoCell has few dependencies and to compile the library a user only requires a C/C++ compiler, MPI libraries, and a build system (e.g. CMake and Make). After compilation, the user can link with HDF5 to enable output of raw data. For pre-processing, HemoCell comes bundled with a cell packer that generates initial red blood cell positionings for various simulation domain. For post-processing, Python scripts are available enabling visualisation using ParaView or rendering through Blender.

HPC usage and parallel performance

The need for HPC is driven by the desire to simulate ever increasing blood volumes. This requires larger simulation domains as well as larger embedded vesicle counts. Additionally, to recreate complete experimental setups, the simulations need to simulate longer time frames. This combination requires large domains, many cells, and many iterations, requiring modern HPC systems to find solutions in practical run times.

Ongoing research aims to investigate and improve the parallel performance of HemoCell. Current measurements show reasonable scaling performance of both fluid and cellular parts with an efficiency of about 70% (as tested on Cartesius up to 512 nodes (x24 cores)). In the coming years, we aim to increase this scaling efficiency and reduce the overall runtime by providing support for accelerated hardware (GPU). Simultaneously, advanced load-balancing methods are being developed to maintain uniform load on heterogenous hardware and continuously changing cell distributions. Furthermore, we aim to extend HemoCell's features to align with future experiments by including a variety of new boundary conditions and cell behaviours.

Multiple numerical examples are provided that can act as a "proxy" problem. These examples have known outcome and can be used as benchmark references when implementing support for new hardware, load-balancing, or parallelisation strategies. Moreover, a dedicated example is provided that is used exclusively for performance measurements (weak and strong scaling), which can be recreated by users on a variety of HPC environments.

6.5 Binding Affinity Calculator

Application description

BAC or Binding Affinity Calculator encompasses two highly related methodologies. These two methodologies are Thermodynamic Integration with Enhanced Sampling (TIES) [1] and Enhanced Sampling of Molecular dynamics with Approximation of Continuum Solvent (ESMACS) [2]. Centrally both these methods make use of ensembles of molecular dynamics (MD) simulations to control for the inherently chaotic nature of the trajectories MD produces. The trajectories collected can be used to calculate the binding affinity of ligand and protein. These affinities are calculated as a free energy of binding which is a critical quantity relevant in drug design and personalized medicine. The free energy difference between two states is an important quantity in biology, because it determines the ensemble at equilibrium. The binding free energy of an inhibitor often correlates to its effectiveness. As such the accurate prediction of binding free energies has been a longstanding goal of computational methods [3]. The two protocols are complementary in that ESMACS is an absolute free energy method for the ranking of binding affinities for highly diverse compounds, whereas TIES is a relative free energy method

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

for the estimation of free energy differences for pairs of similar (congeneric) compounds and/or mutated protein sequences.

TIES aims to alleviate many of the bottlenecks in relative binding free energy calculations however, due to the complexity and large computational cost of running these calculations TIES is targeted at expert users experienced in alchemical free energy calculation. We provide tutorials for how to run calculations using TIES via our online documentation for the project. These tutorials should make it easier for less expert users to apply these methods however the computation cost barriers remain high.

Technical specifications

TIES is written in Python and as such is relatively hardware agnostic. The Python code will call external programs such as AMBERTools [6] or OpenMM and these codes are less hardware agnostic. The external programs which TIES uses can be interchanged for example the MD engine OpenMM can be swapped for NAMD and this allows TIES the flexibility to exploit Intel/AMD/IBM CPUs as well as NVIDIA/AMD GPUs. The main dependencies of TIES are then these external MD engines previously mentioned as OpenMM and NAMD.

Typically, the resulting trajectories from a TIES or ESMACS calculation would be inspected using visualization software such as PyMol [7] or VMD [8]. No other external tools are strictly necessary to run TIES. For ESMACS, AmberTools [6] is required for the post-processing analysis to extract the parameters of interest.

HPC usage and parallel performance

TIES and ESMACS have been deployed on numerous HPC systems including Summit OLCF, Theta GPU ALCF and Longhorn TACC. TIES and ESMACS use a large ensemble of runs to distribute the computation to a large number of CPU cores or GPUs with no communication between individual replicas in the ensemble. A caveat to this is that a one CPU based instance of TIES/ESMACS using NAMD could itself be run in parallel using a hybrid OpenMP and MPI approach however, this relies on the parallel performance of NAMD which is outside the scope of this project. Linear scaling has been measured up to 240 GPUs on the Summit HPC system. Testing beyond this number of GPUs was not performed.

HPC infrastructures are critical to the application of TIES and ESMACS. A calculation of one binding affinity with TIES would typically use 120 v100 NVIDIA GPU for a wall time of 1-3 hours and one study may perform the calculation of tens to hundreds of binding affinities. ESMACS is typically performed with an ensemble of 25 replicas, which uses 25 or 50 compute nodes on HPC resources for a wall clock time of 3-6 hours, depending on the HPC architecture and the size of molecular systems. As such TIES/ESMACS can almost exclusively be applied in the large-scale supercomputers. The motivation for running calculations in this massively parallel way is twofold. First, many replicas of the simulation must be run to appropriately control for the aleatoric error in the MD trajectories. Secondly, binding affinities must be calculated in a timely fashion, on the order of hours, to be actionable in a clinical domain.

The exploitation of alternate hardware and or the more efficient use of hardware via optimization would be provided by the ongoing improvements to the MD engines which underpin the TIES and ESMACS methodologies. These improvements would most likely be made by the respective developers of the MD engines. Joint efforts with NAMD or OpenMM developers to improve the performance of binding free energy calculation would yield performance improvements for TIES and ESMACS.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

Co-design activities

BAC has been an extremely valuable tool for different codesign activities within the project. The recent developments of the BAC workflow, represent the joint co-design efforts of arm, Atos, UCL and Oxford University in the domain of genomics. Our co-design efforts here aimed to extend the activities carried out so far on the influence of sequence on binding affinities through a direct collaboration with arm and the European Processor Initiative. To achieve this aim, we have used the novel amplicon sequencing program, named Viridian, to detect mutations in the SARS-CoV-2 genome which can provide insight into the mechanistic impact of genetic mutations on antibody evasion as well as allowing for the spread and change in variants of interest to be tracked over the globe.

We have investigated the performance of Viridian on new hardware provided by arm. This hardware consists of an Ampere Altra E252-P30 2U Mt Snow server with 80 cores based on the arm Neoverse N1 CPU core design. Comparisons have been made with comparable x86 hardware, specifically Intel 8358 (2 sockets, 32 cores per socket) and AMD 7713 (2 sockets, 64 cores per socket). Our results show the Altra to have performance parity with Intel 8353 on the sequencing of one SARS-CoV-2 genome taking approximately 2 minutes on both platforms. Both the Altra and Intel 8358 ran slightly faster than the AMD 7713. Additionally, in collaboration with Viridian developers, we have investigated several avenues for optimization and achieved 2x speed up for the SARS-CoV-2 application of all platforms. This work so far has combined expertise from many sources with Atos and arm focused on hardware and software optimizations and UCL/Oxford focusing on scientific applications. This collaboration has permitted the optimization of both software and hardware in this co-design effort to meet the demands of impactful and cutting-edge genomics workflows, workflows which are already deployed in clinical settings by collaborators at Oxford.

In addition to these co-design efforts with arm, UCL has recently acquired an arm/Nvidia based development kit. This machine is also built on arm's new Neoverse-N1 CPU core design but has the addition of 2 top specification 80GB Nvidia A100 GPUs. This will allow us to continue our development efforts to support a wider range of hardware solutions, such as arm CPUs, but also to expand this effort to now optimise our codes with GPU acceleration (HemeLB/BAC) for arm/Nvidia platforms.

