Digital twin simulation development and execution on HPC infrastructures*

Marek Kasztelnik[1[0000−0002−8473−5236]], Piotr Nowakowski[1,2[0000−0002−2369−3685]], Jan Meizner[1,2[0000−0003−4094−6557]], Maciej Malawski[1,2[0000−0001−6005−0243]], Adam Nowak[2[0009−0001−7842−5522]], Krzysztof Gadek[2[0009−0001−9504−5484]], Karol Zajac[2[0000−0003−1393−8236]], Antonino Amedeo La Mattina[3,4[0000−0002−9927−2393]], and Marian Bubak[2[0000−0001−6083−0549]]

[1] ACC Cyfronet AGH University, Kraków, Poland
[2] Sano Centre for Computational Medicine, Kraków, Poland
[3] Department of Industrial Engineering, Alma Mater Studiorum - University of Bologna, Bologna, Italy
[4] Medical Technology Lab, IRCCS Istituto Ortopedico Rizzoli, Bologna, Italy

Abstract. The Digital Twin paradigm in medical care has recently gained popularity among proponents of translational medicine, to enable clinicians to make informed choices regarding treatment on the basis of digital simulations. In this paper we present an overview of functional and non-functional requirements related to specific IT solutions which enable such simulations – including the need to ensure repeatability and traceability of results – and propose an architecture that satisfies these requirements. We then describe a computational platform that facilitates digital twin simulations, and validate our approach in the context of a real-life medical use case: the BoneStrength application.

Keywords: Personalized medicine · Digital shadow · Digital execution environment · HPC

1 Motivation

Applications of the Digital Twin paradigm in medical care have recently gained popularity among proponents of translational medicine. A digital twin is typi-

* This work was supported by the EDITH, a coordination and support action funded by the Digital Europe program of the European Commission under grant agreement No. 101083771. This work was also supported by the European Union's Horizon 2020 research and innovation program under grant agreement Sano No. 857533 as well as the Sano project carried out within the International Research Agendas program of the Foundation for Polish Science, co-financed by the European Union under the European Regional Development Fund. This work was (partly) supported by the European Union's Horizon 2020 research and innovation program under grant agreement ISW No. 101016503. We also gratefully acknowledge Poland's high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH) for providing computer facilities and support within computational grant No. PLG/2022/015850.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

cally described as a virtual representation of a real-life process. In the context of medical interventions, it refers to a computational model which represents a specific patient, usually focusing on a specific ailment or pathological process, and enables a medical care professional to make informed choices regarding treatment on the basis of digital simulations. The model can fetch patient data from databases or from the IoT devices worn by the patient [30], in order to simulate the given ailment/organ/etc.

Based on our expertise connected with building patient digital twins on HPC we discovered a set of patterns (as well as antipatterns) for the calculations executed and data managed on supercomputers by researchers. (1) Researchers tend to try to run models as fast as possible and focus on results, often forgetting about model versioning and traceability. (2) Data is usually transferred to HPC from a local computer into the user's personal directory, which is not accessible by other team members. (3) When a calculation for a given use case is finished, other calculations are sometimes executed, which often leads to overwriting result data. As a result, it is difficult or even impossible for other researchers to recreate a specific experimental setup. Moreover, the problem of research sustainability is not new (see e.g. [33] and [27]) and there are even dedicated institutes that assist in performing sustainable research (e.g. [17]). The Cyfronet DICE team [5] has been involved in such initiatives for a long time [32]. We build tools that simplify the way data is managed, versioned, and made accessible to others.

The objective of this paper is to present an overview of functional and non-functional requirements related to specific IT solutions which enable simulations for Digital Twins in medicine – including the need to ensure repeatability and traceability of results. We propose an architecture that satisfies these requirements and promotes the principles of 3R[5]. Subsequently, we present the Model Execution Environment [20] – a reference implementation of our concept. We validate our approach in the context of a real-life medical use case: BoneStrength – an in silico trial solution for efficacy evaluation of treatments preventing proximal femur fracture. Given that we do not focus on interactions between the model and the real patient (instead relying on data fetched from external databases or IoT devices), the described solution can be treated as a digital user shadow, or digital execution environment.

