





Grant agreement no. 675451

## CompBioMed

#### **Research and Innovation Action** H2020-EINFRA-2015-1 Topic: Centres of Excellence for Computing Applications

# D2.3 Report on Extreme Scaling and Porting of Exemplar Applications to Novel Architectures

Nork Package:	
---------------	--

Due date of deliverable: Month 27

Re-submission date: February 11, 2021

2

Start date of project: October 01, 2016

Duration: 36 months

Lead beneficiary for this deliverable: *BSC* Contributors:

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

#### Disclaimer

This document's contents are not intended to replace consultation of any applicable legal sources or the necessary advice of a legal expert, where appropriate. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user, therefore, uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors' view.

PU Page 1 Version 1.1





### **Table of Contents**

1	Version Log	3
2	Contributors	3
3	Acronyms and Definitions	4
4	Executive Summary	6
5	Introduction	6
6	HPC systems: those novel today, those novel tomorrow	8
	6.1 Hardware: the supercomputer architecture	9
	6.2 Software: the parallel programming models	2
	6.3 Cloud Computing: the novel HPC architecture today	3
	6.4 Exascale Computing: the novel HPC architecture tomorrow1	5
	6.5 Integration: the HPC-Cloud infrastructure	7
	Co-design: The (yet) unreached holy grail1	9
7	Discussion2	1
	7.1 Cardiovascular Exemplar	4
	Cardiac fluid-structure interaction simulations in HPC-Cloud using containers	
	(BSC)	4
	Cardiac fluid-electro-mechanical model of the heart for supercomputers and its	
	application to clinical, pharmaceutical and medical devices sectors (BSC, Oxford,	
	UPF)	6
	Blood platelet aggregation simulations using data analysis (UNIGE)	8
	Large-scale intracranial vasculature blood modelling for magnetic drug targeting	
	(UCL, UvA, UNIGE)	9
	Porting and optimisation of a dense cellular suspensions flow application on nove	el.
	and advanced microarchitecture Intel Skylake (BULL)	1
	Load balance strategies for multi-physics problems in large-scale blood flow	
	simulations (UvA)	3
	Increasing MPI communication efficiency for the HemoCell codebase (UvA)34	4
	7.2 Molecularly-based Medicine Exemplar	5
	Machine learning and large-scale computing (UPF)	5
	Web-based application to support the preparation of protein structures (UPF)3	6
	Large-scale computing and binding affinity prediction of bromodomain inhibitors	
	(UCL)	7
	Computational Methods for Structure-Based Drug Discovery (Evotec, UCL) 39	9
	7.3 Neuro-musculoskeletal Exemplar	0
-	7.4 Input/Output and visualisation in the HPC era	0
8	Conclusion	5
9	4 References	7

Version 1.1



#### **1** Version Log

Version	Date	Released by	Nature of Change	
V0.1	02/11/2018	Mariano Vázquez	Draft of template	
V0.2	13/11/2018	Mariano Vázquez	Draft, final version almost complete	
V1.0	21/12/2018	Emily Lumley	Final release version	
V1.1	23/07/2020	Emily Lumley	Update after final review recommendations	
V2.0	11/02/2021	Emily Lumley	Update after 9-month review of CompBioMed2	

#### 2 Contributors

Name	Institution	Role
Mariano Vázquez	BSC	Principal Author
Okba Hamitou	Bull	Co-author
Gabor Zavodszky	UvA	Co-author
Peter Coveney	UCL	Co-author
Alfons Hoekstra	UvA	Co-author
Bastien Chopard	UNIGE	Co-author
Jonas Latt	UNIGE	Co-author
Alfonso Santiago	BSC	Co-author
Gavin Pringle	UEDIN	Co-author
Adriá Pérez	UPF	Co-author
Andrea Townsend-Nicholson	UCL	Co-author
David Wright	UCL	Co-author
Andrew Narracott	USFD	Reviewer
Terry Sloan	USFD	Reviewer
Peter Coveney	UCL	Reviewer
Emily Lumley	UCL	Reviewer

Page 3

Version 1.1





### **3** Acronyms and Definitions

Acronyms	Definitions		
ABC	Approximate Bayesian Computations		
ACEMD	Molecular Dynamics software from Acellera		
AlyaCCM	Alya Cardiac Computational Model		
ΑΡΙ	Application Progressing Interface		
ARP	Architecture Review Board		
AVX	Advanced Vector Extension		
BAC	Binding Affinity Calculator		
CFD	Computational Fluid Dynamics		
CGNS	CFD General Notation System		
CHASTE	Cancer, Heart and Soft Tissue Environment		
CoE	Centre of Excellence		
CPU	Central processing unit		
CUDA	Compute Unified Device Architecture		
DEM	Discrete Element Method		
ESMACS	Enhanced Sampling of Molecular Dynamics with Approximation of Continuum Solvent		
GB	Gigabyte		
GPCR	G-protein coupled receptor		
GPU	Graphics processing unit		
HDF5	Hierarchical Data Format 5		
HemoCell	High PErformance MicrOscopic CeLlular Library		
НРС	High Performance Computing		
НТВАС	High-Throughput Binding Affinity Calculator		
HTMD	High-Throughput Molecular Dynamics		
I/O	Input / Output		
ILP	Instruction-level parallelism		
IPC	Instruction per cycle		
КРІ	Key Performance Indicator		
LBM	Lattice Boltzmann Method		
PU	Page 4 Version 1.1		





LIC	Line Integral Convolution		
MD	Molecular Dynamics		
ML	Machine Learning		
MPI	Message Passing Interface		
NetCFD	Ninf computational component for CFD		
NUMA	Non-uniform Memory Access		
OpenCL	Open Computing Language		
OpenMM	A high performance toolkit for molecular simulation		
OpenMP	Open Multi-processing		
PDB	Protein Data Bank		
RAM	Random access memory		
SIMD	Single Instruction Multiple Data		
SPMD	Single Programme Multiple Data		
SRAM	Static random access memory		
TIES	Thermodynamic Integration with Enhanced Sampling		
UMA	Uniform Memory Access		
WP	Work Package		





#### **Executive Summary**

This report is complementary to D2.2 Report on Deployment of Deep Track Tools and Services to Improve Efficiency of Research and Facilitating Access to CoE Capabilities.

In this document we focus on two issues which result in a combined strategy. On one hand, we look at the current status of those codes from the CompBioMed software stack which are on the road to exascale computing. The CompBioMed software stack is diverse in nature with three specific compute patterns (described below) which are relevant to the code base making the performance and co-design activities related to it equally diverse. On the other hand, we consider the porting to current architectures which are considered as "novel", notably the strongly emergent Cloud Computing model with particular focus on its innovations with respect to HPC capabilities. As we describe in the present document, addressing both issues concurrently results in improving what is becoming a powerful combined solution: HPC-Cloud computing. Whilst we recognise that cloud computing has its advantages, this is just another tool in the goal of expanding our codes to run on future exascale machines. This report is a selected compilation of the research done in this area by CompBioMed partners, in papers published or on the way to being published.

#### 5 Introduction

Until relatively recent times, a large majority of those researchers who strongly rely for their investigations on computational tools were confident that Moore's law would let them obtain progressively better results simply by awaiting a new generation of hardware. Since the 1970's, Moore's law established that computing power should roughly speaking double every two years. Therefore, one could reasonably expect that the same code compiled in a new architecture every two years should run twice as fast as the previous one, providing that its software design does not change that much (neither improving nor impairing its performance). However, as supercomputers became more accessible, a much more powerful "law" appeared to boost research by orders of magnitude beyond Moore's Law: parallelism. Computer parallelism allows tasks to be performed at the same time—concurrently—by distributing work, data or both. This potential can be thoroughly exploited if and only if (a) both hardware and software is parallelisation compliant and (b) software design maps to hardware architecture. It is also true that innovaitons in software based on new ideas and methods can accelerate performance dramatically and in a way not directly related to Moore's law and hardware related acceleration. This is rendered more complex as hardware vendors are continuously evolving their products, resulting from time to time in substantial changes of paradigm. Keeping pace with such computer changes is a significant challenge.

Biomedical research is particularly sensitive to computational tools and their efficiency on large systems because, quoting PRACE experts [1], "Life science is one of the fastest-

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 675451"

Page 6





growing users of high-performance computing both in Europe and worldwide, with a wide range of uses from chemistry, bioinformatics, and structural biology to diagnosis and treatments in clinical settings." However for several reasons, biomedical systems are challenging to simulate. Simulating the human body requires complex computational models that, to be accurate, require very fine space and time discretisation. The models are usually non-linear, increasing the need for sophisticated solver strategies, which are always computationally expensive. Compounding this, space and time scales range extremely widely in these systems, each scale being solved using a different model, substantially increasing the complexity through the strong multi-scale and multi-physics character of such problems. In order to deliver results which can be used in clinical contexts, we need to develop and deploy actionable codes – ones that produce accurate and precise predictions of clinically relevant scenarios ahead of interventions being performed by clinicians. To receive regulatory approval for such applications, the codes must be certified in terms of VVUQ – please spell this out. All this requires ultra-high end HPC.

Mapping efficiently the complex software arising from such modelling to the HPC hardware is one mission of CompBioMed which enables us to put it in the hands of the users who require it, as an equally important mission of the CoE. This document describes the efforts made so far in CompBioMed to analyse the porting and deployment of the project's software stack on what is currently considered "novel" architectures (among them notably HPC-based Cloud Computing) and, on future predictions of HPC hardware, i.e., the exascale computing systems.

#### 6 CompBioMed High Performance Compute Patterns

In [REF: Multiscale Computing for science and engineering in the era of Exascale performance, Hoekstra et al., Phil Trans A, 2019] the authors define a taxonomy for multiscale modelling applications and HPC based on computing patterns. In this document and inspired by those ideas, we extend them to general applications (not only multiscale). The proposed CompBioMed three key patterns are:

- 1. **Monolithic** Deployment of a single computational job spread over a large number of compute resources, including workflows necessary to manage such large computational jobs
- 2. **Coupled** Deployment of multiple, communicating subcomponents each assigned to a sub-section of compute resources. In an heterogenous system, each sub-section can be of different architecture. The application subcomponents can be different codes or different instances of the same code.
- Ensemble Multiple instances of an application launched in parallel with different input data; each such instance may be a monolithic parallel code, itself running at extreme scale. These are now becoming part of highly complex data intensive workflows combining conventional HPC applications with machine learning and/or AI components, as well as for validation, verification and uncertainty quantification.

PU

Page 7

Version 1.1



This categorisation is not unique to CompBioMed, it is recognized widely in the HPC community as key "usage patterns". It is worth mentioning that the Ensemble pattern is sometimes considered as two separate class: *ensemble* and *complex workflows*. In this document we discuss it as a single class. These patterns represent significant differences in the deployment strategy that should be taken into account when assessing scaling and performance as part of exascale readiness.

In CompBioMed, and inheriting the characteristics of the computational medicine domain, we have applications falling under the three aforementioned patterns.