Within the development efforts around BAC, we recently started a collaboration with Intel to investigate and examine the use of Intel's OneAPI or the generation of vendor agnostic GPU applications for applications including HemeLB and BAC. This collaboration will focus on the optimisation of biomedical workflows such as BAC, on Intel's Sapphire Rapids CPUs and Ponte Vecchio GPUs. In collaboration with Intel's expert, which will also provide access to development clusters, we will work to deploy, benchmark, and investigate the scalability of ML/HPC coupled workflows as well as on the production of best practice guidelines for running efficiently on Intel hardware.

Cited references:

[1] Bhati, A.P., Wan, S., Wright, D.W. and Coveney, P.V., 2017. Rapid, accurate, precise, and reliable relative free energy prediction using ensemble based thermodynamic integration. *Journal of chemical theory and computation*, *13*(1), pp.210-222.

[2] Wright, D.W., Wan, S., Meyer, C., Van Vlijmen, H., Tresadern, G. and Coveney, P.V., 2019. Application of ESMACS binding free energy protocols to diverse datasets: Bromodomain-containing protein 4. *Scientific reports*, *9*(1), pp.1-15.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

[3] Stroet, M., Koziara, K.B., Malde, A.K. and Mark, A.E., 2017. Optimization of empirical force fields by parameter space mapping: A single-step perturbation approach. *Journal of chemical theory and computation*, *13*(12), pp.6201-6212.
[4] Eastman, P., Swails, J., Chodera, J.D., McGibbon, R.T., Zhao, Y., Beauchamp, K.A., Wang, L.P., Simmonett, A.C., Harrigan, M.P., Stern, C.D. and Wiewiora, R.P., 2017. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, *13*(7), p.e1005659.
[5] Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kale, L. and Schulten, K., 2005. Scalable molecular dynamics with NAMD. *Journal of computational chemistry*, *26*(16), pp.1781-1802.
[6] Salomon-Ferrer, R., Case, D.A. and Walker, R.C., 2013. An overview of the Amber biomolecular simulation package. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *3*(2), pp.198-210.
[7] DeLano, W.L., 2002. Pymol: An open-source molecular graphics tool. *CCP4 Newsletter on protein crystallography*, *40*(1), pp.82-92
[8] Humphrey, W., Dalke, A. and Schulten, K., 1996. VMD: visual molecular dynamics. *Journal of molecular graphics*, *14*(1), pp.33-38.

6.6 CT2S, ARF10

Application description

The Computed Tomography to Strength (CT2S) workflow is a digital twin solution, used for the prediction of the risk of hip fracture for an individual based on CT scans. CT2S is a web-based service (https://ct2s.insigneo.org/ct2s/), potential users can find all the information about CT2S on the website and users can approach the application owner through the website by making a request. The user will then securely transfer anonymised CT scans to the application owners. The application owners will segment the bones from CT images, generate 3D finite element mesh, assigned local material properties based on CT attenuation, and then perform a set of finite element (FE) analysis to estimate the bone strength (ANSYS) and calculate the risk of hip fracture. The final results will be compacted into a PDF file and sent back to the designated user email who requested the service [1]. The workflow is installed and usable on Sheffield's HPC system ShARC. The workflow has also been tested on ARCHER (EPCC).

The target users for CT2S are primarily clinicians and researchers. The clinical application of this code is to provide a more accurate intervention strategy for elderly people with weaker bones, such as those who are clinically defined as osteopenic but not receiving any treatments. The workflow will provide a complete assessment of bone strength in 3D and analyse the risk factor of that particular individual in sustaining a fall in future. This information can be used in addition to clinical history and Dual-energy X-ray Absorptiometry (DXA) evaluations in order to determine the necessary intervention strategy. For researchers, CT2S can be applied to a wide range of problems concerning the mechanical response of long bones, including the prediction of fracture risk at different locations and for different age groups. Using this approach, the code is not directly distributed and the HPC elements are hidden from the clinicians to allow them easy access to this technology without the need to have a good understanding of HPC systems.

USFD has also developed a multiscale model for current absolute risk of hip fracture (ARF0) [3]. In addition to femur strength (obtained from CT2S pipeline) this model included information on body height, weight and fall rate. These were used to predict severity and frequency of falling. Recent work in our group further included in this model subject-specific information on three-dimensional trochanteric soft-tissue geometry from proximal femur CT images. These improved

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

hip fracture classification accuracy to 87% in the same 100 subject postmenopausal osteoporotic cohort mentioned above. We also modified the ARF0 model to predict 10-year risk of hip fracture (ARF10) by coupling it to a model simulating spatiotemporal loss of bone density driven by hormonal activity.

Based on ARF10 workflow developed by USFD, UNIBO has replicated the biomechanical computed tomography pipeline, and is developing an *in silico* trial solution, namely BoneStrength. The code is running on Cartesius (SURF, The Netherlands), Galileo, and Galileo100 (CINECA, Italy). By using a statistical anatomy atlas trained with the Sheffield cohort [4], over 1000 synthetic patient proximal femur FE models were generated from 94 real patient data. The ARF0 pipeline was applied to the synthetic patients, showing a good persistence of the original cohort biomechanical characteristics in the synthetic cohort. UNIBO is now working on bone ageing (physiology/disease) and drug effect (treatment) models in order to run a massive simulation of placebo vs drug phase III clinical trial, involving 1000 patients over 5-10 years.
The natural users of the solution will be pharmaceutical companies, which will be able to tune the cohort inclusion criteria to maximise drug effects, or even reach statistical significance with less actually enrolled patients in a clinical trial.

Technical specifications
The modelling elements of the CT2S workflow focus on providing an estimation of the strength of an individual's femur under a series of loading conditions, using a subject-specific FE model. This step involves the use of ITK-Snap for image segmentation to isolate the bone surface, ICEM CFD (ANSYS Inc, PA, USA) for meshing the segmented bone geometry using tetrahedral elements, Bonemat for personalising the bone material properties, and ANSYS (ANSYS Inc, PA, USA), running on ShARC, for performing the simulations. A total of 28 load simulations are performed (as replica computing) to comprehensively investigate the bone response under several fall configurations. Each load simulation requires 96 GB of memory shared across three cores in an OpenMP session. USFD has the capability of running the entire batch of 28 simulations simultaneously. Results are post processed in MATLAB (MathWorks, Natick, MA) with custom written scripts [1].
The ARF0 pipeline requires additional subject-specific information, specifically body weight and body height. The pipeline is implemented as a MATLAB function which takes these scalar values as input, in addition to a matrix of values of bone strength in dependence of the 28 impact orientations mentioned above. The MATLAB function gives the value of ARF0 as output. It performs a Monte-Carlo integration to estimate the risk of fracture over multiple fall scenarios. A sensitivity analysis has shown that 10 000 fall simulations are sufficient to estimate ARF0 with a reasonable residual uncertainty [2]. The fall scenarios can be analysed in parallel and the ARF0 code has been tested on a 20-core compute node on ShARC HPC using the MATLAB default MPICH2 library. Each ARF0 job has negligible memory and CPU requirements in addition to those of one CT2S job.