2 Typical requirements of digital twins simulation

Digital twins are intended to stand in for the patient when medical simulations need to be carried out. Accordingly, the concept of a digital twin should be understood as a set of data which represents the specific patient in relation to a specific condition or treatment process, in particular circumstances. The data in question is not typically restricted in any manner - indeed, it may include unstructured textual data (such as measurement results), binary data (images and scans), structured repositories (databases storing patient information), or

[5] Repeatability, Replicability, Reproducibility

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

even free-text descriptions, such as those provided by medical practitioners who interact with the patient.

A system capable of performing digital twin simulations, must – unless geared for a specific procedure or disease type – be agnostic as to the supported classes and formats of data items. This is a primary requirement facing such systems, which, in turn, translates into the need to support (by means of integration) a comprehensive data repository, where various data items may be queried, retrieved and fed into the computational models which constitute the given simulation workflow.

In addition to the above, the platform must provide access to a computational infrastructure – given that, naturally, the very concept of digital twin simulations implies that data needs to be processed in some manner. Depending on the scope and specific aims of the simulation, a variety of specific requirements may emerge. First of all, the scale of the simulation must be taken into account. The following types of computational resources may be required:

- standalone servers (for small-scale simulations),
- cloud computing infrastructures (for simulations in which a moderately sized set of data is processed using complex algorithms),
- classical HPC (High Performance Computing) solutions such as computing clusters (for scale-out studies which involve processing large amounts of data and "parameter study" types of computations).

Additional functional requirements associated with digital twin simulations refer to the properties of the underlying computational infrastructure - both in terms of hardware and software. The following notable requirements are often encountered:

- availability of specific hardware components, mainly in the context of GPGPU processing,
- availability of specific software packages and libraries - which are often commercial and sometimes costly to use,
- ensuring security of data and computational models themselves - as digital twin simulations frequently process sensitive medical data, special arrangements must be made to guarantee that such data is protected against unauthorized access.

3 A critical review of platforms for digital twins

One of the important aspects that we have focused on was the thorough State of the Art analysis, to identify potential candidates for a digital twin platform. We focused on infrastructural as well as software aspects.

Relevant medical simulations may be run on a wide range of computational platforms, ranging from dedicated workstations, through local clusters to large-scale Cloud [31] and HPC Systems [26]. Each of those systems provides distinct features and challenges. As local workstations lack sufficient power for serious

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

computations, the preferred option is usually to utilize more sophisticated systems – which, due to their multi-tenant and shared nature, require appropriate integration with an execution platform.

The most straightforward type of infrastructure is a computational cloud, as while the platform is usually multi-tenant in nature (even for private clouds), the user usually obtains unrestricted access to a set of instances. However, not every model can be run in the cloud. Even though modern cloud providers offer dedicated solutions for batch or HPC-like use cases such as Azure Batch [4], virtualization overhead and the need for flexibility may nevertheless hinder performance when compared to purpose-built HPC systems.

The aforementioned HPC systems provide great performance in terms of computational power, storage and interconnect; however, they carry drawbacks in other areas that need to be addressed by the platform. Those are mostly related to ensuring secure access, which, while standardized to some degree, may vary from system to system (GSI-SSH [8], plain password-based SSH, key pairs, native API), as well as the multi-tenant nature of the system, which makes it more difficult to install additional end-user software, and also imposes the need to strictly control access to data.

Clearly, the infrastructure by itself is not sufficient to solve the problem of digital twin simulations. Another important aspect relates to the software components required to build an integrated simulation platform. Due to the complex nature of such applications, the most manageable way to express and run simulations is via a specialized workflow system [22].

An example of a workflow platform is the Arvados Workflow System [3]. It is based on the well-developed Common Workflow Language (CWL) [21] and provides a mechanism for integration with a wide range of aforementioned infrastructures both cloud- and HPC-based; however there are two significant drawbacks. First and foremost, integration with HPC requires significant modification of the HPC cluster itself, including deployment of tools and services both on the login and compute nodes. For large-scale clusters, this may be infeasible for operational and/or security reasons. Secondly, even though Arvados supports Singularity (now referred to as Apptainer) containers [2] commonly used for HPC, it is clearly stated that this support is still experimental and Docker should be used instead. However, Docker is not commonly encountered on multi-tenant HPC systems for security reasons. It is also important to mention that Arvados requires some form of containers (Docker or Singularity/Apptainer) to deploy workflows on the HPC cluster, and cannot operate on standalone jobs.