#### 7 HPC systems: those novel today, those novel tomorrow

HPC systems are becoming more and more complex and the hardware is exposing massive parallelism at all levels, with their own needs depending on the computational pattern, as described below. Exploiting these resources is a huge challenge. In order for the reader to understand the concepts in the following sections, we first explain briefly and in layman's terms the different levels of parallelism available on a supercomputer. We do not aim to give a full description of the state of the art of the different architectures available in supercomputers, but a general overview of the most common approaches and concepts. This follows the ideas introduced in [2], whose authors belong to the CompBioMed consortium. We firstly describe hardware and then software.



Figure 1. Anatomy of a supercomputer. Memory latency and size (left) and parallelism (right) to exploit the different levels of hardware (middle). Extracted from [1].





#### 7.1 Hardware: the supercomputer architecture

**Figure 1** shows the different levels of hardware in a supercomputer together with the associated memory latency and size, as well as the type of parallelism to exploit them. The numbers are expressed in terms of orders of magnitude and provide a general idea, as they are system dependent.

These levels are the following:

**Core/Central Processing Unit (CPU):** We could consider a core as the first unit of computation, as a core is able to decode instructions and execute them. Here, within the core, we find the first and lowest level of parallelism: instruction-level parallelism. This parallelism is offered by so-called superscalar processors, and its main characteristic is that they can execute more than one instruction during a clock cycle. There are several developments that allow instruction-level parallelism such as pipelining, out-of-order execution, or multiple execution units. These techniques allow more than one instruction per cycle (IPC). The exploitation of this parallelism relies mainly on the compiler and on the hardware unit itself. Reordering of instructions, branch prediction, renaming or memory access optimisation are some of the techniques that help to achieve a high level of instruction parallelism.

At this level, complementary to instruction-level parallelism, we can find data-level parallelism offered by vectorisation. Vectorisation allows the application of the same operation to multiple pieces of data via a single instruction. Nowadays almost every processor in the market provides some kind of vectorisation through SIMD registers and compilers allow exploiting these features. For instance, MareNostrum's processors allow 32-byte (AVX2) and 64-byte (IMCI/AVX-512) SIMD registers in its different partitions. It is worth mentioning that the programmer is responsible for exposing this parallelisation level, requiring coding good practices. Otherwise, the compiler does not identify data level parallelism. The performance obtained from this is highly dependent upon the type of code being executed. Scientific applications or numerical simulations can often benefit from vectorisation as the computation they must perform usually consists of applying the same operation to large pieces of independent data.

**Socket/Chip:** Coupling of several cores in the same integrated circuit is a common approach and is usually referred to as multicore or many-core processors (depending on the amount of cores it aggregates). One of the main advantages is that the different cores share some levels of cache memory. The cache memory is a static random access memory (SRAM), which the chip can access faster than regular random access memory (RAM). The shared caches can improve the reuse of data by different threads running on cores in the same socket, with the added advantage of the cores being close on the same die (higher clock rates, less signal degradation, less power).

Page 9

Version 1.1



Having several cores in the same socket allows thread-level parallelism, as each different core can run a different sequence of instructions in parallel whilst having access to the same data.

These two deepest levels, "Core/CPU" and "Socket/Chip" are and will be definitely present in any future architecture, being the deepest layers at which a programmer can have control. Therefore, the three patterns should devote equal optimization effort here: in the three cases data and process ordering will be very influential on, for instance, memory access and vectorization.

<u>Accelerators/GPUs</u>: Accelerators are specialised hardware that consist of hundreds of simpler computing units that can work in parallel to solve specific calculations over large pieces of data. They include their own memory.

Accelerators need a central processing unit (CPU) to process the main code and off-load the specific kernels to them. To exploit the massive parallelism available within the GPUs, the application kernels must be rewritten. The dominant programming language is OpenCL (Open Computing Language) that is cross-platform, while other alternatives are vendor dependent, such as nVIDIA's Compute Unified Device Architecture programming language, widely known as CUDA.

In monolithic patterns, codes can run either completely on the accelerators or offloading part of them (for instance the solver) to the accelerator while the other part runs in the host. Both situations represent different challenges, and in the second one host/accelerator data transfer becomes critical. Data transfer is also critical in heterogeneous coupled patterns.

**Node:** A computational node can include one or several sockets and accelerators along with main memory and Input/Output. A computational node is, therefore, the minimum autonomous computation unit as it includes cores to compute, memory to store data, and a network interface to communicate. The main classification of shared memory nodes is based on the kind of memory access they have: uniform memory access (UMA) or non-uniform memory access (NUMA). In UMA systems, the memory system is common to all the processors and this means that there is just one memory controller that can only serve one request at a time; when several cores are issuing memory requests, this becomes a bottleneck. On the other hand, NUMA nodes partition the memory among the different processors; although the main memory is seen as a whole, the access time depends on the memory location relative to the processor issuing the request.

Within the node, thread-level parallelism can also be exploited as the memory is shared among the different cores inside the node.

This level is very important specially for monolithical applications or coupled ones in which of the components is a monolithical one.

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 10



PU



**<u>Cluster/Supercomputer</u>**: A supercomputer is an aggregation of nodes connected through a high-speed network with a specialised topology. We can find different network topologies (i.e., how the nodes are connected), such as a 2D or 3D torus or hypercube. The kind of network topology will determine the number of hops that a message will need to reach its destination or communication bottlenecks. A supercomputer usually includes a distributed file system to offer a unified view of the cluster from the user point of view.

The parallelism that can be used at the supercomputer level is a distributed memory approach. In this case, different processes can run in different nodes of the cluster and communicate through the interconnect network when necessary.

We have seen all the computational elements that form a supercomputer from a hierarchical point of view. All the levels explained above also include different levels of storage that are organised in a hierarchy too. Starting from the core, we can find the registers where the operands of the instructions that will be executed are stored. Usually included also in the core or CPU, we can find the first level of cache. This is the smallest and fastest one; it is common that it is divided into two parts: one to store instructions (L1i) and another to store data (L1d). The second level of cache (L2) is bigger, still fast, and placed close to the core too. A common configuration is that the third level of cache (L3) is shared at the socket and L1 and L2 are private to the core, but any combination is possible.

The main memory can be of several gigabytes (GB) and much slower than the caches. It is shared among the different processors of the node, but as we have explained before it can have a non-uniform memory access (NUMA), meaning that it is divided in pieces among the different sockets. At the supercomputer level, we find the disk that can store petabytes of data. For instance BSC's MareNostrum IV has a total capacity of 24.6 petabytes, EPCC's Archer 4.4 petabytes and SURFsara's Cartessius 7.7 petabytes.

At this level, the three patterns reveal all their complexity and variety. **Monolithic** applications stress the OpenMP/MPI interconnects in a relatively uniform way, with eventually load balance problems which can be addressed with careful programming and/or smart middleware. **Coupled** applications have the same problems, but adding the difficulties coming heterogeneous architectures on which we can deploy the different components of a coupled application. Multipoint communications on heterogeneous system is a complete challenge. Finally, **ensembles/workflows** stress all of the above, plus the parallel file systems themselves. This last pattern is the newest "usage pattern" and therefore the least well understood in performance terms.

In CompBioMed applications, the computational patterns require different ways of addressing the bottlenecks, found at the supercomputer levels mentioned above. By profiling the applications, we are led to optimize the porting. This is done within

Version 1.1

\*\*\* \* \* \* \*

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 11

ΡU



CompBioMed as well as in collaboration with other CoEs, including E-CAM and POP in particular.

#### 7.2 Software: the parallel programming models

The different levels of the HPC software stack are designed to help applications exploit the resources of a supercomputer (i.e., operating system, compiler, runtime libraries, and job scheduler). We focus on the parallel programming models because they are close to the application and specifically on OpenMP and MPI because they are, at the moment, the standard *de facto* HPC environments.

**OpenMP (Open Multi-Processing):** A parallel programming model that supports C, C++, and Fortran programming languages. It is based on compiler directives that are added to the code to enable shared memory parallelism. These directives are translated by the compiler supporting OpenMP into calls to the corresponding parallel runtime library. OpenMP is based on a fork-join model, meaning that just one thread will be executing the code until it reaches a parallel region; at this point, the additional threads will be created (fork) to compute in parallel and at the end of the parallel region all the threads will join. The communication in OpenMP between the different threads, is done through the shared memory. The user must annotate the different variables with the kind of data sharing they need (i.e., private, shared). OpenMP is a standard defined by a non-profit organisation: OpenMP Architecture Review Board (ARB). Based on this definition, different commercial or open source compilers and runtime libraries offer their own implementation of the standard.

The loop parallelism in OpenMP had been the most popular in scientific applications. The main reason is that it fits perfectly the kind of code structure in these applications: loops. And this allows a very easy and straightforward parallelisation of the majority of codes.

Since OpenMP 4.0, the standard also includes task parallelism, which offers a more flexible and powerful way of expressing parallelism. But these advantages have a cost: the ease of programming. Scientific programmers still have difficulties in expressing parallelism with tasks because they are used to seeing the code as a single flow with some parallel regions in it.

**MPI** (Message Passing Interface): A parallel programming model based on an Application Programming Interface (API) for explicit communication. It can be used in distributed memory systems and shared memory environments. The standard is defined by the MPI Forum, and different implementations of this standard can be found. In the execution model of MPI, all the processes will run the main code (or function) in parallel. In general, MPI follows a so-called single program multiple data (SPMD) approach although it allows running different binaries under the same MPI environment (multicode coupling). In particular for the CompBioMed case, while most of its simulation software stack exploit hybrid MPI / OpenMP parallelisation, only BSC's Alya actively uses multi-code coupling on top of MPI / OpenMP parallelisation. HemeLB was a use case for

PU

Page 12

Version 1.1





the new MPI-4 standard which for the first time handles clean 64-bit communication to alleviate the blocks arising from huge communication demands of the very large models one seeks to simulate today. [ref our J R Soc Interface Focus 2020 paper on HemeLB.]

#### 7.3 The Cloud Computing paradigm

In the original CompBioMed proposal we mentioned the fact that academically run HPC systems invariably support diverse groups of many users and so are subject to resource contention issues. Although within research environments this lack of quality of service guarantee is an inconvenience, scientists are resigned to it and simply work around the problem in order to benefit from the ultimate goal of producing high-quality research papers which make use of and often depend on access to large-scale computing resources. However, in environments subject to hard deadlines (for example, when decision support is required for urgent clinical intervention) it becomes important to be able to immediately and reliably allocate appropriate resources sufficient to meet the goal. When the original CompBioMed proposal was written this certainly was an important aspect to explore, two years later cloud providers have become key players in both academic and non-academic environments for the provision of on-demand computing.

There are many reasons for this, the following two being the most important for us: (a) containers are progressively more powerful, flexible and efficient, even when deployed on traditional HPC resources, and (b) at the same time that governmental supercomputing centres are selling core hours on their resources, commercial cloud providers offer "bare metal" instances improving their HPC capacities. It is worth remarking the importance of containers: they are reduced versions of an operating system with all associated software that a given application needs to run. They are therefore an extremely flexible way of deploying applications reliably, reproducibly, and rapidly, in heterogeneous cloud environments. For these reasons, we strongly believe that Cloud Computing is the real novel HPC architecture today.