For the ARF10 pipeline, the CT2S pipeline needs to be repeatedly executed 10 times, each time changing only the bone material properties encoded in the FE model. The algorithm to modify the FE model is implemented as a sequence of executions of an ANSYS macro (to write out current material properties into a text file), a MATLAB script (to read the text file, modify the property values and write these to a text file) and an ANSYS macro (to read the modified text file). The output of each of the 10 FE models is passed through ARF0 pipeline, and the result is aggregated to obtain ARF10. A full analysis of resource usage of the ARF10 pipeline is underway, but the following must be noted; the analysis of each of the 10 FE models is identical to that of one CT2S job. However, these 10 CT2S jobs can be performed independent of each other. The

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

rest of the ARF10 pipeline comprises (a) generating the 10 FE models, (b) executing ARF0 on the output of each CT2S analysis, and (c) aggregating the ARF0 results to obtain ARF10. These steps are only partially parallelisable, however their resource requirement relative to the 10 CT2S jobs is negligible. Hence, the ARF10 pipeline is highly parallelisable as a whole.

BoneStrength pipeline has been tested on Intel Xeon processors in several HPC clusters, namely Cartesius, Galileo and Galileo100. The core of the computation is the simulation of over 500,000 FE models (28 falling simulation for each of the 1000 patients, for placebo and treatment arm, for 10 years) following a replica computing pattern. Each instance leverages ANSYS parallelization, using either OpenMP or MPI (IntelMPI and OpenMPI tested) on 6-8 processors; each simulation requires from 5 to 15 GB of RAM, depending on the I/O reduction used to avoid bottlenecks during massive runs. Postprocessing and data analysis are performed with custom Python scripts. To increase CPU efficiency, UNIBO is exploring the use of QCG-PilotJobManager to manage simulation submission.

HPC usage and parallel performance

The main motivation of HPC usage by CT2S and BoneStrength pipeline is the need for a rapid and cost-effective response to the clinical question. In particular, by leveraging HPC parallelism CT2S FE simulation can run simultaneously, and the physician can decide about the patient therapy within an hour. Conversely, BoneStrength simulations will require around 100 years if run serially, and HPC use can keep the user waiting time under two weeks.

While the single fall simulation scalability relies on commercial software performance (with 8 cores using MPI parallelism we reached a 6x speedup), the full pipeline shows linear scaling, being a replica computing pattern simulation. The largest BoneStrength production run involved 300 nodes on Cartesius (7200 cores) for a week.

Each single fall simulation by ANSYS default minimizes RAM consumption (approximately 5 GB) but has an important file I/O requirement (50-80 GB); while this approach is useful on workstations, or for single patient or small cohorts, for large cohorts (> 100 simultaneous patients) it is preferable to store all the temporary information in RAM and minimize file I/O to avoid bottlenecks. Indeed, ANSYS allows this approach, so the final requirements become 15 GB of RAM and 2-4 GB of file I/O. Each single-patient 1-year simulation (28 fallings) produces around 135 GB of final data, from which 20 MB are extracted for further postprocessing. These data can be stored for quality checks or deleted to limit space consumption.

In CompBioMed2 the CT2S/BoneStrength workflow has been automated and optimized, by reducing file I/O and switching to the more efficient ANSYS MPI parallelism. Next developments include the use of a dedicated job manager (we are exploring QCG-PilotJob, deployed at PSNC) and the possible porting of the workflow to the highly parallel Alya solver developed at BSC. We also plan to collaborate with ATOS to investigate simulation scaling and performances on various traditional and experimental architectures.

Cited references:

[1] I. Benemerito, W. Griffiths, J. Allsopp, W. Furnass, P. Bhattacharya, X. Li, A. Marzo, S. Wood, M. Viceconti, A. Narracott, Computer Methods and Programs in Biomedicine, 208, 106200 (2021).

[2] Bhattacharya, P., Altai, Z., Qasim, M. *et al. Biomech Model Mechanobiol* 18**,** 301–318 (2019).

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

[3] Mark Taylor, Marco Viceconti, Pinaki Bhattacharya, Xinshan Li, Journal of the Mechanical Behavior of Biomedical Materials, 118, 104434 (2021).

6.7 openBF

Application description

openBF has been used for modelling pathologies of the cardiovascular system such as vasospasm and ischaemic stroke. It has also been used for modelling mechanical thrombectomy procedures and support the development of next generation thrombectomy devices together with Anaconda, a Spanish SME.

Available for academic and clinical researchers, the code is Distributed open-source through Github at https://github.com/INSIGNEO/openBF. Information and documentation on the code are available on Github. Users can download the code from the Gihub repository and install it on their local machines/HPC clusters. The code is deployed on ShARC, USFD's HPC cluster, but external users need a guest account to access it.

Technical specifications

OpenBF is written in Julia and has typical memory requirement of 500 MB/job. The output size of a typical simulation is below 100 MB. The code is not parallel at the moment and has been deployed and test on Intel CPUs. It is distributed with in house developed pre-processing and post-processing Python script (numpy, GPy, SALib). Work is in progress to incorporate openBF with the VECMA toolkit (https://www.vecma-toolkit.eu/) to support sampling in the pre-processing stage and management of simulations

HPC usage and parallel performance

openBF is used for performing biomarker identification and sensitivity analysis of 1D cardiovascular models defined by hundreds of input parameters. The sensitivity analysis is performed through a statistical emulator that is trained on datasets whose training requires around 8000 simulator runs, each of which requires 15 minutes to complete on a single CPU. This would result in total computing time of 2000 hours. Concurrent simulations on 12 full ShARC nodes bring the needed time to 15 hours. Current developments are focusing on the improvement of the criteria for evaluating the convergence of openBF models and to define criteria for pathology-driven exploration of the input space.

6.8 PlayMolecule

Application description

PlayMolecule is a drug discovery web-service. It contains a variety of applications which allow users to accelerate and improve their drug discovery workflows using novel machine learning methods (such as binding affinity predictors) or through molecular dynamics simulations to elucidate biological structures and binding modes.

PlayMolecule is used daily by academic institutions and industry. Both students as well as experienced researchers are using PlayMolecule to evaluate machine learning methods or simplify their molecular dynamics workflows. PlayMolecule already counts close to two thousand registered users with around 80 new users registering every month.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

PlayMolecule is available at https://playmolecule.com/. The free access has restrictions imposed on atom count of molecular systems due to constraints in computing resources which are available.

Technical specifications

PlayMolecule consists of various separate components. The web server, the backend server and the individual applications. The web server is written in Python, React and Angular, the backend is written in GoLang and the applications are written mostly in Python with some lines in C++. The web server and backend server have a total memory requirement of around 12 GB RAM. The individual application runs usually require 16 GB or more RAM, at least 2 CPU cores and a few of them relating to molecular dynamics and neural networks require NVIDIA GPUs. The code is deployed with a custom Python installation script which handles all the system setup and data shipping from Google Cloud Storage.

The architecture of PlayMolecule can be seen in Annex 11.1.3. The web server communicates with the computation backend (written in GoLang) which in turn sends job executions through RabbitMQ to the Queue app which communicates with the queueing system. The queueing system then distributes the jobs to the workers and once completed the jobs send their results back to the backend which stores them in the MinIO server. The backend also employs a MySQL database to manage users, job executions, MinIO file tags and other information necessary.

The code itself in the PlayMolecule backends is mostly serial (excluding the Apache server). However individual jobs are typically sent to a queueing system such as SLURM and can run in an embarrassingly parallel manner. All PlayMolecule applications are containerized with Singularity to ensure easy deployment and reproducibility. For visualisation of the user interfaces and the results, PlayMolecule offers a web server which integrates a state-of-the-art 3D molecular viewer.

HPC usage and parallel performance

HPC resources can be leveraged by PlayMolecule both to serve a larger number of users as well as to provide faster computation. Applications such as AdaptiveSampling depend on multiple GPUs being available so that multiple simulations can be run in a high-throughput parallel manner to explore faster protein conformational space, or protein-ligand interactions and thus can fully leverage large computing clusters.