Another large and well-known workflow system is Pegasus [23]. The system is robust and offers the ability to run computations on a wide range of infrastructures including grids, clusters and clouds. It also provides advanced features such as provenance and error recovery. However, while the above features are desirable, we have decided that at the current stage the overall complexity of this solution is too great given the requirements of the Digital Twin Platform. On top of that, we would still need to implement domain-specific features that are not present in this generic platform.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

Additionally we have analyzed the High-Performance Computing and Cloud Computing with Unified Big-Data Workflows platform created in the scope of the LEXIS Project [25]. The platform integrates HPC and Cloud computing enabling multi-system and multi-site deployment of workloads as well as provides required utilities such as orchestration, billing, AAI, API and Portal. HPC access is enabled by custom middleware called HEAppE [9] that utilizes SSH and SCP/Rsync protocols for running jobs on clusters. The Cloud component utilizes Alien4Cloud [1] with extended TOSCA [19] templates for deployment to platforms such as OpenStack [13]. Regarding data, iRODS and EUDAT/B2SAFE [6] are used. Access is secured by the Keycloak [11] based AAI system where accounts are mapped to cluster accounts, as well as appropriate custom integration with iRODS [24]. While the overall work done in the scope of the LEXIS Project is impressive and we believe that the platform may be useful for large multi-site deployments involving both HPC and Clouds the significant deployment overhead, which involves installation of custom middleware (both on HPC and cloud compute sites, as well as on storage sites), along with a dedicated AAI system, is too great given the stated needs of our research. Moreover, as in the case of the Pegasus system described above, the platform is not dedicated to the medical domain and would require extensions to be fit for such purpose. Finally, we have found that one of the crucial components, Alen4Cloud, is no longer maintained according to its official website (since July 2022); thus, the user would need to provide their own maintenance services, or find a substitute - which might require significant effort as this component lies at the core of the platform.

4 The concept of a universal platform for digital twins

Our aim was to create a platform for digital twin simulations that may be used by scientists to submit their computations to e-infrastructures such as HPC clusters in an easy and accessible way. This concept is presented in Figure 1. Users interact with the platform through a web browser after being authenticated and authorized by an appropriate IDP, which also can generate required credentials (such as a user grid proxy certificate, to be used for access to the e-infrastructure). When the simulation is started by the user, all required code, along with input data, is automatically transferred to the infrastructure (such as an HPC cluster). The platform monitors the status of the simulation and notifies the user when it is finished. It also provides a way to download and compare results online.

Our conceptual work on the presented platform was guided by the following principles:

- **Model versioning** - previous versions of the model are stored and may be referred to if needed
- **Repeatable runs (3R vision)**
  - **Repeatability** - a researcher can reliably repeat their own computation provided the conditions are the same (same team, same experimental setup)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

- **Replicability** - an independent group can obtain the same results using the authors' own artifacts (different team, same experimental setup)
- **Reproducibility** - an independent group can obtain the same result using artifacts that they develop completely independently (different team, different experimental setup)

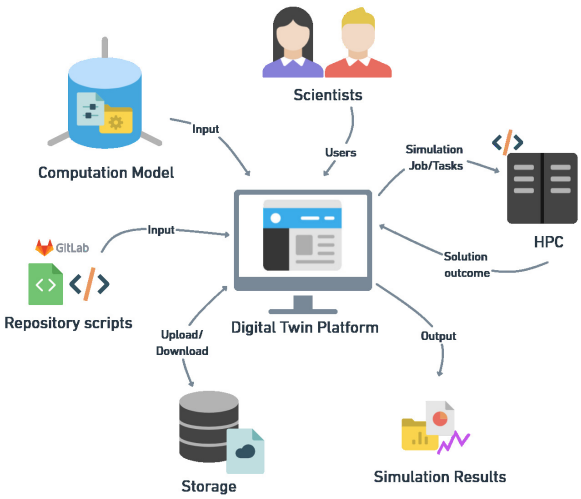


Fig. 1. General concept of the platform.

Following up on the above vision, we also adhered to the key requirement to enable a group of scientists with various backgrounds (such as medicine, physics, chemistry and computer science) to take part in digital simulation-driven experiments via a coherent and manageable system. As mentioned above, the system should enable execution of computational models controlled by a set of scripts with a versioning system, on the one hand enabling collaborative editing, while on the other hand tagging specific versions that may be later selected to suit the researchers' needs. Those models need to operate on data stored in a storage backend appropriate for the compute infrastructure. Another important goal is to streamline access to said compute infrastructures, which involves abstraction of the underlying APIs such as remote access mechanisms and system tools (including queuing tools). In addition to the above, the platform needs to provide a straightforward way to display, download and analyze simulation results.