Cloud computing advances for biological systems and its importance as an HPC resource are described in [3], a work partially funded by CompBioMed. This makes the observation that in the last ten years, virtualisation technologies underwent significant enhancements to their accessibility and ease of use. Considering also the current great abundance of hardware resources, not only full-stack but also lightweight virtualisation (containers) are becoming ideal platforms on top of which users can build their own "cloud-based platforms".

Operating-system-level virtualisation, also known as "containerisation", is an increasingly popular strategy today. There are several challenges with this, among them: *portability* (which isolates an application and its dependencies), *security* (for safer use of containers), *reproducibility* (to reproduce results no matter where the container is deployed) and *performance* (to keep the container performant, reducing latencies and overheads). Due to CompBioMed's focus on HPC-based applications, we started by

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 13





investigating two specific aspects: a container's *performance* and *portability*, leaving security and reproducibility for future research.

HPC is by definition focusing on maximising performance while running a single welloptimised parallel task. As such, having yet another software layer for handling containers that are stealing precious resources from our simulation is often seen by HPC scientists as a waste or an unnecessary complication. On the other hand, in data centres and HPC centres access and use is becoming progressively more complex, both from the software and from the hardware point of view, and these are strongly penalising portability. The software layers needed for operating a large production cluster require a slow, unique, error-prone, and often non-portable deployment effort. With the recent advances in emerging technologies (processing units, storage, network) the hardware also requires extra effort from system administrators and users. Thus, we have the request for absolute performance from pure HPC users, and on the other hand, we have the compromise of trading performance for an easier and more portable deployment of system software and applications. Additionally, it is obvious that all the modern general purpose supercomputers should allow that all applications running on them can expect to benefit from HPC. Therefore, such supercomputers to which we have access are never optimised for one kind of application only. This is even more the case now, with heterogeneous high-end architectures which are also used by people doing ML/AI in an embarrassingly parallel manner on hundreds to thousands of GPUs.

Additionally, simulating biological systems is fundamentally a multi-scale / multi-physics problem [4]. Predictions of real world events, such as weather forecasting, when cement will set, the occurrence of an earthquake or what medical intervention to perform in order to save a person's life, all require the bringing together of substantial quantities of data together with the performance of multi-dimensional simulations before the event in question occurs. Such forms of calculation are among the most demanding in computational science, as they need to be done rapidly, accurately, precisely and reliably. Moreover, they must include the quantification of the uncertainties associated with them. All these systems are *multi-scale* in nature, as their accuracy and reliability depend on the correct representation of processes taking place on several length and time scales. Only now, as we move toward the exascale era in high performance computing (HPC) can we expect to be able to tackle such problems effectively and, eventually, in a routine manner.

In a multi-scale simulation, and considering the aforementioned monololithic, coupled and ensemble patterns, each relevant scale needs its own type of solver, which represents the scales own requirements derived from their respective focus of the associated physical model. Accordingly, a multi-scale model is no more than a collection of coupled single scale models (loosely defined based on the dominant physical properties that can be computed reliably with a dedicated, so-called "monolithic" solver). With this in mind, it is very reasonable to think that, instead of a very large hardware system to solve "all at once" in the largest possible monolithic algorithm, multi-scale / multi-physics problems will be solved by efficient coupling scales and

PU

Page 14

Version 1.1





physics in a relatively controlled manner, which from the computational point of view involves the coupling of both codes and hardware systems. Each problem should be solved as efficiently as possible and coupling must be done avoiding bottlenecks due to algorithmic reasons or communication latency.

In this context, containerisation appears to be a good option to allow rapid deployment in different parallel architectures, providing that efficiency is sustained. It is important to remark that containerisation is not at all an exclusively cloud-related topic, because in heteogeneous supercomputing systems it could also be an important weapon to somewhat smooth the heterogeneity. Therefore, CompBioMed should keep an eye on any type of container, from those which can offer HPC capabilities, such as Singularity, down to very popular containers which are useful for smaller applications, such as Docker. The reason is that both coupled and ensemble patterns will eventually need mixtures of large and small size parts. Although similar evaluations have already shown promising results for small benchmarks, we base our evaluation not on specific benchmarks but on a large production, biological, organ-level simulation involving hundreds of thousands of lines of code. CompBioMed therefore aims to assist both HPC developers, end users and HPC system administrators to make the best choice to find the optimal scenario regarding performance and portability for running a production scientific application using containers on different supercomputers. Thanks to the wide scope of our applications, we explore containerisation in several scenarios. At the molecular level, an example of their effective use can be found in the work of the CompBioMed consortium partner Acellera. Acellera's "In Silico Binding Analysis" service makes extensive use of Amazon's EC2 Cloud computing platform. The large quantity of computation required, coupled with the need to meet the pressing deadlines of customers would be uneconomic to undertake on in-house resources, or via outsourcing to a conventional HPC centre. At the other end lies Alya, the multi-physics code developed by BSC, which solves biomedical applications at the cell, tissue and organ level. In the discussion section, these and more examples will be presented, and in the following deliverable D2.4, we will include more examples, analysing the impact of our work.

#### 7.4 Exascale Computing

We previously mentioned Moore's law. Moore's law states that the number of transistors in a dense integrated circuit doubles approximately every two years. This law formulated in 1965 not only proved to be true, but it also translated into the doubling of the computing capacity of cores every 24 months. This was possible not only by increasing the number of transistors but also by increasing the frequency at which they worked. However, we can say that in the last few years the end of this paradigm seems to have arrived. The reason is that the performance of a single core is no longer increasing at the same pace. There are three main reasons for this:

**The memory wall:** This refers to the gap in performance that exists between processor and memory (CPU speed improved at an annual rate of 55% up to 2000, while memory speed only improved at 10%). The main method for bridging the gap has been to use

Ρ	U

Page 15

Version 1.1



caches between the processor and the main memory, increasing the sizes of these caches and adding more levels of caching. But memory bandwidth is still a problem that has not been solved.

The instruction-level parallelism wall: Increasing the number of transistors in a chip as Moore's law says is used in some cases to increase the number of functional units, allowing a higher level of instruction-level parallelism (ILP) because there are more specialised units or several units of the same kind (i.e., two floating point units that can process two floating point operations in parallel). However, finding enough parallelism in a single instruction stream to keep a high-performance single-core processor busy is becoming more and more complex. One of the techniques to overcome this has been hyper-threading. This involves making a single physical core presented as two (or more) to the operating system. Running two threads allows exploitation of instruction-level parallelism.

**The power wall:** As we stated above, not only the number of transistors has been increasing but also their frequency. Yet there exists a technological limit to surface power density, and for this reason, clock frequency cannot scale up freely any more. Not only would the amount of power that must be supplied be unfeasible, but also the chip would not be able to dissipate the amount of heat generated. To address this issue, the trend is to develop simpler and specialised hardware and aggregate more of them (i.e., Xeon Phi, GPUs).

Nevertheless, computer technologists are also focussed on making exascale machines, but they cannot rely on increasing the performance of a single core as they used to. The workaround is that the number of cores and accelerators per chip and per node has grown fast in the recent years, along with the number of nodes in a cluster. This is pushing research into more complex memory hierarchies and network topologies.

Once exascale machines are available, the challenge is to have applications that can make efficient use of them by scaling to these levels. The increase in complexity of the hardware is a challenge for scientific application developers because their codes must run efficiently on more complex hardware and address a higher level of parallelism at both shared and distributed memory levels. This is where co-design is essential, and where we have invested time to adjust and update our codes to run efficiently on these future archetectures. Co-design implies a mutual effort of hardware developers, system integrators, and application developers to build future compute platforms together with performant software that runs on them. Being a domain-science centric CoE covering a broad range of applications, we see our main contribution on the software side. We will mainly address two aspects namely, support and guidelines for code adaptation and porting (software co-design) and performance modelling for future simulations that can be used for the design of new hardware and systems (hardware co-design). For both aspects, we need reliable data, which we obtain by setting up a scalable CompBioMed benchmark suite. It will cover the monolithic and coupled pre-exascale codes PALABOS, HemoCell, Alya, HemeLB as well as a set of MD codes such as GROMACS, NAMD, AMBER,

PU

Page 16

Version 1.1





and OpenMM, which are all used by the pre-exascale workflow BAC. Note that, for BAC, we have already achieved ports of ESMACS and TIES onto GPUs and hybrid workflows that use RADICAL-Cybertools (RCT) on Summit to fully exploit all cycles available on such hybrid machines. Scalability in this respect implies that for each application we provide different size-consistent input sets targeting single nodes to tier-0 machines that show similar characteristics such as memory consumption, communication, etc. with respect to one node across all scales.

At this point, only a global unified effort (hardware manufacturers, middleware and software developers, researchers, users, etc.) incorporating a co-design approach will enable exascale applications. Scientists in charge of HPC applications need to be able to trust in the parallel middleware and runtime libraries available to help them exploit the parallel resources. Additionally, completely new debugging and profiling tools must be designed and developed to cope with large-scale runs. The complexity and variety of the hardware no longer allows the manual tuning of the codes for each different architecture, requiring flexible programming languages and extensions. On this regard, a project like CompBioMed, partnered by stakeholders with all the possible viewpoints, is arguably the best forum to propose co-design strategies on all the aforementioned departments.

#### 7.5 Integration: the HPC-Cloud infrastructure

At this point it is clear that a seamless integration of computing infrastructures of different sizes in a cloud environment is a must. Exascale and cloud computing are not isolated technologies. Since a few years ago, and in an almost exponential reaction, almost all cloud providers are including HPC instances among their available offers (notably Amazon AWS, Microsoft Azure, Oracle Cloud Infrastructure or Google Cloud). Moreover, since very recently all of them provide tools to integrate the cloud provider's infrastructures with on-premises' data centers to create a unified environment.

Based on the pyramidal scheme of **Figure 1**, which describes the anatomy of a supercomputer, **Figure 2** pictures HPC-Cloud integration. Pyramids of different sizes (from single nodes to exascale supercomputers) are combined with a cloud orchestration, to which the user gains access through a web-based frontend. This integration combines the best of both worlds in a highly democratised fashion, making computer power accessible to the largest possible number of researchers. It is worth remarking that by facing exascale and cloud computing challenges simultaneously, CompBioMed is providing a large corpus of scientific and technology research about the deployment of simulation applications on HPC-Cloud infrastructure.





Figure 2. The HPC-Cloud computing structure.

Examples of HPC-Cloud integration can be found throughout the CompBioMed exemplars, as discussed in the Discussion section.

One of the main exascale computing patterns is concerned with ensemble computing, one which is advocated by many of our sister CoE as their surefire route to exascale. It is also a widespread modus operandi on modern supercomputers which is required for performance of validation, verification and uncertainty quantification (VVUQ) and for building applications based out of workflows of increasing complexity. Applications such as the IMPECCABLE workflow for drug discovery cannot be run on cloud resources – it is a workflow, itself the integration of four separate complicacted workflows, combining HPC with Machine Learning/Artificial Intelligence (ML/AI) methods. This is due to the speed at which we need/expect to obtain results and the data flow that is required to deliver them. An exascale machine which is optimised for such widespread usage patterns could well have different design features from ones which run extreme scale and coupled models. It will support the concurrent launching and execution of huge numbers of tasks (eventually up to millions), each task being potentially an MPI process itself. Today, many large supercomputers have hardwired limits to the maximum number of tasks that they can support, a direct legacy of the old model in which only monolithic applications were supported. Such complex workflow applications will put more stress on such a computer's parallel file system and less on indicidual interconnects than those in the monolithic and coupled compute patterns. This in no way means that the parallel compute pattern is not an exascale class application; rather it means the nature of the architecture that optimises its performance would be somewhat different from that on which the other patterns are optimised.