PlayMolecule applications mostly run in an embarrassingly parallel manner and thus scale linearly to the number of resources which are available and jobs which are run. Network usage is limited to the transfer of input and output files for jobs which execute on cluster nodes. The data transfer is done from the applications to the PlayMolecule backend which in turn stores the files in the MinIO server. Output files are typically in the order of a few megabytes, however applications such as SimpleRun and AdaptiveSampling generate molecular dynamics trajectories of multiple gigabytes and thus can increase network traffic. Computation time varies between applications ranging from a few seconds in ProteinPrepare to multiple weeks in AdaptiveSampling.

Ongoing work in PlayMolecule is focused on reducing data duplication between cluster nodes, the MinIO server and the web server to also reduce network load as well as storage requirements. All PlayMolecule singularity applications are built using internal Python libraries which we regularly evaluate for performance and algorithms.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

6.9 TorchMD

Application description

UPF has been focused on the continuous development of TorchMD, our software for the development and usage of machine learning potentials for molecular simulations. TorchMD is divided into three different parts/applications, with their own code and repositories, all hosted in Github and freely available for everyone (https://github.com/torchmd).

The first, TorchMD itself, which is an end-to-end differentiable molecular dynamics code. TorchMD-NET is a software for training and creating the neural network potentials. TorchMD-CG is a specific application to learn coarse-grained potentials, currently specialized in protein folding.

Technical specifications

At the moment, our efforts for application hardening are focused on TorchMD-NET, and trying to optimize both the speed of neural network training and evaluation in a production environment. Regarding the first problem, we have rewritten our current code and network architectures using PyTorch Geometric, a PyTorch-based library with optimized code to easily write, implement and run Graph Neural Networks (GNNs). Implementing TorchMD-NET with PyTorch geometric provided us with a ~3x speed up in training, which is especially significant if you are dealing with large training datasets.

Another issue we were encountering is the simulation speed of TorchMD with neural network potentials, particularly in our application for coarse-graining simulations. In order to gain some time, we have been using TorchScript to compile our models and switch from a pure Python code to a TorchScript program, optimizing the code and making it independent of Python, making it possible to run the scripts in a production environment where Python speed might be disadvantageous. By using TorchScript with our trained neural network potentials, we have given a ~1.8x speed-up in our coarse-grained simulations using TorchMD.

6.10 Virtual Assay

Application description

Virtual Assay is used to perform simulations of drug effects in populations of human ventricular cells, and to predict potential side effects of the drugs on the human heart. Virtual Assay has been designed to be accessible by everyone, including people with little or no expertise in computer modelling and simulations. The target users are mainly pharmaceutical companies and academia. In industry, the main application would be for drug safety testing during pre-clinical drug development. In academia, it can be used both for research and teaching. Virtual Assay is licenced through Oxford University Innovation. A free academic licence and a commercial licence are available, and they can be requested at:
https://innovation.ox.ac.uk/licence-details/virtual-assay-drug-screening-software-v-3-0/.

Technical specifications

Virtual Assay it is not a fully integrated HPC application and can run on most modern 64-bit PCs and laptops. The software can be installed only on Windows machines (or virtual machines) at present. The software provides a framework where in a population, drug simulations are performed, analysed and stored independently for each model, and this facilitates parallelisation. The software is set to automatically run simulations in parallel, up to the number of cores available.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

The code it is mainly written in C++ and the following is the minimum recommended hardware specification:

- Intel i7 quad core processor or AMD equivalent
- 8 GB RAM
- 5 GB hard disk space

7 HPC and Data Services Infrastructures

In this section we describe the HPC and data storage systems used by partners and available within the project. In this section we describe the services available through CompBioMed or within any collaboration initiated within the project, to access HPC and data storage systems as well as support activities provided for the users' applications.

7.1 Compute and Data Systems

7.1.1 CompBioMed HPC allocation programme

Since the beginning of the CoE, CompBioMed has been granting allocations on several large scale HPC resources to support the work of the CoE. Within the CoE we have offered, to the consortium partners, both Core and Associate Partners, the opportunity to access some of the systems managed by CompBioMed organisations (LRZ, EPCC (UEDIN), and SURF) and obtain direct support from the HPC experts at each site. This has been crucial for partners to efficiently use the resources and employ the queue systems on board the supercomputers.

Until now, we have used through this way about 7,000,000 cores hours on both CPU and GPU based systems hosted by project partners. The allocations listed on the website (https://www.compbiomed.eu/high-performance-computer-allocations/) are those that all partners in the consortium, as well as supported members from the wider user-community, are able to access to support both the pursuit of scientific goals as well as to improve the computational performance of their applications.

In addition to the CompBioMed internal allocations, we have benefitted from access to Tier 0 HPC systems offered through PRACE, which have been used by several partners within the project. Moreover, we actively monitored the deployment of the project applications and required underlying libraries and tools (e.g: Jupyter notebooks for training courses) on the different systems, in order to provide our users with programming environments for developing, maintaining, or porting their codes, which can help with software deployment and increase the computational capacity of biomedical applications and workflows.

7.1.2 CompBioMed co-design systems

Within the CompBioMed CoE, we have a diverse set of testing clusters for co-design purposes. They include different CPU architectures such as classical x86_64 as well as emerging ARM architectures combined with AMD or NVIDIA accelerators. We maintain a living document showing the current composition of the testing clusters.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

7.1.3 CompBioMed data storage infrastructures

In addition to the access to HPC systems, we have recently included to the possibility, for users affiliated with the consortium, to request access to data storage and management facilities.
Partner UEDIN, has made available access to the RDFaaS service (https://docs.archer2.ac.uk/user-guide/data/#rdfaas-file-system) and to the DDN WOS Object Store (https://cirrus.readthedocs.io/en/master/user-guide/object_store.html) which can be used as temporary storage for project partners data, and to support the development of data management tools and services.
In order to satisfy the need for services able to handle and make available for publication the large data produced during the project, we are in collaboration with a European project called DICE (Data Infrastructure for Capacity for EOSC, https://www.dice-eosc.eu/), to deploy a dedicated instance of B2SHARE at partner UCL. The plan is to setup a dedicated Virtual Machine (VM) and deploy the B2SHARE code (which is open source and available in the EUDAT project GitHub: https://github.com/EUDAT-B2SHARE/b2share) and first make a B2SHARE test environment. Then setup 50TB of data storage space in UCL, by mounting storage on our GPFS-based facility to the VM using NFS. We will make the configurations to run B2SHARE data repository as a service. UCL will be the service provider and the storage in the backend can be scaled upon need of the community. It is important to mention that the UCL Research IT department is directly involved in this collaboration to setup the data repository. The concrete plan for setting up the infrastructure and running B2SHARE as a service is made in collaboration with DICE partners (UCL, SURF, and BSC) and financed by the DICE project WP5, Task5.2 “CompBioMed Data Platform Integration". Defining the metadata schema to be implemented in the data repository to add metadata for publishing is being addressed in CompBioMed WP3.

7.2 Support Service for User Applications

7.2.1 CompBioMed applications benchmark and performance analysis tools

LRZ benchmark suite
LRZ has developed a first version of a python-based automated benchmark environment for the testing and benchmarking of CompBioMed applications which can be used for the systematic testing of the performances of the developed codes as well as for our co-design activities. The framework works together with JUBE (https://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/JUBE/_node.html), any other benchmarking systems and also with shell scripts.