The platform should also provide a mechanism to reuse models while supporting custom artifacts, thus realizing our 3R vision by producing consistent results regardless of conditions (teams, artifacts, setup).

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

5 Overview of implementation

The Model Execution Environment (MEE) is a prototype implementation of the architecture described in the previous section. It was deployed [12] and it is used to validate core digital twin creation and modeling concepts in the scope of applications from a range of projects, including EurValve [7] (human heart simulations), PRIMAGE [14] (Neuroblastoma, Diffuse Intrinsic Pontine Glioma tumor growth simulations), and In Silico World (ISW) [10] (in silico trials for bone fracture risk prediction, along with various other models). Below we discuss implementation details and list the advantages and disadvantages of the proposed tool.

MEE is a specialized high-level service to manage data and computations in the context of a patient cohort. It is integrated (via GSI-SSH and Proxy Certificate delegation, with automatic proxy generation using the OpenID mechanism) with several HPC clusters available with the scope of the PLGrid infrastructure[6], delivering access to 8 PFlops of computing power and multiple petabytes of storage. The platform follows the architecture described in Section 4 and presented in Figure 1. The main goal of MEE is to hide the complexity of the underlying infrastructure (HPC) as well as introduce a unified way for patient/case data to be stored and maintained. It promotes the following principles:

- **Model versioning** with the integration of git repositories. Users can simply push code to a repository and then run their calculations using the specified model version. Managing and transferring model code to HPC is done automatically.
- **Repeatable runs** achieved through integration with git repositories. Each run record model stores a version (git SHA) which can be used to rerun the same calculation. As a result, we can meet 3R (repeatability, replicability, reproducibility) criteria.
- **Integration with HPC.** MEE is integrated with the PLGrid infrastructure which allows us to delegate user rights from MEE to the HPC supercomputers.
- **Organized way to store patient data and the calculation results.** The main building block of the MEE environment is a single patient. Each patient has a dedicated storage space on a storage resource (e.g. HPC, S3) where patient data is stored. Inside, we also have a space for calculation-specific inputs and results. Patients can be grouped into cohorts upon which simulation campaigns can be executed (a campaign involves running the same pipeline for each member of a given cohort). This unification represents another step towards realizing the 3R concept.

The user interacts with MEE by using a local web browser (no further dependencies need to be installed on the user's machine). The first step is for the MEE

[6] The PLGrid infrastructure is a joint effort of the largest HPC centers in Poland. It offers coherent management of users, groups and computational grants, as well as unified access to the integrated HPC clusters [28].

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

administrator to define a set of simulations - which we call **steps** (see Fig. 2), by providing credentials enabling access to the simulation's code repository. Each step can define a list of required inputs needed to start the calculation. Steps can be grouped into **flows**. A flow is a template used to build a pipeline for a specific patient/case. The patient data structure defines the location where the personalized input data is stored. It contains a list of pipelines executed on the patient data. Each pipeline defines the output location and a list of computations executed on HPC. Each computation needs to be configured before it can be started (a specific version of the simulation, as well as the required input parameters, need to be selected).

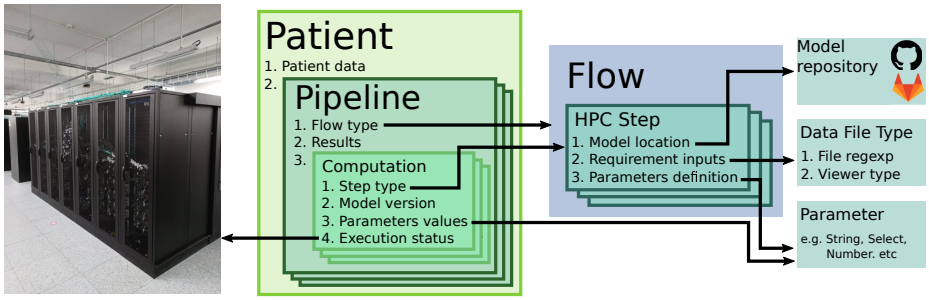


Fig. 2. Data structure used in MEE.