While the former is true in an academic environment, the advantages of cloud computing will be more relevant to commercial entities because the cost model and QoS issues are entirely different. Even in cases where one might claim that an application could be run on a cloud, it is not cost effective to any academic to do so. We do not have computer budgets comprised of real money and we get allocations on supercomputers,

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 18

#### \*\*\* \* \* \*\*\*

PU



not on clouds. That being said, there is a region in which the two modes overlap in order to allow certain applications, which may turn out to be ultimately destined in part for the cloud, to be developed initially on HPC architectures.

#### Co-design: The (yet) unreached holy grail

Before considering how to efficiently use the power available in future systems (mostly exascale machines, but others too), there are other, not so obvious considerations, that are of even larger relevance: will we be able to use it at all, even on a small scale? The core issue, therefore, is not if an application will be efficient but if it is even prepared to run on these future machines. For example, if exascale computers or cloud environments are made using an architecture in which the code can run. Preparations for exascale computing have led to the realization that future computing environments will be significantly different from those that provide petascale capabilities, not to mention High Performance Cloud. This change is driven by many issues, among them energy constraints, which are compelling architects to design systems that will require a significant re-thinking of the development and implementation of algorithms. Codesign has been proposed as a methodology for scientific application, software and hardware communities to work together [5]. HPC performance has improved by three orders of magnitude almost every decade, a fact made possible by technology scaling and advances in the system design (hardware and software) following an evolutionary approach. Next generation hardware has been delivered to software developers who ported and optimized the software stack to the delivered hardware. However, future computing facilities will be different, and the entire system architecture will be vastly transformed, and thus to realize the performance goal, a revolutionary approach is required with strong software and hardware co-design strategy at its centre.

According to Barret et al. [5] co-design considers the **entire system stack** from underlying technologies to applications (Figure 3). Applications provide insight into compute patterns and data movement patterns to optimize the system, the execution model implemented in the system software is tailored to provide services and manage resources, and the programming system hides the underlying hardware and provides programming productivity. From the bottom, technological issues and opportunities need to be comprehended, and in the middle, the system architecture effort need to devise the optimal system architecture considering top down requirements and bottom up issues and opportunities.







### Analysis of applications to devise the most efficient solutions



## Issues and opportunities to exploit

Figure 3. The co-design strategy, as stated in Barret et al. (see reference in the note)

In CompBioMed we develop applications for users, we do not build computers. Therefore, to make this strategy work, we need a kind of "social contract" amongst the different players. While hardware manufacturers rarely include application developers in their staff, application development groups need to be closely connected to hardware manufacturers and/or have hardware expertise internally, i.e, to know how the underlying hardware responds to their algorithms. This gap is even larger if we consider simultaneously both exascale hardware manufacturers and application researchers. There have been (and are) many collaborative efforts to fill this gap, favouring co-design, but to make this happen effectively will need a sustained commitment from all stakeholders. Regarding HPC cloud, the situation is somewhat better, because cloud providers take the role of "manufacturers" as they deploy the cloud facilities according to user needs. Even so, things should be improved. So far, any HPC hardware architecture becomes imposed from the manufacturer, while the buyer and developers must make the best of the situation, with the highly esteemed help of middleware and programming tool developers to fill the gaps as best as possible.

Even when conventional HPC system design involves a pipelined collaborative process that includes all the requirements throughout the design process, when it is delivered, 4-6 years later, the rapid and disruptive changes anticipated in hardware design over the next decade necessitate a more systematic and agile development process. This process could be the hardware-software co-design processes developed for rapid product development in the embedded space. Design methodologies on which we have relied so far, never had to consider power constraints or parallelism of the scale being contemplated for exascale systems. Furthermore, the programming model and software environment for future extreme-scale systems is anticipated to be substantially

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 20



ΡU

different from current practice. The designers of HPC hardware and software components have an urgent need for a systematic design methodology that reflects future design concerns and constraints.

The co-design strategy is based on developing partnerships with computer vendors (and cloud providers) and application scientists and engaging them in a highly collaborative and iterative design process well before a given system is available for commercial use. The process is built around identifying leading edge, high-impact scientific applications and providing concrete optimization targets rather than focusing on speeds and feeds (FLOPs and bandwidth) and percent of peak. Rather than asking "what kind of scientific applications can run on an Exascale system" after it arrives, instead we need this application-driven design process to ask, "what kind of system should be built to meet the needs of the most important scientific application performance in the design process is essential to define a common optimisation target that spans hardware and applications. Target application programs often consist of a million source lines of code involving multiple programming languages, third party library dependencies, and other complexities.

The Centres of Excellence are a very good environment to help implement co-design practices. Centres like CompBioMed puts together high- and low-level expertise that can provide hardware vendors and cloud providers the required knowledge to foresee future needs, if not to steer their production efforts. In our centres, even if we cannot provide the bottom-up arrow of Figure 3, we can go deeply into analysing the top-down arrow approach, providing indications that help the manufacturers to bring their powerful tools closer to the developers' needs, especially in configuring the exascale arena in a way that all its potential can be exploited. A decisive weapon to steer codesign is performance analysis tools, because they not only can help developers to improve software efficiency, but also, they can show manufacturers which problems we face that could be improved with architecture some changes. In the discussion section we provide several examples on the kind of information that such tools can provide.

In the next section, we highlight the top-down strategy co-design areas. With this information to hand, it is our mission to establish deeper links with vendors and providers discussions of the future of HPC computing. The supercomputing centres partnering CompBioMed are in a privileged position to streamline these communication channels.

#### 8 Discussion

In this section we will summarise the results so far attained by CompBioMed in porting applications to novel architectures and investigating the path to exascale computing. This joint effort will lead us to the final goal, the efficient use of HPC-Cloud computing resources. The results are grouped in sections devoted to each exemplar, with a final

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 21



PU



section focused on the important issue of Input/Output (I/O) and data access through visualisation in the HPC era.

Due to the variety of applications supported within CompBioMed, there cannot be one overriding metric to measure exascale performance. The compute pattern, expected outcome and application all need to be considered in order to determine the best measure for exascale potential. The indication of such metrics is an ogoing discussion between all CoEs and whilst this takes place we propose in the table below a list of metrics with their advantages and disadvantages:

Metric	Туре	Advantages	Disadvantages
Cores	Scale	Simple measure of number of cores deployed	<ul> <li>Assumes equivalence between CPUs and GPUs</li> <li>Reflects poorly on GPU codes</li> <li>HPC facilities have far fewer GPUs than CPUs</li> </ul>
Threads	Scale	Gives strong indication of total number of parallel processes utilised by a job	<ul> <li>Assumes a GPU thread and a CPU core are equivalent</li> <li>Reflects poorly on CPU codes</li> </ul>
Nodes	Performance	Simple measure of total nodes used by a job	<ul> <li>Resources available on a node varies widely between HPC facilities</li> <li>GPUs may not be available on all nodes</li> <li>Job may not demand all resources of a node</li> </ul>
Resource %	Scale	Indicator of how much of a computer's resources are being requested	<ul> <li>HPC policies may limit the maximum resources that can be requested</li> <li>Dependent on resource</li> </ul>

 Table 1: List of metrics for determining exascale potential with advantages and disadvantages of each.

PU

Version 1.1





			distribution between nodes
Wall time	Performance	Easiest measure to record	Geometry     dependent measure
Speed up	Performance	Straightforward measure to interpret performance	<ul> <li>Derived unit from wall time or MLUPS of FLOPS</li> </ul>
MLUPS	Performance	Geometry independent	<ul> <li>Measure that is most relevant to LBM codes</li> </ul>
Requested FLOPS	Scale	<ul> <li>Measures total available computer power</li> <li>Allows direct comparison between GPUs and CPUs</li> </ul>	<ul> <li>Peak FLOPS of given resource is always a theoretical value, actual proportion of this utilised depends on hardware specific optimisations</li> <li>Not a direct measure of parallel performance</li> </ul>
Load / Data balance	Performance	<ul> <li>Increases efficiency, especially in heterogeneous systems</li> <li>Measures compact and "fair" use of the resources</li> <li>Has a strong beneficial impact on energy consumption</li> </ul>	<ul> <li>Requires careful study of each software component, and how resources are used</li> <li>Requires thorough knowledge of target code</li> <li>Requires major effort in code re- engineering, from algorithms to implementation</li> <li>Geometry dependent measure</li> </ul>
Task start- up time	Performance	Is relatively invariant to scale, i.e., is largely independent of the number or the length of the tasks	<ul> <li>Dependent on the communications with resource managers and parallel</li> </ul>

Version 1.1





				programming libraries
Sustained number of tasks	Performance	Measures total number of tasks completed	•	HPC policies may limit the maximum resources to which tasks can be assigned Dependent on the stability of hardware

#### 8.1 Cardiovascular Exemplar

#### Cardiac fluid-structure interaction simulations in HPC-Cloud using containers (BSC)

In [3], the authors present a summary of three container solutions to evaluate their feasibility for HPC environments using Alya as the simulation code: Docker, Singularity and Shifter, considering that containerization could be a fantastic solution to port codes to large-scale system. In particular, the information published can be studied by container developers to adapt their software to, on one hand, our codes and, on the other hand, to larger systems up to the exascale arena. Alya is an example of an HPC code that can take part in the three multiscale patterns, making it very appealing to do such analysis.

From the analysis, we observe that Docker, the standard *de-facto* solution in cloud environments, presents some security and performance issues that make its adoption in HPC centers very unlikely, at least in its current state of development. On the other hand, Singularity is gaining attention in the HPC domain due to its performance, ease of use and integration with MPI and Slurm. The authors also looked at emerging solutions like Shifter, which seems to follow the trend of Singularity regarding its lightweight overhead but still has some issues concerning usability and portability to non-Cray systems. It is worth remarking that a deep analysis of applications containerisation can provide front line information for HPC cloud co-design because it is a clear way of highlighting our needs and steering their service provision.

A performance comparison of the three container technologies, done on a cardiac mechanics fluid-structure simulation problem, has shown that Singularity and Shifter can provide close to bare-metal performance in up to 112 MPI ranks. Docker performs worse with degraded simulation performance due to communication overhead. This overhead can be explained by the network virtualisation inherent to the Docker approach.







**Figure 4.** Scalability plot of an Alya case running on bare metal and on Singularity containers on MareNostrum IV. "Bare metal" means no use of containers. "Singularity Generic" means that the container image does not hold any host-specific features. "Singularity Host" means that the container image leverages the host's MPI high-performance network, with an integration of the container environment with the host MPI libraries. In this way, the container invokes the host's MPI taking advantage of whatever specific configuration is available. Taken from [3].