The intention is to write one configuration file per benchmark (using the yaml data-serialisation language), including general sections as well as the sections “execute”, “analyse”, “result” and “postprocessing”. In “execute”, the commands to execute the benchmark on a given system can be specified, in “analyse”, the commands needed to generate the output (text) files and in “result”, the commands to parse the output file(s) and to extract the relevant data which are then fed into a sqlite database. The data from the sqlite database are then used to generate a json data file as well as plots which are then used for building the static webpage. Here, one can finally take extensive control over the static webpage within the “postprocessing” section. So, in total, after collecting the yaml files, the testing framework can automatically execute single or multiple different tests in an efficient way, analyse the results automatically and illustrate them in a very graphical and intuitive way within a static webpage. One can then, e.g., directly see the scaling plots of the generated benchmarks.

*This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No **823712**“*

So far, we have created JUBE tests for several molecular dynamics' codes such as GROMACS, AMBER, OpenMM, NAMD and LAMMPS, we however intend to write further tests for our most scaling codes HemeLB, Alya, PALABOS and HemoCell which will then enormously increase the speed with which we are benchmarking our codes with ease on many different architectures. This will then, in turn, be the basis of our co-design efforts where we e.g., investigate the dependence of the code performance on the hardware composition such as e.g., the CPU/GPU ratio and number.

To promote reproducibility and easy access to applications and workflows, LRZ has already created a python as well as SPACK package of the testing framework, which is however currently only used internally within the project. After experimenting at different CompBioMed HPC sites and further modifications, we however aim to provide the framework to the wider community.

SURF Easybuild/Reframe automated testing suite
SURF has worked with BSC on the integration of the Alya code within the automated workflow for software deployment and testing at SURF. The work, which also involved a student from the "Summer of HPC" programme, involved the porting of the building system used by Alya within the Easybuild framework (https://docs.easybuild.io/en/latest/Introduction.html) used at SURF, and set up specific benchmarks that have been integrated within the Reframe framework (https://reframe-hpc.readthedocs.io/en/stable/) used to execute fully automated regression tests on SURF systems. We are planning to extend the support of these technologies to other codes in the project, to allow users to easily access optimised codes and updated benchmarks tools and results.

7.2.2 Middleware and workflows support

QCG-Pilotjob benchmarks on CompBioMed systems
Within CompBioMed we are working in collaboration with the QCG-Pilotjobs development team from the Poznan Supercomputing and Networking Center (PSNC) in Poland to deploy a European workflow manager named QCG Pilot Job Manager (https://qcg-pilotjob.readthedocs.io/en/develop/) on different CompBioMed HPC systems and to evaluate the overhead of such tools and the impact they could have on exascale computational campaigns. This workflow manager enables automatic handling of complicated heterogenous workflows involving large number ensembles with different types of calculations performed concurrently. This falls under the planned activities to support the applications within "Ensemble" compute pattern, which is particularly relevant especially for validation, verification and uncertainty quantification; these are increasingly becoming essential for uptake of *in silico* methods in clinical medicine. We are currently testing and benchmarking it on all major supercomputing centres including PSNC, LRZ, EPCC and SURF. Preliminary results are reported in Annex 11.2, where we report the overhead measured on the HPC systems ARCHER2 (EPCC) and Cartesius (SURF) during our tests. As shown, the use of a middleware such as QCG-Pilotjob, does not impact the overall performances, proving to be an excellent tool to manage large computational campaigns on HPC systems. We are currently working to extend this support to more applications and simulation scenarios. Successful implementation of such middleware on HPC clusters would substantially facilitate performing complicated workflows on HPCs at large scale, including of course exascale type of runs.

LEXIS workflow
CompBioMed and LEXIS, (http://lexis-project.eu), are collaborating to create a workflow designed to introduce a level of resilience to exascale HPC simulations. Future exascale machines will have a very high node count. Nodes individually have a reasonable mean time to failure, or

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

MTF; however, when one aggregates 10-100 thousand nodes together in a single system, the overall MTF is far higher. Moreover, given exascale applications will employ MPI, and a typical MPI simulation will abort if a single MPI Task fails, then a node failure will cause entire simulations to crash.

Computational biomedical simulations employing these Exascale platforms may well employ time- and safety-critical simulations, where results are required at the operating table in faster than real-time. Given the increased threat of node failure, one mitigation is to employ what we have named Resilient HPC Workflows.

In CompBioMed, we are working with LEXIS to create a Resilient HPC Workflow, wherein a simulation that finds its host HPC has crashed will simply continue from where it left off on another HPC platform. The simulation's results will be available as if no catastrophic failure had occurred.

LEXIS already has a form of resilient HPC workflow in its arsenal, wherein the same simulation is launched concurrently on multiple HPC platforms: the chances of all the platforms failing is far less than any individual. However, such replicated computation can prove expensive, especially when employing the millions of cores expected at the Exascale.

The CompBioMed Resilient HPC Workflow, currently under development by CompBioMed2 Task 3.5, is using a simulation of blood flow through a heart with a pre-existing stent. Such a simulation may guide a surgeon in real-time when aligning a new stent, for instance.

The workflow currently involves three sites, namely LRZ, IT4I and EPCC at UEDIN. The application will launch at the initial HPC centre, currently at LRZ. It often writes data snapshots and, less frequently, restart files. As soon as these restart files are created, the LEXIS Platform will automatically duplicate these files to all the data nodes in the network, currently only IT4I, but we may involve the EUDAT data node at EPCC. These other platforms are designed to be located in different countries to mitigate against a centre-wide failure, e.g., power-outage, at the initial HPC centre.

As the simulation progresses, a single MPI task will abort which will cause the entire simulation to fail, thus mimicking a node failure. This failure will trigger the LEXIS Platform to restart the simulation on one of the remote HPC platforms, where it will stage the latest restart file at all the HPC nodes in the network from its nearest data node. The choice of HPC platform will then be performed by the LEXIS Platform's broker tool, namely its “Dynamic Allocation Module”.

The duplication of data for exascale simulations must consider not only the amount of data that needs to be copied and the bandwidth required, but also ensure both the duplication and subsequent staging follows FAIR data principles, given biomedical simulations can contain patient sensitive data. The amount and bandwidth will be addressed by employing the existing GEANT2 network, whilst FAIR data principles will be addressed by the LEXIS Platform itself, as the underlying mechanism for staging data employs the EUDAT services which, in turn, ensures FAIR data principles.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**“*

8 Support Services

This section focuses on the support services we provide to the CompBioMed community to help with the development and optimisation of the supported applications.

8.1 Scalability Support Service

The Scalability Support is a technical service to support users in improving the scalability of biomedicine solutions and applications.

CompBioMed offers free support to improve the scalability of your computational biomedicine solutions with high performance computers. Are you struggling because your code doesn't run in an acceptable time? Perhaps you are trying to simulate over 1000 virtual patients? Or do you need to perform some large-scale sensitivity analysis to get your solution certified by a regulatory authority? Whatever is the reason, we can help. Thanks to the funding of the European Commission, the CompBioMed Centre of Excellence in Computational Biomedicine now offers free support to organisations in their initial steps towards improving the scalability of existing computational biomedicine applications and deploying them on high performance computing resources.

The service is accessible at https://www.compbiomed.eu/compbiomed-scalability-channel/, which includes links to support@compbiomed.eu, application form, contact web form, and our Slack channel #scalability, a public channel in combination with "*In Silico* World" Community of Practice hosted on Slack.

This Support service was originally named the Parallelisation Support Service and focused on purely the parallelisation of serial applications. This Service is designed to bring in new users to HPC who currently employ serial applications within their company yet find their applications are unable to cope with the demands of increasingly larger data sets and/or find their applications take too long to run. CompBioMed offers to work with such users and authors to help parallelise their applications and port to our HPC platforms, thereby increasing the quality of their application's output.