Pipelines consist of separate computations which are called pipeline steps. In most cases, these are pieces of software (e.g. Matlab scripts, CFD simulations, etc.) executed on the HPC cluster. Each pipeline step can be configured for integration with a collaborative source control project. For this purpose, MEE is integrated with the Git versioning systems (e.g. `github.com`, `GitLab.com` services can be used). There, MEE users are able to apply the typical features of a sophisticated source control tool to collaboratively develop, share and test their code (e.g. a simulation). Inside the repository, the template of the HPC queuing configuration script, as well as the rest of the simulation source code, should be stored. In order to launch a computation on HPC, the user needs to select which model version should be used. The model version is taken from the Git repository (MEE shows all repository branches and versions). When this value is selected, the queuing configuration template file is downloaded from the repository and converted to a final queuing system configuration file (in our case, a Slurm [16] startup script) dedicated for the selected case. To enable customization, MEE delivers a set of helpers which can be applied in the queuing configuration script in order to customize it. Notable helpers are briefly listed below:

- **clone_repo** injects code responsible for cloning the simulation repository in the version specified by the user.
- **stage_in input-file-type** searches for the simulation input file in the results of past computations, pipeline input, and patient input directories.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

– **stage_out file-path** uploads simulation results to the pipeline output directory.
– **value_of parameter-key** injects the value specified by the user while starting the calculation.

Below we present an example queuing configuration template script taken from the PRIMAGE project. It runs an agent-based simulation which is a hybrid model pertinent to the tissue level: patches of the whole tumor. The simulation comprises a continuous automaton representing the microenvironment, discrete agents representing neuroblasts and Schwann cells occupying the microenvironment, and a centre-based mechanical model for resolving cell-cell overlap.

```
1  #!/bin/bash -l
2  #SBATCH -N 2
3  #SBATCH --ntasks-per-node=2
4  #SBATCH --mem-per-cpu=3GB
5  #SBATCH --time=04:00:00
6  #SBATCH -A {% value_of grant_id %}
7  #SBATCH -p plgrid-gpu
8  #SBATCH --gres=gpu:2
9
10 {% clone_repo %}
11
12 module load plgrid/apps/cuda/10.1
13 module load plgrid/tools/gcc/8.2.0
14
15 nvidia-smi
16
17 {% stage_in amb-input amb-input.json %}
18
19 ./amb-input Models/Prototype_v2.0/ABM13.4/FGPU_NB --in amb-input.json --
       primage amb-output.json
20
21 {% stage_out amb-output.json %}
```

Listing 1.1. Example queuing configuration template script.

Lines 2 to 8 define the required resources for the simulation. In line 6 we can see how to inject the value of the parameter selected by the user while starting the simulation. The `clone_repo` directive (line 10) injects code to clone the simulation repository in the version selected by the user. The certificate used to clone the repository is registered in the Gitlab/Github as a deploy key. It has only read capability, thus the computation cannot push any modification the the repository. In line 17 we request input for the simulation which should be downloaded from the platform. Next, the simulation is started, and in line 21 we request upload of simulation results to the pipeline output directory. Once the simulation begins, the template is converted to a customized script, specific for the selected patient and pipeline:

```
1  #!/bin/bash -l
2  #SBATCH -N 2
3  #SBATCH --ntasks-per-node=2
4  #SBATCH --mem-per-cpu=3GB
5  #SBATCH --time=04:00:00
6  #SBATCH -A plgprimage4
7  #SBATCH -p plgrid-gpu
8  #SBATCH --gres=gpu:2
9
10 export SSH_DOWNLOAD_KEY="-----BEGIN RSA PRIVATE KEY-----
```

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----END RSA PRIVATE KEY-----
"
ssh-agent bash -c '
  ssh-add <(echo "$SSH_DOWNLOAD_KEY");
  git clone git@gitlab.com:primageproject/Models
  cd `basename cyfronet/Models .git`
  git reset --hard c0719f1c7e2990554821181a505d056a837eb236'

module load plgrid/apps/cuda/10.1
module load plgrid/tools/gcc/8.2.0

nvidia-smi

curl -o "Models/Prototype_v2.0/ABM13.4/amb-input.json" "https://mee.s3p.cloud
    .cyfronet.pl/production/patients/case2341/inputs/amb-input.json?X-Auth-
    Secrets=xxxxxxx"