In this paper [3], the authors also consider containers as a portability solution to mitigate the divergence of architectures. They used three clusters with three different architectures to evaluate the performance of their biological use case. Moreover, they quantified the trade-off between portability and performance when building Singularity images. It was demonstrated that it is possible to build a generic container that is independent of the software stack on the host. But it was also shown that the performance of such a generic container is far from that obtained in bare-metal executions. On the other hand, they found that a container image built with the performance libraries available in the host can achieve a performance comparable to the bare-metal one.

In this light, the main conclusion is that, in the case of an HPC-based simulation code, it is possible to adapt the kind of containerisation to a wide range of scenarios, combining fast simulations with medium size and large-scale ones. In particular, the authors are interested in cloud deployment of our simulation code, Alya, which solves complex multi-physics / multi-scale problems, and in assessing its use within an HPC-Cloud environment. The goal is to use the code in the biomedical context running cell, tissue and organ level simulations of a given system (not only cardiovascular but also respiratory, etc.) in a simultaneous and combined way. As such, the authors are more interested in complex orchestrated sets of simulations of different computational cost than in a single and heroic large-scale run. While Docker allows a flexible solution for smaller simulations, Singularity provides great efficiency for larger ones. Combining the options in a smart way should provide a very well suited solution for cloud deployment of the kind of multi-scale / multi-physics problems we are attacking.



Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 25

PU



The paper presents scalability tests of a production biological simulation on up to 12,000 cores of a tier-0 cluster using containers in an unprecedented manner for HPC containers. With this test, the authors expose that container technologies can scale at the same rate as bare-metal if we sacrifice portability.

Since this was written in 2018, BSC has also worked on various optimisation aspects of the code, to ensure that it makes use of the latest supercomputing architectures, and is ready for the future architectures and potential exascale machines. Since its beginnings in 2004, Alya has scaled well in an increasing number of processors when solving single-physics problems such as fluid mechanics, solid mechanics, acoustics, etc. Over time, we have made a concerted effort to maintain and improve scalability for multi-physics problems. This poses challenges on multiple fronts, including: numerical models, parallel implementation, physical coupling models, algorithms and solution schemes, meshing process, etc.

Further details of scaling and the outlook for Alya can be found in the CompBioMed2 deliverable D2.1.

## Cardiac fluid-electro-mechanical model of the heart for supercomputers and its application to clinical, pharmaceutical and medical devices sectors (BSC, Oxford, UPF)

In this group of theses and papers [6–10], cardiac computational models, based either on BSC's Alya or Oxford's CHASTE, are used in an HPC context to simulate specific problems. In [6,7,10] a multi-physics / multi-scale fluid-electro-mechanical model of the human heart is presented, called the Alya Cardiac Computational Model (Alya CCM) and in [8,9] high definition simulations of electrophysiology including the torso are used to solve complex problems. This is an example of a coupled computing pattern, in which two or more large-scale parallel instances of the same code (Alya) are tightly coupled together. Moreover, together with another partner (UCL) we are studying the possibility of integrating this pattern with another large-scale code: HemeLB. Through these papers, the model shows its potential in biomedical research to become computational cardiac platforms, to study diseases, healing therapies and devices design and operation. In both models (especially Alya CCM) the tight coupling between the different structures and physics involved is fundamental to the problem solution, simulating cardiac function as a complex fluid-electro-mechanical system.

This research line focuses on three aspects: the model's physiological similarity, its computational complexity and its efficient implementation on supercomputers. Being a multi-scale / multi-physics system, coupling between electrophysiology, tissue mechanics and blood flow must be accurately yet efficiently modelled. The Alya based work also includes a 1D-3D coupling model to link the arterial network to the beating heart. This is currently being extended and improved in collaboration with UCL and Sheffield.

PU

Page 26

Version 1.1







**Figure 5.** Parallel performance of the Alya Red cardiac model. Speed up and efficiency for the computational fluid dynamics (CFD, blood) and computational solid mechanics (CSM, mechanics and electrophysiology) on up to 5000 core counts with a 10 million nodes problem. Extracted from [7].



**Figure 6.** Fully-coupled fluid-electro-mechanical simulation of a heart. The sequence shows the systole process of a third degree atrio-ventricular block and the action of a trans-catheter intra-ventricular pacemaker (the small red cylinder close to the heart apex). In this disease, the initial electrical stimulus is not delivered to one of the ventricles, producing a strong heart malfunction. Tissue is coloured by electrical activity and blood shows the so-called Q-criterion, which depicts blood flow vortices evolution.

	Extracted from [7].	
PU	Page 27	Version 1.1





#### Blood platelet aggregation simulations using data analysis (UNIGE)

A powerful way to attack multi-scale simulations deployed in HPC-Cloud environments is with data analysis techniques of all kinds, including genetic algorithms, machine learning and artificial intelligence. In their most complex form, these techniques can combine high- and low-resolution simulations with experiments of all kinds, to obtain the correct input parameters, to assess sensitivities of inputs or to create surrogate predictive models.

In [11] the authors introduce a model in which data analysis techniques are used to tackle the aggregation of blood platelets, which is part of the sequence of events leading to the formation of a thrombus (clot). The authors had previously developed a numerical model that quantitatively describes how platelets in a shear flow adhere and aggregate on a deposition surface [12]. Five parameters specify the deposition process and are relevant for a biomedical understanding of the phenomena. Experiments give observations, at five-time intervals, on the average size of the aggregation clusters, their number per mm<sup>2</sup>, the number of platelets and the ones activated per  $\mu$ l still in suspension. Then, by comparing in-vitro experiments with simulations, the model parameters can be manually tuned. Here, the authors use instead approximate Bayesian computation (ABC) to calibrate the parameters in a data-driven automatic manner. ABC requires a prior distribution for the parameters, which are taken to be uniform over a known range of plausible parameter values. As ABC requires the generation of many pseudo-data by expensive simulation runs, the authors have thus developed a highperformance computing (HPC) ABC framework, taking into account accuracy and scalability. The present approach can be used to build a new generation of platelets functionality tests for patients, by combining in-vitro observation, mathematical modelling, Bayesian inference and high-performance computing, deployed in an HPC-Cloud environment. In this example, while code optimisation was specially focused at the single node level, HPC-Cloud capabilities were put to a test, from deployment up to managing complex workflows for the optimisation process.



**Figure 7.** The deposition surface of the Impact-R device after 300 seconds (left) and the corresponding results of the deposition in the mathematical model (right). Black dots represent the deposited platelets that are grouped in clusters. Extracted from [11].

Page 28

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

#### PU



#### Large-scale intracranial vasculature blood modelling for magnetic drug targeting (UCL, UvA, UNIGE)

In [13], the authors tackle a multi-scale / multi-physics problem which addresses the issues arising as simulations approach the exascale range. The authors present an efficient computational model for simulating magnetic drug targeting in patient specific brain geometries, via the steering of paramagnetic nanoparticles with an external magnetic field. The model couples the dynamics of spherical particles to a lattice-Boltzmann hydrodynamics simulation, taking into account body forces (e.g. gravity), diffusivity, and dipolar interactions. A study of the model's computational performance found favourable results, with a performance drop of ~15% (relative to a simulation of the hydrodynamics alone, i.e. in the absence of any particles) in the most extreme case of load imbalance (all particles clustered in one region).

The code performance was assessed on up to 96,000 cores in EPCC's Archer supercomputer, a European system and on up to 250,000 cores on NCSA's Blue Waters, a US system, in another strong effort to push the limits to Exascale computing and CFD. In this regard, the authors report these two main issues and solutions:

- The existing optimisation and profiling tools are generally inadequate at core counts over 30,000 and only SCALASCA from POP CoE - including their support - was sufficient to reach these scales effectively.
- MPI-2 and -3, such as it is, still only use 32-bit communication and this is insufficient to manage the huge data movements on the machine when running at core counts of hundreds of thousands. HemeLB has become a "use case" for the MPI Forum and MPI-4 will be released in 2020 with clean 64-bit communications as a result. Work is now getting underway in support of this using BigMPI, planned for SuperMUC-NG.

The authors demonstrated the use of the model to predict the particle density (as a function of time) near a target site for a specific patient cerebral vascular system and heart rate, using a single point dipolar magnet. Through multi-scale coupling with a 1D representation of the wider vascular system, we obtained inlet velocity profiles for a patient in a range of physiological states (varying heart rate, cardiac output and mean blood pressure). Initial results allow confidence in the viability of the model to answer a wide range of questions relating to the design and manipulation of iron oxide nanoparticles in a clinical context. Comparison to phantom flow results and medical imaging research will allow further tuning of system parameters to further increase the accuracy of the model. A next step toward using the simulation technique in a more realistic manner will involve coupling of the flow solver to a comprehensive electromagnetic simulation. This will allow for the investigation of particle behaviour when exposed to more complex magnetic fields created by a combination of multiple electromagnets.

In this problem, the authors addressed the difficulties of coupling particle transport with continuum mechanics problems. Each of the two problems has its own parallelisation PU Page 29

Version 1.1



paradigm, which when coupled together, can become antagonistic facing efficiency. This is a typical difficulty of multi-physics coupling.



**Figure 8.** Volume rendering of the circle of Willis, constructed from an MRI scan of a human subject. The circle of Willis is the main blood distribution system in the brain, and is located roughly in the center of the head. Extracted from [9].

However, current effort is largely focussed on a new collaboration with a new associate partner, The Foundation for Research on Information Technologies in Society (IT'IS), to study blood flow in the entire human arterial tree, with eventual inclusion of the venous tree too. Efficient voxelization of the input meshes, memory optimization of HemeLB for a system of extreme sparsity, and development of load balancing schemes capable of scaling to 100k+ cores is necessary for such an endeavour, and we have made good progress on all fronts. This case is a very good example to help establish co-design ideas due to the different algorithms present, especially the coupled ones, and the different parallelisation schemes exposed.

Since this deliverable was written in 2018, further work has been conducted on this task in collaboration with the POP CoE. The collaboration was able to demonstrate strong scaling up to >300k cores (full machine scale) on SuperMUC-NG. Strong scaling has also been shown in a self-coupled version up to 30k cores with full human venous and arterial geometries.

In line with the upcoming pre-exascale machines expected at the end of 2021/beginning of 2022, a new version of HemeLB is in development that makes use of GPUs within the machines. This has been tested on Summit supercomputer and shown scaling of 90% up to 6144 GPU cores with continued strong scaling up to 18,432 cores. Given the early stages of this GPU-based codes, this shows an excellent outcome with the potential for further optimisation to come.

Version 1.1



"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 30



Porting and optimisation of a dense cellular suspensions flow application on novel and advanced microarchitecture Intel Skylake (BULL)

In [7], the authors develop a method which simulates mechanical models for red blood cells and reproduces the emergent transport characteristics of complex cellular systems. The computational code, Hemocell (high pErformance MicrOscopic CELlular Library), models the flow of blood at the cellular level. Blood plasma is represented as a continuous fluid simulated with the Lattice Boltzmann Method (LBM) while the cells are represented as discrete element method (DEM) membranes coupled to the fluid by the immersed boundary method.



Figure 9. Hemocell profile chart. Palabos performs CFD operations based on the lattice Boltzmann method (LBM). The HDF5 library is responsible for I/Os.