The service has now been rebranded as the Scalability Support Service, and has been extended to include already parallel codes into the pool of new, potential applications that yet suffer from poor execution speed and/or need larger data set capabilities. Our Scalability Support Service looks to increase such applications' ability to scale to larger core counts and, thereby, increase the quality of its output.

The Scalability Support Services we offer to clients both external to CompBioMed and internal, from our own Core and Associate Partners

The Scalability Service itself is managed across a number of Tasks, namely Task 2.4 Emergent Community Application Support (Leader: UEDIN; Partners: BSC, UCL); Task 4.3 Optimising Biomedical Application Usage of Current and Emerging e-Infrastructures (Leader: SARA; Partners: UCL, UEDIN, UVA); Task 4.5: Relations with EU Initiatives and Other User Communities (Leader: UEDIN; Partners: SARA, CBK, UNIBO, LRZ).

Firstly, T2.4, seeks to advertise and draw in users to the Service. Secondly, the work to either parallelise and/or scale the applications falls to T4.3. Finally, Task 4.5 seeks to exploit existing

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

collaborations or affiliations to both help T2.4 advertise the Service and to help T4.3 with embedding the application within the CompBioMed HPC ecosystem. This process may eventually see the scaled and embedded applications become part of incubation processes planned in WP5 (i.e. *"T5.3 Preparing Content for External Use and Commercialisation"* and *"Task 5.2: Application Hardening").*

We are currently advertising this service within the *In Silico* World Scalability Slack channel. The Scalability channel capitalises on the expertise and knowledge of HPC experts among the CompBioMed partners to offer free support to those who need to improve the performance of their computational biomedicine solutions with high performance computers. But it is also a safe space where users share experiences and problems related to the computational efficiency and scalability of their computational biomedicine solutions.

However, in the near future, we will greatly extend the number of avenues available to potential clients. Via a single web page within the CompBioMed website, clients will see an overview of the Service, along with links to the Terms and Conditions of the Service itself, including our Corporate SLA, and links to CompBioMed's Data Policies. These Data Policies are currently available on our website, and are intended to both clearly state our adherence to the required data handling standards, such as EU GDPR and UK GDPR, but also to convince the potential clients that their data will be in safe hands. We are also planning a data adequacy statement, outlining how data moving in and out of the EU will be handled.

We shall also introduce more methods for potential clients to approach the service, over and above the current Slack channel, namely via a new email address, an application form in Word and, for the less experienced applicant, a simplified web form of the latter. The email address will be generic group email thus any traffic will be seen by a collection of HPC experts, thereby avoiding any single-point-of-failure found in an individual's email address. The Application Form has been created in collaboration with the PRACE SHAPE programme, which itself seeks to help scale SME applications, and the POP CoE, which seeks to help exascale HPC applications. The unique aspect of CompBioMed's Scalability Service is the biomedical expertise embedded in our partners, which permits an immediate rapport with any potential client. The application form includes the opportunity for clients to stipulate precisely what effort is required from CompBioMed and what effort will be supplied by the client themselves. Finally, the aforementioned web form is a simpler, reduced form of the application form, created to be a more friendly, open and less intimidating door for the less confident potential clients.

8.2 Internal Helpdesk

The number of avenues to access our Services is increasing and, indeed, the number of Services themselves have increased, we anticipate an increase in demand from end-users. To pre-empt the associated management issues of ensuring all contacts are maintained, retained and resolved to meet our SLA, an internal helpdesk system has been created. This now provides CompBioMed with better control of the internal management of our services and the implementation of specific agreement between the members of the consortium and external collaborators (Operation Level Agreements).

Our Internal Helpdesk system is now up and running and fully functional. It is a simple, lightweight system, based on a single Google Sheet [1]. Despite this apparent simplicity, the helpdesk actually retains the features of a fully functional Service Desk, with the light touch of a simple single interface, designed to be maintained by one or more staff members concurrently.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

Unlike typical Service Desks, however, our Internal Helpdesk is for staff only, and it is this restriction that permits its light touch implementation. The Helpdesk is entirely self-contained, such that clear instructions are included, a live list of services and associated experts may be maintained, and an exhaustive, extendable list of new, live and resolved tickets all appear in the single sheet.

The Helpdesk was designed and implemented by UEDIN, based on their decades of experience with User Support and follows many of the ITIL processes. UEDIN offers training to any interested parties from with the Core Partners, but it is hoped that the Helpdesk itself is intuitive and self-describing.

The Internal Helpdesk system can now ensure all our client-facing services are supported; enforcing each service to have a group of contact Experts: never a single-point-of-failure, accessible to the end-user via the Helpdesk Operator. The Operators monitor the status of the different services' requests, submitted to the Operators' group via the Slack channel, the new CompBioMed support email address, application forms, webforms, face-to-face contacts, etc. The Helpdesk Operators provide a triage for the simpler requests: provide a customer-facing conduit to the more technically-minded experts and can include OLA and SLA monitoring for both CompBioMed and third-party services, i.e., planned processes and procedures are executed timely and effectively. Finally, our Internal Helpdesk can also be used to monitor internal processes and inward-facing services, such as our HPC allocations process.

Cited references

[1] https://docs.google.com/spreadsheets/d/18HVyJC7vvXusND2m3d3Rb7DiGlY0EIS5rJWbOsnQpP0

9 Risk Management

Risk	Level	Mitigation
Complicated Service Management System will introduce overhead	Low	We designed the SMS to introduce as little overhead as possible, aligning procedures and roles with the already well-established organisation within the CoE. We will be monitoring the status of the service and its provisioning, adjusting and extending procedures if needed.
Lack of innovation and diversity in the service offer	Low	We will use the IP registry and support of the EEAB to building on the innovation infrastructure established during the first 3 years of operation of the CoE in Phase 1 (CompBioMed 1). The large network of associate partners established in CompBioMed will provide an extensive source of ideas and use-cases.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

Lack of customers/users	Low	The design and provision of services will be user-driven and coming from specific requirements of the community. We have an extensive network of core and associate partners, with already established collaborations, who will help in promoting services and capture needs.
Information related to existing or upcoming services too scattered. Risk to miss updates and input.	Medium	Within the SMS proposed here we will intensify the control and overview of IP capture within the different WP (e.g.: dedicated reports within WP meetings). The roles proposed here will help in overviewing the service lifecycle and support in the service building processes.
Unable to provide resources to maintain service delivery	Medium	We are working to establish services that will be used and adopted by the biomedical community at large and we are strengthening our collaborations with external EU projects and initiatives (e.g.: DICE, Lexis).
Services applications too technical/difficult to use in clinical set up	Medium	The project not only supports the development of large scale HPC applications, but also services which can already be used within clinical setup. We also are working in close collaboration with WP5 to analyse the supported HPC applications and develop solutions to facilitate non-expert users through tools and interfaces or by providing consulting for external clients.

10 Conclusions

This deliverable provides an overview of the different application and support tools we maintain within the compute and data services delivered within the project.

The report presents the heterogenous set of software solutions, currently developed within the project, reporting for each their technical specifications and infrastructure usage. As we can see, the project portfolio includes both highly parallel HPC applications, which are serious candidates to efficiently use emerging exascale systems, and compute services which do not require large HPC systems to be executed. Supporting both type of applications is an essential part of our project plans, as we need from one side to push the technical development of future exascale applications, but at the same time we to support the adoption of computational techniques within existing user communities and particularly the clinical contexts, where the use of HPC system is still not routinely adopted. Moreover validation, verification and uncertainty quantification types of run, which are increasingly becoming essential for uptake of *in silico*

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

methods in clinical medicine, need access, also for the simplest simulation scenario, to tools and hardware able to facilitate the large campaigns required for this type of analysis.