./amb-input Models/Prototype_v2.0/ABM13.4/FGPU_NB --in amb-input.json --
    primage amb-output.json

curl -T amb-output.json "https://mee.s3p.cloud.cyfronet.pl/production/
    patients/case2341/pipelines/32/outputs/amb-output.json?X-Auth-Secrets=
    xxxxx"
```

Listing 1.2. Customized for the patient and pipeline queue system starting script.

To run the pipeline on the HPC cluster the user simply needs to log in to the system and click the run button. Underneath, during the login process, MEE asks the PLGrid IDP[7] about the user's grid proxy certificate, which is later on saved in an encrypted form in the MEE database. The Grid proxy certificate enables user rights delegation and it is used to submit Slurm jobs to HPC by using the Rimrock service [15]. MEE monitors job execution and notifies the user about the result. Once the calculation is finished, the user can click a link to download results. Whenever a new pipeline is launched, a separate storage space is created. As a result, we can be sure that outputs are never overwritten.

In order to perform the same calculations on multiple patients, MEE supports cohorts and campaigns (see Fig. 3). A cohort provides a way to group patients. Once it is created, the user can schedule a campaign: this creates a pipeline with a shared configuration for each cohort patient. The difference between these pipelines is that each one has inputs dedicated to a specified patient and produces outputs in a dedicated storage space. Once the campaign ends, the user can inspect or download results.

[7] PLGrid identity provider, which is capable of generating user proxy certificates that delegate user rights to the HPC infrastructure.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

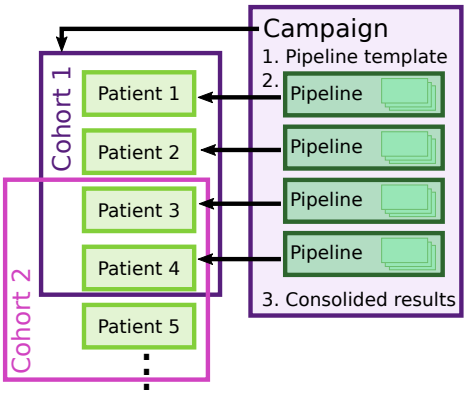


Fig. 3. The cohort groups patients and allows execution of the same set of calculations for each cohort member in the scope of the campaign.

6 Example of usage - the BoneStrength application

We will demonstrate the usage of the MEE platform on the basis of BoneStrength - an in Silico Trial solution for efficacy evaluation of treatments preventing proximal femur fracture. It is one of the two Fast Application Track solutions in the ISW project and consists of two main models.

- A Finite Element (FE) model, which predicts bone strength with a patient-specific model in a side fall condition as a function of the direction of impact.
- A stochastic patient-specific model of the side fall, which predicts the probability distribution of the impact force that a large number of random falls would cause in a certain patient.

The standard approach is patient-driven and it was born as a Digital Twin solution. The simulation computes 28 falls (by varying the falling load direction in the antero-posterior and medio-lateral directions, in order to explore all possible fall configurations) per year of a single patient and then estimate the average risk of fracture. A validated strain-based failure criterion allow predicting the femoral strength, which means the force causing bone failure. By calculating the ratio between the number of simulated falls that caused a fracture and the total number of simulated falls, and considering the falling annual rate, the Absolute Risk of Fracture at time zero (e.g., at the time the CT scan was performed) (ARF0) can be calculated [18]. Another approach, called "Markov" version of BoneStrength, is a simulation that aims to estimate the number of fractured patients over certain observation time. The very first phase is to generate series of falling along 10 years and randomly assign their occurrence to the patients. Each single case simulation will mark the patient as already fractured if the falling lead exceeds the femur fracture load. This indicates that the next simulations will not be executed for this patient. As it is a stochastic process, a bunch of realizations – named campaign – are required to obtain an average result.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