BULL's team performed the porting of the Hemocell HPC code on their computing platform. This platform offers several Intel Skylake nodes with the AVX-512 instruction set (i.e. 512 bits Advanced Vector Extensions). The first analysis of the application's profile shows that the application is divided mainly into calls to the LBM solver Palabos and I/Os using the HDF5 library (Figure 8). Therefore, considering these ratios in Figure 9, the LBM solver efficiency is responsible for the efficiency of the whole application.



**Figure 10.** Hemocell communication and execution time ratio and top MPI functions (Intel ITAC, profiling tool). This analysis shows a code which does not exploit OpenMP parallelisation and suffers from large MPI wait time.

Page 31

Version 1.1





The communication and execution time ratio and the MPI communication tracing quickly identified the main issues (see Figure 10). The large MPI time is caused by imbalance in the workload which can be mitigated using load balancing techniques.

FPU Utilization	FPU Utilization
SP FLOPs per Cycle	SP.FLOPs per Cycle
0.45 Out of 64.00	0.41 Out of 64.00
Vector Capacity Usage	Vector Capacity Usage 30.60%
FP Instruction Mix % of Packed FP Instr.: 0.00% % of 128-bit: 0.00% % of 256-bit: 0.00% % of 512-bit: 0.00% % of Scalar FP Instr.: 100.00%	FP Instruction Mix % of Packed FP.Instr.: 21.00% % of 128-bit: 0.20% % of <u>256-bit</u> : 0.30% % of <u>512-bit</u> : 20.50% % of <u>Scalar FP.Instr.</u> : <b>79.00%</b>
FP. Arith/Mem Rd Instr. Ratio	FP Arith/Mem Rd Instr. Ratio
0.43	0.17
FP. Arith/Mem Wr. Instr. Batio	FP Arith/Mem Wr Instr. Ratio
1.61	0.68

**Figure 11.** Floating point unit (FPU) utilisation (Intel Application Performance Snapshot). On the left, Analysis of the FPU utilisation with standard strong optimisation flags. On the right, FPU utilisation with previous optimisation and strained vectorisation.

The standard optimisation flags do not allow retrieval of the vectorisation capabilities of the CPU as shown in the left Figure 10. By forcing the vectorisation, the execution time of the application does not show any improvement. As said in the precedent section, the programmer must expose parallelisation at vectorisation level in order to make the compiler react. Therefore, and after this preliminary analysis, further investigations will be carried out, particularly on the data structure which very likely do not allow efficient vectorisation due to a non-unit stride (see Figure 11).



**Figure 12.** Hemocell's implementation of the data structure as Array of Structures (right). A Structure of Array (right) allows to get more efficiency from a computational point of view as it allows vectorisation.

The conclusion is that good performance can only be ultimately obtained by efficient algorithm implementation. Although some optimisations are done by tuning compiler flags, greater speedups are obtained by choosing the proper implementation strategy and data structure. Considering that current and future microarchitecture shall exploit parallelism at all levels as described in the introduction of this document, it is necessary to take advantage of these features in the code implementations. Further research will be performed following these lines. Having studied and improved the application performance with a hardware vendor partnering CompBioMed (Atos-Bull) is certainly a very good way of pushing co-design.



#### Load balance strategies for multi-physics problems in large-scale blood flow simulations (UvA)

The non-homogeneous distribution of computational costs is often challenging to handle in highly parallel applications, especially in multi-physics problems. In [14], the author studied the fractional load imbalance overhead in a high-performance biofluid simulation aiming to accurately resolve blood flow on a cellular level, using a methodology based on fractional overheads. In general, the concentration of particles in such a suspension flow is not homogeneous. Usually, there is a depletion of cells close to walls, and a higher concentration towards the centre of the flow domain, causing a time-dependent and potentially high computational work imbalance. We perform parallel simulations of such suspension flows. The emerging non-homogeneous cell distributions might lead to strong load imbalance, resulting in deterioration of the parallel performance. The authors formulate a model for the fractional load imbalance overhead, validate it by measuring this overhead in parallel lattice Boltzmann based cellbased blood flow simulations, and compare the arising load imbalance with other sources of overhead, in particular the communication overhead. They find a good agreement between the measurements and our load imbalance model. We also find that in our test cases, the communication overhead was higher than the load imbalance overhead. However, for larger systems, we expect load imbalance overhead to be dominant. Thus, efficient load balancing strategies should be further developed.



**Figure 13.** Haematocrit distribution for the channel flow case with different numbers of Red Blood Cells (N), average haematocrit H = 38%. 'small' (left), 'intermediate' (middle), and 'large' (right) systems. Extracted from [14].

Since the initial test cases, the communication overhead has been further investigated and optimisation work has been conducted on the code. A dynamic load-balancing mechanism has been employed which tracks the load-imbalance and if it exceeds a preset threshold, the simulation is checkpointed and restarted with a new decomposition that ensures homogenous load distribution. Whilst scaling is maintained at 25% over 8192 cores, further realistic cases are beginning which will result in a much larger scale study.

In parallel to these efforts, evaluation of a heterogeneous CPU-GPU version of the code has begun, where the cell mechanics (unstructured grid) are computed by the CPU, while the local fluid field (structured grid) computation is offloaded to node-local GPUs.





This is a necessary step to be able to exploit a larger part of the computational resources on emerging machines. While this development is in early stages yet, for a small realistic geometry we find an approximately factor 2 speed-up.

#### Increasing MPI communication efficiency for the HemoCell codebase (UvA)

For HemoCell, their developers have also looked into improving the efficiency of the communication between MPI processes. The communication of the cell material information between the processes was one of the major bottlenecks of the simulation, therefore, UvA researchers targeted this area by restructuring the communication pattern and restricting information exchange to data that is strictly necessary. Improving the efficiency in this context means that the communication structure needed to be altered, but not the resulting computation. A new step in the communication was added where instead of the fixed communication envelope a pre-compiled list of necessary information was used. This presents a minimal computational overhead that is counterweighted by the gain in reduced communication time.

The results are shown in Figure 14. By reducing the amount of data communicated and improving on the algorithms and data structures used, we managed to get an overall improvement of approx. 100% in wall clock time, and in the strongest scaled case we get an improvement of 350%. Furthermore, the strong scaling properties are improved as well. In practical scenarios this roughly means that when simulating blood flow in microfluidic chip for 1 second, the computation time is reduced from 10 days to around 3 days.





r of cores Figure 14. The top figure displays the performance of HemoCell before optimisation. The blue fluid bar encompasses communication as well as computation. The total time per iteration is written in black. The bottom figure displays the performance of HemoCell after the optimisation of the material communication. Both graphs are generated on different supercomputers top: SuperMUC, bottom: Marenostrum, therefore the difference between wall-clock times might not be due to optimisation, however, this should not affect the parallel efficiency numbers (green percentages).

128

(size of al

256 (64.32.32)

1024 (32.32.16)

2048 (32, 16, 16

32 (128,64,64)

(64,64,64)

#### 8.2 Molecularly-based Medicine Exemplar

#### Machine learning and large-scale computing (UPF)

CompBioMed directs its research to classical molecular dynamics (MD) simulations, which will be able to reach sampling in the second timescale within five years, producing petabytes of simulation data at current force field accuracy [15]. Notwithstanding this, MD will still be in the regime of low-throughput, high-latency predictions with average accuracy. In this paper, the authors envisage that machine learning (ML) will be able to solve both the accuracy and time-to-prediction problem by learning predictive models using expensive simulation data. On these grounds, such techniques can be considered as a post-process stage: the predictive model is built upon those expensive individual simulations. Apart from the research on the proper ML algorithms, the post-process stage presents more difficulties, such as Input / Output (I/O), storage and data analysis. The synergies between classical, quantum simulations and ML methods, such as artificial neural networks, have the potential to drastically reshape the way we make predictions in computational structural biology and drug discovery. This case also represents a potential for co-design of a different kind than the precedent ones, because it focuses on the efficiency of ML applications and MD.

PU

Page 35

Version 1.1





PU



Figure 15. Overview of a combined simulation and machine learning approach. a. MD data generation is expected to reach the second aggregated timescale by 2022 and an output files size of several petabytes by 2022 based on a trend of maximum aggregated time per paper per year using the ACEMD software.
b. A first example of ML replacing QM to predict dihedral energies given a neural network trained with QM simulations. c. An example of data augmentation by MD: augment protein-ligand binding poses for a set of protein-ligand pairs with unknown binding mode; augment binding affinity data for a set of resolved protein-ligand complex structures of unknown affinities. Extracted from [15].

#### Web-based application to support the preparation of protein structures (UPF)

Protein preparation is a critical step in molecular simulations that consists of refining a Protein Data Bank (PDB) structure by assigning titration states and optimising the hydrogen-bonding network. In [16], the authors describe ProteinPrepare, a web application designed to interactively support the preparation of protein structures. Users can upload a PDB file, choose the solvent pH value, and inspect the resulting protonated residues and hydrogen-bonding network within a 3D web interface. Protonation states are suggested automatically but can be manually changed using the visual aid of the hydrogen-bonding network. Tables and diagrams provide estimated pKa values and charge states, with visual indication for cases where review is required. The authors expect the graphical interface to be a useful instrument to assess the validity of the preparation, but nevertheless, a script to execute the preparation offline with the High-Throughput Molecular Dynamics (HTMD) environment is also provided for non-interactive operations.

Version 1.1

Page 36





Figure 16. Main screen with the results of the preparation. Extracted from 11.

#### Large-scale computing and binding affinity prediction of bromodomain inhibitors (UCL)

As applications of computational chemistry in biomedicine become more established it is increasingly important that results are reproducible and that the level of certainty associated with them is well defined. With respect to the exascale, it is highly likely that, even for those applications exhibiting excellent strong scaling characteristics, the tradeoff between resolving time or physical length scales in the system will frequently render such simulations inefficient on enormous core counts when compared to the weak scaling case (the use of multiple, so called replica, runs). We therefore expect that the actual impact of exascale resources on future science applications will be to encourage the use of uncertainty quantification (techniques that often require multiple runs) in a field where researchers too often only run large simulations once [17,18].

In this context we have developed simulation protocols and workflow tools that derive both results and associated error bars from ensembles of replica simulations. Within our binding affinity calculator tool (BAC) we have automated two ensemble binding free energy calculation protocols; ESMACS (enhanced sampling of molecular dynamics with approximation of continuum solvent) and TIES (thermodynamic integration with enhanced sampling)[17]. ESMACS is a faster but more approximate method, whereas TIES employs a more exact yet more expensive methodology. Thus the two protocols are designed to work in combination as drug discovery workflows move from hit to lead to lead optimisation phases.







Figure 17. Bromodomain inhibitor I-BET726 and its binding mode in BRD4-BD1 as studied by Wan *et al.*[13]. Two views are displayed for the binding mode (PDB ID: 4BJX), in which I-BET726 is represented as stick in cyan/blue/red/green, the protein is shown as cartoon in silver, the crystallographic water molecules are shown as red balls, and clipped protein surfaces are shown in orange.