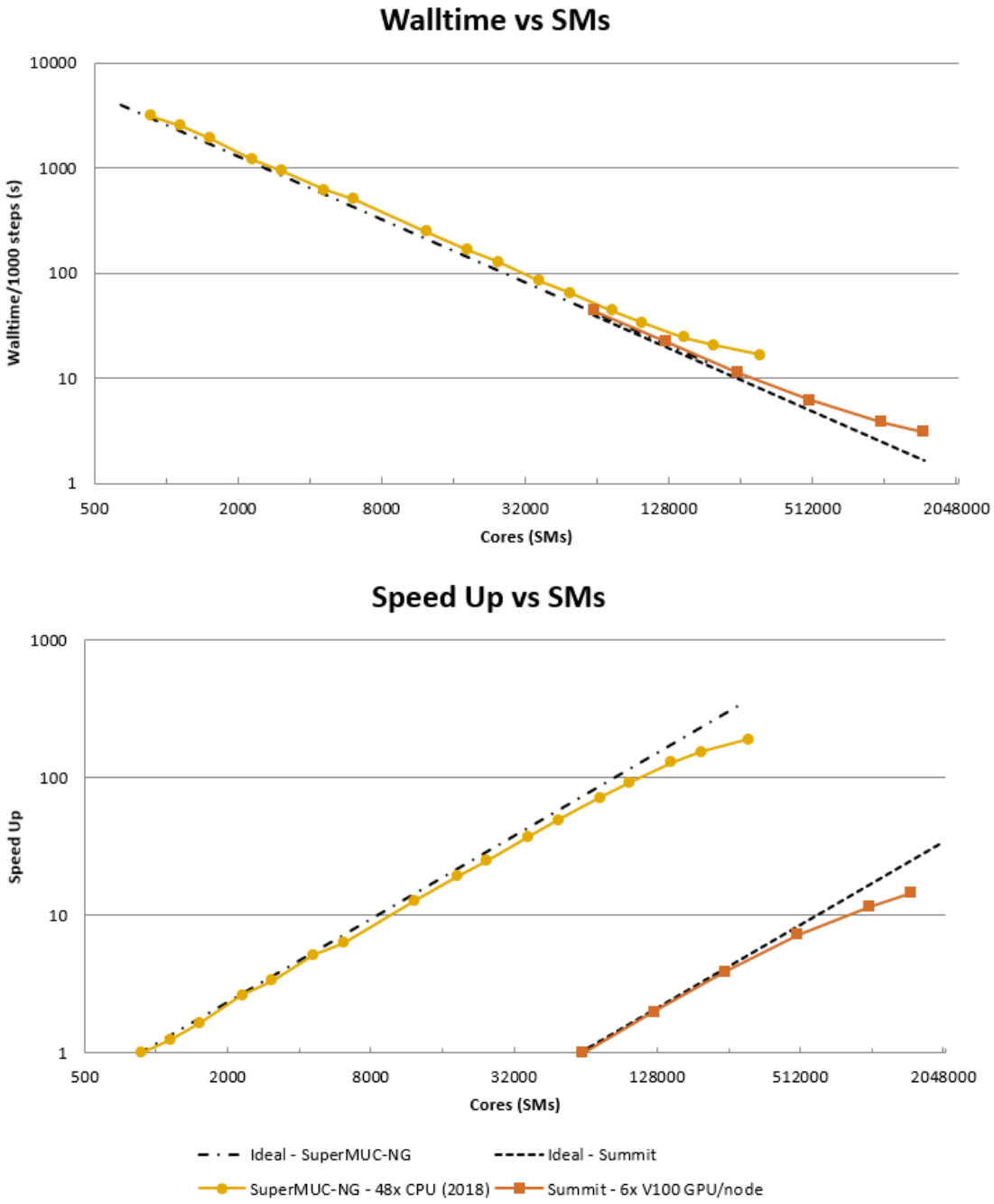
To support the development of new biomedical solutions, and to improve the performances and efficiency of the current compute and data services, within WP4, WP2 and WP5 we have designed several support activities to help the users to efficiently run their codes on large HPC systems, prepare and run effective and efficient benchmarks and to integrate their applications with middleware and data storage tools. These activities are essential to be able to build biomedical services that can be used within clinical contexts. We have plans to extend this type of support to more applications within the project and will offer our expertise as a service for external customers in the future. Through the use of our support services, we aim, indeed, to reach out to biomedical researchers outside of the consortium partners. Within WP4 we will provide technical support to develop and optimise applications, which can be then taken into WP5 incubation processes to extend their usage and potentially exploit commercialisation.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