The need to manage large campaigns of related simulations, where for example ESMACS rapidly rules out poor binders before TIES is employed when refining the best binders, has led us to enhance the simulation execution of BAC. To this end we have developed HTBAC [19], designed to manage complex workflows on multiple target HPC machines. Not only does this provide us with flexibility in terms of determining at run time which simulations should be continued based on their results (or the need to reduce uncertainty) but it allows us to design adaptive protocols that optimally use resources dependent on the sampling performance of individual simulations. Combining these two techniques with a range of computational kernels (BAC/HTBAC support the use of multiple MD engines, including those like ACEMD and OpenMM designed to fully exploit GPUs) will allow the efficient exploitation of the exascale to produce new scientific results with a greater level of robustness and reproducibility.

Our automated workflow and middleware development have allowed us to run simulations on the entirety of the SuperMUC supercomputer (>250,000 cores) run by the Leibniz Rechenzentrum, LRZ, near Munich, Germany. More recently, we were presented with the HTBAC the 2018 IEEE/ACM International Scalable Computing Challenge (SCALE) award [20] for our work in facilitating methods which allow automated trade-off between accuracy and computational cost. These developments are helping to prepare our protocols for the exascale an enable rigourous application of uncertainty quantification as we move towards the exascale. They have also facilitated our work using molecular dynamics simulations to machine learning techniques which has led to our award of a US DOE INCITE HPC leadership award, of 80 million core hours on the Titan supercomputer at the Oakridge National Laboatory in the US [21]. This application will help foresee the hardware needs of large-scale MD applications and help a better design of the future systems.

Since this has been written, UCL have also been working in a complementary study (IMPECCABLE) in which they have coupled Machine Learning (ML) and Molecular Dynamics (MD) techniques to increase throughput for drug discovery in light of the Covid-19 pandemic. For this BAC was used to perform free energy calculations on a

Version 1.1

\*\*\* \* \* \*\*\*

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 38

PU



variety of supercomputers including Leibniz Rechenzentrum's SuperMUC-NG, Hartree Centre's ScafellPike, Swiss National Supercomputing Centre's PizDaint, the University of Edinburgh's Advanced Computing Facility' Archer, as well as other supercomputers worldwide. We also use RADICAL-Cybertools (RCT) middleware for flexible task-level parallelism [9], which allows us to submit multiple jobs to the batch system like a single large job and is thus easier to schedule. BAC application uses an ensemble workflow, in which high-throughput "embarrassingly" parallel workloads are run on a wide range of node counts. The mean aggregated overhead of the middleware – RADICAL-Pilot (RP) – is typically below 5% of the execution time, making the scaling of the overall workflow very close to an ideal scaling (Figure 1).



Figure 1. Performance of binding affinity calculator (BAC) on different HPC facilities. Using a single job submission with RCT middleware, the number of compounds evaluated by CG-ESMACS increases linearly with the number of nodes up to the entire machines (see Table 1). The ideal scalabilities are shown as the dashed lines. The performance in practice, however, is often limited by allocation policies enforced on HPC facilities, which do not permit routine jobs at very large scale.

#### Computational Methods for Structure-Based Drug Discovery (Evotec, UCL)

The approach employing large-scale computing and binding affinity calculation of bromodomain inhibitors using ESMACS and TIES is also being applied to G proteincoupled receptors (GPCRs), the primary site of action of 60% of modern drugs and one of the most important and underexploited classes of current pharmacological targets [22]. A series of  $A_{2A}$  adenosine receptor ligands for which kinetic binding data from existing radioligand binding experiments existed is used. The purpose of the present study is to assess the use of ESMACS and TIES for the accurate and reproducible prediction of binding free energies in ~50kDa membrane-bound receptors, as opposed to ~18kDa globular bromodomains, and to use the BAC software tool and associated

Page 39

Version 1.1





services to improve industrial structure-based design approaches for novel GPCR therapeutics.



**Figure 18.** Top down view of the A2A receptor containing a) XAC, b) Theo, c) ZMA, d) NECA, e) UK, f) NGI in the binding pocket of the receptor. The coloring represents the ligand locations based on initial crystal structure (black), the cluster of the ligands' poses after the first 4 ns of production (blue), and the more favored/adopted ones (red) after the 20 ns of production. The  $\Delta$ G values represent the difference between the computed binding free energies, inclusive of entropy from ESMACS ( $\Delta G_{ESMACS}$ ) after the first 4 and last 4 ns of the production runs.

#### 8.3 Neuro-musculoskeletal Exemplar

In the period covered by this report and due to the character of the problems this exemplar deals with, the CompBioMed partners involved were mostly focused more on complex workflows development, image processing and clinical assessment of the results. Therefore, there is nothing to report, for this period, specifically related to HPC-Cloud computing.

#### 8.4 Input/Output and visualisation in the HPC era

In the three CompBioMed exemplars, input, output and the way data is analysed are key issues. Molecular dynamics, 3D flow features, muscular fibre contraction, particle transport or bone structure, are clear examples of the difficulties and the power of visualisation once the problematic issues are solved. Elegant yet efficient solutions are required, always keeping in mind the large-scale sizes of the resulting datasets. In this section we summarise and discuss the different strategies we employ to tackle the visualisation problems of CompBioMed exemplars. Additionally, I/O and visualisation is a decisive arena for the hardware co-design, because CompBioMed applications are not

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 40

PU



only greedy enough to provide a deep insight on the future system needs, but also because they expose a common requirement of large-scale applications.

As discussed in [1], scientific visualisation focuses on the creation of images to provide important information about underlying data and processes. In recent decades, the unprecedented growth in computing and sensor performance has led to the ability to capture the physical world in unprecedented levels of detail and to model and simulate complex physical phenomena. Visualisation plays a decisive role in the extraction of knowledge from these data—as the mathematician Richard Hamming famously said, "The purpose of computing is insight, not numbers..." [23]. Visualisation supports improved understanding of large and complex data in two, three, or more dimensions from different applications. In the kind of problems we are dealing with, visualisation is of great importance, as simulation results are often best represented in the timedependent three-dimensional form we are familiar with.

Traditionally, I/O and visualisation are closely related as, in most workflows, data used for visualisation are written to disk and then read by a separate visualisation tool. This is also called "post-mortem" visualisation, since the visualisation may be done after the simulations have finished. Other modes of interaction with visualisation are becoming more common, such as *in situ* visualisation (in which the simulation code directly produces visualisation images, using the same nodes and partitioning), or "in-transit" visualisation (in which the simulation program, possibly running on other nodes and with a different partitioning scheme).

**Input/Output.** Complex multi-physics, organ level simulations (cardiovascular, respiratory, etc.) writing files for post-mortem visualisation usually involve the highest volume of output. There are however other operations, especially explicit check-pointing at restart, that require the writing and reading of large datasets. Logging or output of data subsets also requires I/O, often with a smaller volume but higher frequency.

As these simulations can be quite costly, codes usually have a "checkpoint/restart" feature, allowing the code to output its state (whether converging for a steady computation or unsteady state reached for unsteady cases) to disk, for example, before running out of allocated computer time. This is called check pointing. The computation may be restarted from the state reached by reading the checkpoint from a previous run. This requires both writing and reading. Some codes use the same file format for visualisation output and check pointing, but this assumes data required are sufficiently similar and often that the code has a privileged output format. When restarting requires additional data (such as field values at locations not exactly matching those of the visualisation, or multiple time steps for smooth restart of higher order time schemes), code-specific formats are used.

**Parallel Input/Output.** There are several ways of handling I/O for parallel codes. The simplest solution is to read or write a separate file for each MPI task. On some file

PU

Page 41

Version 1.1





systems, this may be the fastest method, but it leads to the generation of many files on large systems, and requires external tools to reassemble data for visualisation, unless using libraries which can assemble data when reading it (such as VTK using its own format). Reassembling data for visualisation (or partitioning on disk) requires additional I/O, so it is best to avoid this if possible. Another approach is to use "shared" or "flat" files, which are read and written collectively by all tasks. MPI I/O provides functions for this (for example MPI\_File\_write\_at\_all using MPI), so the low-level aspects are quite simple, but the calling code must provide the logic by which data are transformed from a flat, partition- independent representation in the file to partition-dependent portions in memory. This approach provides the benefit of allowing check pointing and restarting on different numbers of nodes and making parallelism more transparent for the user, though it requires additional work for the developers. Parallel I/O features of libraries such as HDF5 and NetCFD seek to make this easier (and libraries build on them such as CGNS and MED can exploit those too).

Performance of parallel I/O is often highly dependent on the combination of approach used by a code and the underlying file system. Even on machines with similar systems but different file system tuning parameters, performance may vary. In any case, for good performance on parallel file systems (typical of shared file systems on modern clusters), it is recommended to avoid funnelling all data through a single node except possibly as a fail-safe mode. In any case, keeping data fully distributed extending to the I/O level is a key to handling very large datasets, which do not fit in the memory of a single node.

**Visualisation pipeline.** The "visualisation pipeline" is a common method for describing the visualisation process. When the pipeline is run through, an image is calculated from the data using the individual steps Filtering - Mapping - Rendering. The pipeline filter step includes raw data processing and image processing algorithm operations. The subsequent "mapping" generates geometric primitives from the pre-processed data together with additional visual attributes such as colour and transparency. Rendering uses computer graphics methods to generate the final image from the geometric primitives of the mapping process.

Regardless of the dimensionality of the data fields, any visualisation of the whole threedimensional volume can easily flood the user with too much information, especially on a two-dimensional display or a piece of paper. Hence, one of the basic techniques in visualisation is the reduction/transformation of data. The most common technique is slicing the volume data with cut planes, which reduces three-dimensional data to two dimensions.

Colour information is often mapped onto these cut planes using another basic wellknown technique called colour mapping. Colour mapping is a one-dimensional visualisation technique. It maps a scalar value to a colour specification. The scalar mapping is done by indexing into a colour reference table—the lookup table. The scalar values serve as indices in this lookup table including local transparency. A more general

Version 1.1

\*\*\* \* \* \* \*

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 42

PU

form of the lookup table is the transfer function. A transfer function is any expression that maps scalars or multidimensional values to a colour specification.

Colour mapping is not limited to 2D objects like cut planes, but it is also often used for 3D objects like isosurfaces. Isosurfaces belong to the general visualisation technique of data fields, which we focus on in the following.

**Visualisation of scalar fields**. Scalar fields are, for instance, pressure, temperature, electrical activation, ion concentration, etc. For the visualisation of three-dimensional scalar fields, there are two basic visualisation techniques: isosurface extraction and volume rendering.



**Figure 19.** Visualisation of flame simulation results (left) using slicing and color mapping in the background, and isosurface extraction and volume rendering for the flame structure. Visualisation of an inspiratory flow in the human nasal cavity (right) using streamlines colored by the velocity magnitude.

Isosurface extraction is a powerful tool for the investigation of volumetric scalar fields. An isosurface in a scalar volume is a surface in which the data value is constant, separating areas of higher and lower value. Given the physical or biological significance of the scalar data value, the position of an isosurface and its relationship to other adjacent isosurfaces can provide a sufficient structure of the scalar field.

The second fundamental visualisation technique for scalar fields is volume rendering. Volume rendering is a method of rendering three-dimensional volumetric scalar data in two-dimensional images without the need to calculate intermediate geometries. The individual values in the dataset are made visible by selecting a transfer function that maps the data to optical properties such as colour and opacity. These are then projected and blended together to form an image. For a meaningful visualisation, the correct transfer function must be found that highlights interesting regions and characteristics of the data. Finding a good transfer function is crucial for creating an informative image. Multidimensional transfer functions enable more precise delimitation from the important to the unimportant. Therefore, they are widely used in volume rendering for medical imaging and the scientific visualisation of complex three- dimensional scalar fields.