11 Annexes

11.1 Applications Services Supplemental Material

11.1.1 HemeLB

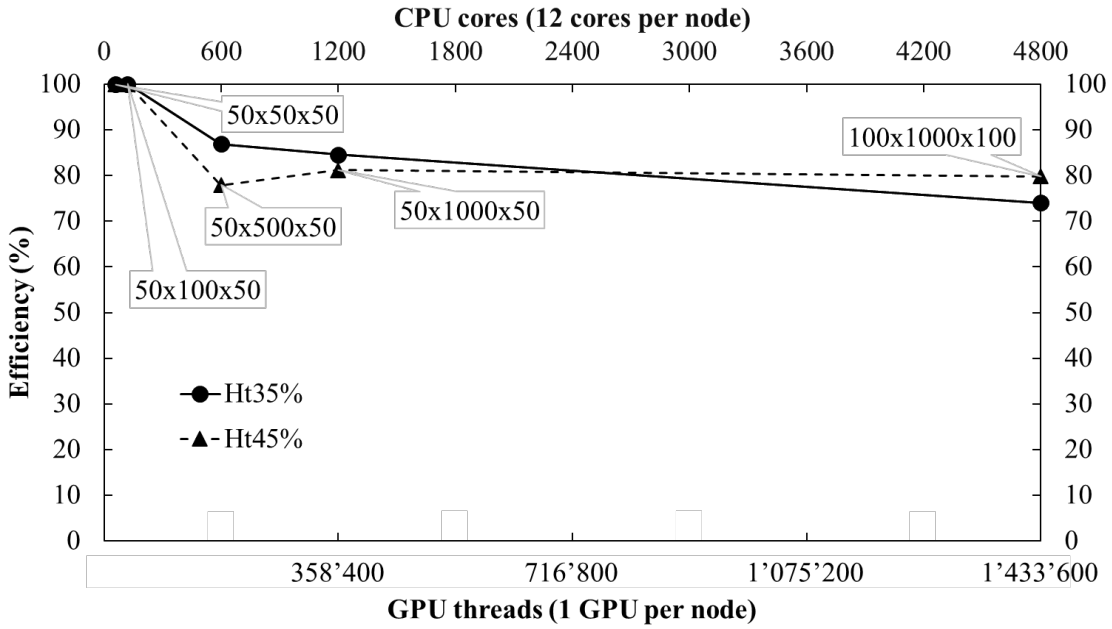


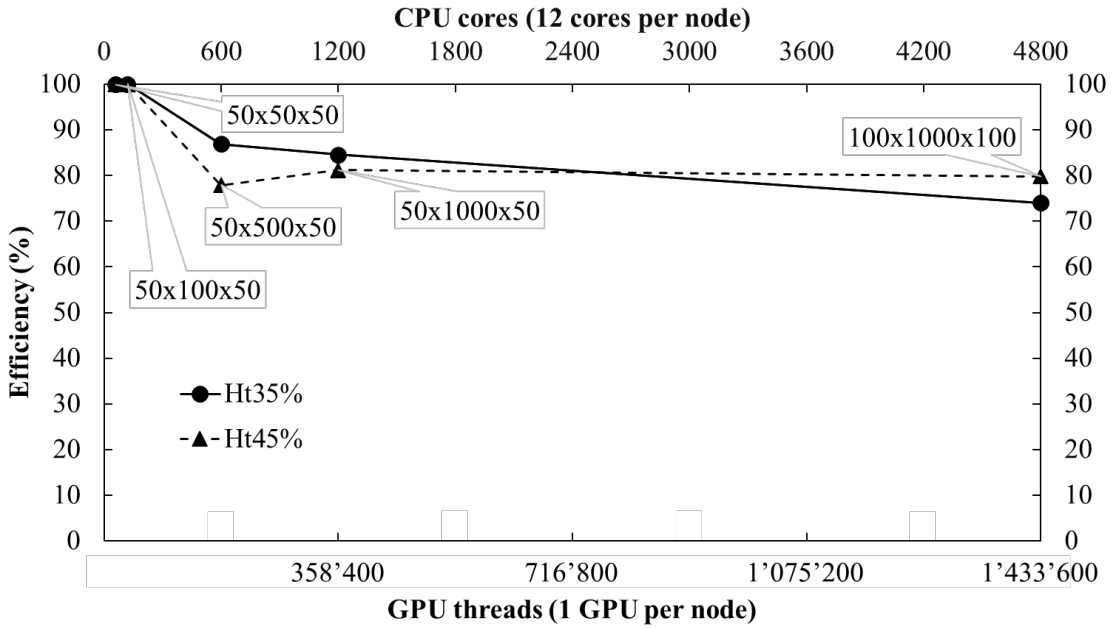
Strong scaling behaviour of walltime and speed up of the HemeLB CPU and GPU codes on large fractions of Tier-0 supercomputers SuperMUC-NG (Germany) and Summit (USA) using the same test geometry. The CPU code was run on up to 309,696 cores of SuperMUC-NG (99.6% capacity). The GPU code was run on up to 18,432 GPUs on Summit (66.6% capacity), in further

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

testing we have run up to 88.9% of Summit's capacity. For comparison we have used the measure of 1 CPU core being equivalent to a single GPU Streaming Multiprocessor (SM), this is the measure used by the Top500 list.

11.1.2 Palabos

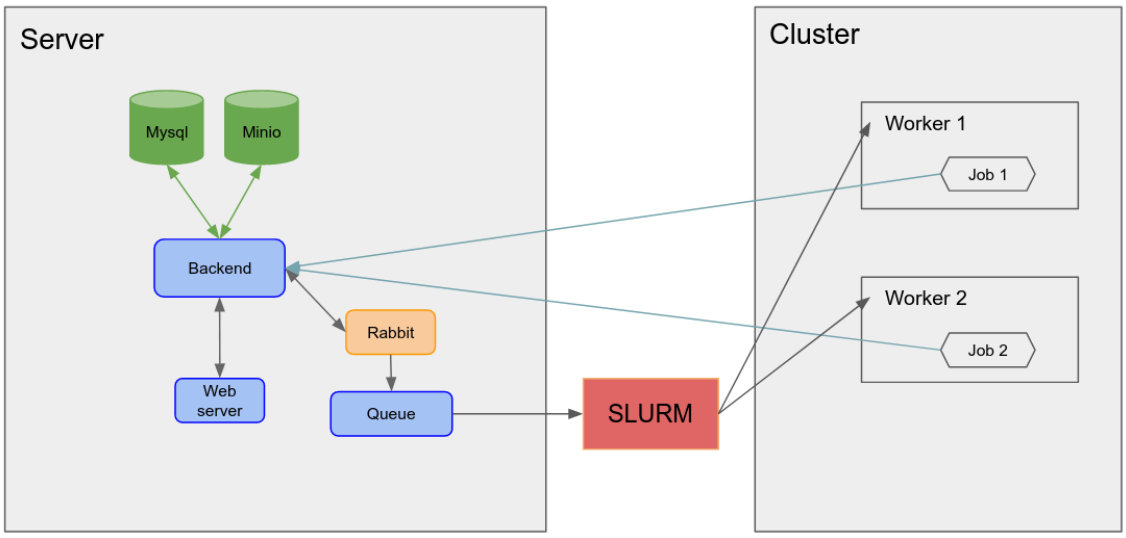




Scaling behaviour of the Palabos-npFEM code, as it simulates a coupled fluid – Red Blood Cell system on a heterogeneous platform (fluid on CPU, blood cells on GPU). The graph shows weak scaling, and the size of the simulated blood volume (in micro-meters) is indicated for each data point.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

11.1.3 PlayMolecule



Conceptual architecture and key components of PlayMolecule

11.2 QCG Performance on CompBioMed HPC Systems

# of nodes	# of cores	# of jobs	# of parallel jobs	qcg-pilotjob runtime (s)	qcg-pilotjob start&finish overhead (s)	qcg-pilotjob start&finish overhead (%)	resources usage (%)	qcg-pilotjob efficiency (%)	average job runtime (s)	avg job launch overhead (s)	# of estimated core-hours	lammps performance (ns/day)	resources usage (%)	qcg-pilotjob efficiency (%)
20	2560	160	40	57,73	3,27	5,664299324	86,7	99,8	12,51	0,01	81,18	4.118 - 4.382	81,03570068	94,13570068
40	5120	320	80	57,08	3,16	5,536089699	86,9	99,7	12,4	0,013	163,84	4.138 - 4.410	81,363910 3	94,163910 3
80	10240	640	160	57,6	3,54	6,145833333	86,3	99,3	12,42	0,0123	214,4	3.990 - 4.413	80,15416667	93,15416667
100	12800	800	200	60,3	3,91	6,484245439	83,3	99,2	12,55	0,0116	311,41	3.985 - 4.413	76,81575456	92,71575456
150	19200	1200	300	58,39	3,63	6,216817948	84,9	98,6	12,39	0,016	421,9	3.987 - 4.422	78,68318205	92,38318205
200	25600	1600	400	59,33	3,97	6,691387157	83,5	98,3	12,39	0,028	681,28	3.087 - 4.416	76,80861284	91,60861284
300	38400	2400	600	63,87	5,34	8,360732738	79,2	97,5	12,65	0,074		4.074 - 4.420	70,83926726	89,13926726

Performance results measured on ARCHER2 (EPCC) for the LAMMPS code with single task size of 64 cores and single run average walltime of 12.5s.

total jobs:	failed jobs:	total cores:	total nodes:	service runtime (s):	init overhead (s) :	finish overhead (s):	total overhead (s):	overhead ratio:	overhead core-hours:	rusage	efficiency
200	0	1200	50	132.69	7.60	0.29	7.89	5.90%	2.63	91.5	99.4
400	0	1200	50	252.72	7.02	0.36	7.38	2.90%	2.46	92.9	98.5
800	0	1200	50	480.88	7.22	0.35	7.57	1.60%	2.52	96.1	99.4
400	0	2400	100	148.56	14.05	0.35	14.40	9.70%	9.6	87.3	99

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*

800	0	2400	100	283.73	12.49	0.37	12.87	4.50%	8.58	85	95.2
800	0	4800	200	168.13	14.25	0.50	14.75	8.80%	19.67	72.9	90.3
2500	0	2400	100	819.14	2.38	0.70	3.08	0.40%	2.05	88.7	99.3
10000	0	4800	200	1543.05	3.43	1.32	4.76	0.30%	6.34	94.2	98.3
10000	0	9600	400	875.51	5.75	2.21	7.95	0.90%	21.21	88.3	95.3

Performance results measured on Cartesius (SURF) for the LAMMPS code with single task size of 6 cores and single run average walltime of 110s.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **823712**"*