\* \* \* \* \* \* \* \* \*

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 43

ΡU



**Visualisation of vector fields.** Vector fields are, for instance, velocity, displacement, acceleration, magnetic or electric fields, etc. The visualisation of vector field data is challenging because no existing natural representation can convey a visually large amount of three-dimensional directional information. Visualisation methods for three-dimensional vector fields must therefore bring together the opposing goals of an informative and clear representation of a large number of directional information. The techniques relevant for the visual analysis of vector fields can be categorised as follows.

The simplest representations of the discrete vector information are oriented glyphs. Glyphs are graphical symbols that range from simple arrows to complex graphical icons, directional information, and additional derived variables such as rotation.

Streamlines provide a natural way to follow a vector dataset. With a user-selected starting position, the numerical integration results in a curve that can be made easily visible by continuously displaying the vector field. Streamlines can be calculated quickly and provide an intuitive representation of the local flow behaviour. Since streamlines are not able to fill space without visual disorder, the task of selecting a suitable set of starting points is crucial for effective visualisation. A limitation of flow visualisations based on streamlines concerns the difficult interpretation of the depth and relative position of the curves in a three-dimensional space. One solution is to create artificial light effects that accentuate the curvature and support the user in depth perception.

Stream surfaces represent a significant improvement over individual streamlines for the exploration of three-dimensional vector fields, as they provide a better understanding of depth and spatial relationships. Conceptually, they correspond to the surface that is spanned by any starting curve, which is absorbed along the flow.

Texture-based flow visualisation methods are unique means to address the limitations of representations based on a limited set of streamlines. They effectively convey the essential patterns of a vector field without lengthy interpretation of streamlines. Its main application is the visualisation of flow structures defined on a plane or a curved surface. The best known of these methods is the line integral convolution (LIC), which has inspired a number of other methods. In particular, improvements have been proposed, such as texture-based visualisation of time-dependent flows or flows defined via arbitrary surfaces. Some attempts were made to extend the method to threedimensional flows.

Furthermore, vector fields can be visualised using topological approaches. Topological approaches have established themselves as a reference method for the characterisation and visualisation of flow structures. Topology offers an abstract representation of the current and its global structure, for example, sinks, sources, and saddle points. A prominent example is the Morse-Smale complex that is constructed based on the gradient of a given scalar field.

Version 1.1

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"

Page 44

PU



Visualisation of tensor fields. Tensor fields are, for instance, velocity gradients, deformation tensor, stress tensor in all its modalities (Cauchy, Piola-Kirchoff,...), etc. Compared to the visualisation of vector fields, the state of the art in the visualisation of tensor fields is less advanced. It is an active area of research. Simple techniques for tensor visualisation draw the three eigenvectors by colour, vectors, streamlines, or glyphs.

In situ visualisation. According to the currently most common processing paradigm for analysing and visualising data on supercomputers, the simulation results are stored on the hard disk and reloaded and analysed/visualised after the simulation. However, with each generation of supercomputers, memory and CPU performance grows faster than the access and capacity of hard disks. As a result, I/O performance is continuously reduced compared to the rest of the supercomputer. This trend hinders the traditional processing paradigm.

One solution is the coupling of simulations with real-time analysis/visualisation—called in situ visualisation. This technique necessarily starts before the data producer finishes. The key aspect of real-time processing is that data are used for visualisation/analysis while still in memory. This type of visualisation/analysis can extract and preserve important information from the simulation that would be lost as a result of aggressive data reduction.

Various interfaces for the coupling of simulation and analysis tools have been developed in recent years, notably ParaView/Catalyst and VisIt/libSim. These interfaces allow a fixed coupling between the simulation and the visualisation and integrate large parts of the visualisation libraries into the program code of the simulation. Recent developments favour methods for loose coupling as tight coupling proves to be inflexible and susceptible to faults. Here, the simulation program and visualisation are independent applications that only exchange certain data among each other via clearly defined interfaces. This enables independent development of simulation code and visualisation/analysis code. It is worth mentioning that HemelB and Alya are already experimenting with in situ visualisation capabilities.

#### Conclusion 9

In this document we have focussed on two issues, which result in a combined strategy. On one hand, we reviewed the status of those codes from the CompBioMed software stack that are on the road to exascale computing. On the other hand, we looked at porting to architectures currently considered as "novel", notably Cloud Computing especially focusing on its burgeoning HPC capabilities. As stated above, the former, exascale, is the present "novel" architecture and the latter, HPC-Cloud, is a future "novel" architecture. We strongly believe that the combined situation, HPC-Cloud environments will be ideal platform for Software-as-a-Service in the years to come. With the experience gained in CompBioMed we are in a privilege position to establish a fluent PU Page 45

Version 1.1





dialogue with hardware vendors and cloud providers, helping that future facilities become closer (by conception) to the application needs.

This report is a selected compilation of the research in this area by CompBioMed partners, in papers published or on the way to being published, related to the three research exemplars, with a special section devoted to Input-Output and Visualisation.

It is worth mentioning that this report is complementary to D2.2 Report on Deployment of Deep Track Tools and Services to Improve Efficiency of Research and Facilitating Access to CoE Capabilities.

Page 46





#### **10** References

- 1. PRACE CSS. The Scientific Case for Computing in Europe 2018-2026.
- Houzeaux G, Borrell R, Fournier Y, Garcia-gasulla M, Göbbert JH, Hachem E, et al. We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists TOP 1%. InTechOpen. 2018; https://www.intechopen.com/books/computational-fluid-dynamics-basicinstruments-and-applications-in-science/high-performance-computing-dos-anddon-ts.
- Ruddy O, Garcia-Gasulla M, Mantovani F, Santiago A, Sirvent R, Vazquez M. Containers in HPC: A Scalability and Protability Study in Production Biological Simulations. In: 2018 International Parallel and Distributed Processing Symposium. 2019.
- Alowayyed S, Groen D, Coveney P V, Hoekstra AG. Multiscale Computing in the Exascale Era. J Comput Sci. 2016;abs/1612.0:(Submitted). http://arxiv.org/abs/1612.02467. 10.1016/j.jocs.2017.07.004
- Barrett RF, Borkar S, Dosanjh S, Hammond SD, Heroux MA, Hu S, et al. On the Role of Co-design in High Performance Computing. In: Volume 24: Transition of HPC Towards Exascale Computing. IOS Press; 2013. p. 141–55. 10.3233/978-1-61499-324-7-141
- 6. Lopez M. Multimodal ventricular tachycardia analysis: towards the accurate parametrisation of predictive HPC electrophysiological computational models. Universitat Politecnica de Catalunya; 2018.
- 7. Santiago A. Fluid-electro-mechanical model of the human heart for supercomputers. Universitat Politecnica de Catalunya; 2018.
- Lyon A, Mincholé A, Martínez JP, Laguna P, Rodriguez B. Computational techniques for ECG analysis and interpretation in light of their contribution to medical advances. J R Soc Interface. 2018 Jan 1;15(138). http://rsif.royalsocietypublishing.org/content/15/138/20170821.abstract.
- 9. Cardone-Noott L, Rodriguez B, Bueno-Orovio A. Strategies of data layout and cache writing for input-output optimization in high performance scientific computing: Applications to the forward electrocardiographic problem. Oliveira RS, editor. PLoS One. 2018 Aug;13(8):e0202410. 10.1371/journal.pone.0202410
- Garcia-Canadilla P, Dejea H, Bonnin A, Balicevic V, Loncaric S, Zhang C, et al. Complex Congenital Heart Disease Associated With Disordered Myocardial Architecture in a Midtrimester Human Fetus. Circ Cardiovasc Imaging. 2018 Oct;11(10). 10.1161/CIRCIMAGING.118.007753
- Dutta R, Chopard B, Latt J, Dubois F, Zouaoui B, Mira A. Parameter Estimation of Platelets Deposition: Approximate Bayesian Computation With High Performance Computing. Front Physiol. 2018; https://doi.org/10.3389/fphys.2018.01128. 10.3389/fphys.2018.01128
- Chopard B, Ribeiro de Sousa D, Latt J, Mountrakis L, Dubois F, Yourassowsky C, et al. A physical description of the adhesion and aggregation of platelets. R Soc PU
   Page 47
   Version 1.1
- "This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No **675451**"





Open Sci. 2017;4. https://royalsocietypublishing.org/doi/pdf/10.1098/rsos.170219. 10.1098/rsos.170219

- Patronis A, Richardson RA, Schmieschek S, Wylie BJN, Nash RW, Coveney P V. Modeling Patient-Specific Magnetic Drug Targeting Within the Intracranial Vasculature. Front Physiol. 2018; https://doi.org/10.3389/fphys.2018.00331. 10.3389/fphys.2018.00331
- Alowayyed S, Závodszky G, Azizi V, Hoekstra AG. Load balancing of parallel cellbased blood flow simulations. J Comput Sci. 2018 Jan;24:1–7. 10.1016/j.jocs.2017.11.008
- Perez A, Martinez-Rosell G, De Fabritiis G. Simulations meet Machine Learning in Structural Biology. Curr Opin Struct Biol. 2018;49:139–44. https://www.sciencedirect.com/science/article/pii/S0959440X17301069?via%3 Dihub.
- Martínez-Rosell G, Giorgino T, De Fabritiis G. PlayMolecule ProteinPrepare: A Web Application for Protein Preparation for Molecular Dynamics Simulations. J Chem Inf Model. 2017 Jul;57(7):1511–6. 10.1021/acs.jcim.7b00190
- Wan S, Bhati AP, Zasada SJ, Wall I, Green D, Bamborough P, et al. Rapid and Reliable Binding Affinity Prediction of Bromodomain Inhibitors: A Computational Study. J Chem Theory Comput. 2017;13(2):784–95. 10.1021/acs.jctc.6b00794
- Bhati AP, Wan S, Hu Y, Sherborne B, Coveney P V. Uncertainty Quantification in Alchemical Free Energy Methods. J Chem Theory Comput. 2018 Jun;14(6):2867– 80. 10.1021/acs.jctc.7b01143
- 19. Dakka J, Farkas-Pall K, Turilli M, Wright DW, Coveney P V, Jha S. Concurrent and Adaptive Extreme Scale Binding Free Energy Calculations. 2018 Jan;
- 20. Lab BN. Software Framework Designed to accelerate Drug Discovery Wins IEEE International Scalable Computing Challenge.
- 21. US Department of Energy INCITE Leadership Computing.
- Heifetz A, Southey M, Morao I, Townsend-Nicholson A, Bodkin MJ. Computational Methods Used in Hit-to-Lead and Lead Optimization Stages of Structure-Based Drug Discovery. In: Computational Methods for GPCR Drug Discovery. 2017. p. 375–94.
- Hamming R. Numerical Methods for Scientist and Engineers. 1962. https://epiportal.com/Ebooks/Numerical Methods for Engineers and Scientists.pdf.

Version 1.1

Page 48