FE model simulations use ANSYS Mechanical APDL as solver, with MPI parallelism to speed-up the computations. Each execution requires around 5-6 GB of disk space, 20 GB of RAM and 4-8 cores, and needs 1-2 core-hours to be completed. Since multiple configurations for many patients need to be simulated, approximately 100 000 executions in total are intended to be processed including both BoneStrength model versions. In this case, array jobs are used for submitting multiple runs at the same time. In general, patient-driven simulations are characterized by the association with a specific subject or case described with its patient-related inputs. When it comes to large-scale realizations, where a set of patients – named cohort – is going to be simulated under the same scenario, but with different parameters, researchers must be able to identify the single runs in order to detect most interesting results (e.g., find patient and parameters that lead to bone fracture). The concept of the BoneStrength workflow is presented in Fig. 4.

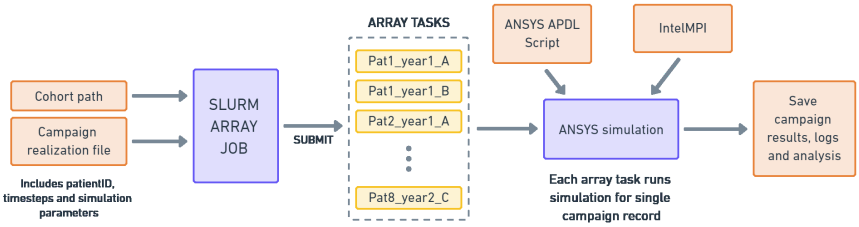


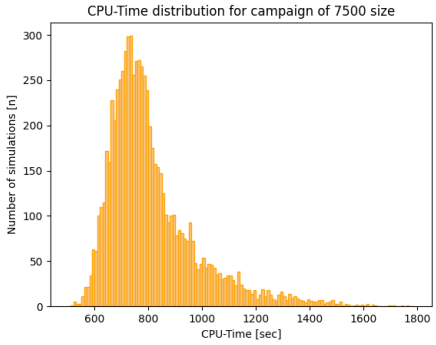
Fig. 4. Simulation workflow of the Markov BoneStrength version.

7 Evaluation - assessment of proof of concept

The presented architecture and MEE technology were used to simulate two Markov versions of the BoneStrength application on a cohort of 1080 patients, running a total of 7500 simulations as a campaign per model version. Each submitted campaign task allocated 28 GB of RAM and 4 CPUs, used ANSYS software and parallelized simulations with IntelMPI. The campaign was run on Prometheus Cluster at Cyfronet. The resource usage presented in Figure 7 indicates that each thread used in the ANSYS solver used an average of 813 CPU-seconds and 6.3 GB of RAM. The Slurm mechanism reports over 90 % of job efficiency, which is the ratio of the total useful runtime of a job to the wall-clock time requested by the user, expressed as a percentage value. This shows that almost the entire CPU time for all allocated cores was utilized during the job's elapsed time – which, in turn, indicates that HPC resource wastage is kept to a minimum.

The observed distribution of campaign CPU time allocation reflects the usual distribution resulting from different inputs/meshes of patients' bones or the difficulty of finding a solution by the ANSYS software. Some reported outliers are a

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2



	CPU-Time	Memory used	Efficiency
Mean	813.2 sec	6448.9 MB	94.5 %
Median	769.8 sec	6447.0 MB	92.9 %

Fig. 5. HPC infrastructure resource usage for campaign run of 7500 simulations of BoneStrength. The figure presents distribution of simulation's CPU time, average memory usage and HPC job efficiency regarding the provided allocation.

consequence of using an external ANSYS license server which sometimes forced a reconnection during high load times. Thus, the course of the campaign was limited to 400 simultaneously running jobs in order to avoid overuse of software licenses.

As a part of the ISW project, the conducted research allowed us to prepare initial sensitivity analyses for the BoneStrength solution, obtaining result files from bone analysis and fracture risk prediction. The same concept of a Digital Twin simulation is going to be applied in other BoneStrength model versions and workflows, as well as other patient-specific medical solutions.

8 Conclusions and future work

The presented architecture and its reference implementation (the MEE platform) have been validated in the course of three projects (EurValve, PRIMAGE and InSilocoWorld) implementing medical applications from different fields of the VPH domain. The feedback received from applications running on the platform enabled us to identify platform elements which should be improved, as well as new features which should be implemented in MEE. Additionally, based on the Model Execution Environment, a separate platform was created for the purpose of the PROCESS Project [29] called the Interactive Execution Environment (IEE). Given its reliance on an advanced version of MEE, it provides a comprehensive set of infrastructural features. Moreover, the IEE features a mechanism for running Singularity containers which can be useful for provisioning of models. In the near future we intend to change the way patient, pipeline, and computation data is stored to support easy usage traceability (who used the data, who produced the results, etc.) We will also investigate how to introduce a more generic abstraction of the computational process, which will enable support for additional computational platforms such as native Kubernetes containers.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

References

1. Alien4cloud. https://alien4cloud.github.io, accessed: 2023-04-18
2. Apptainer - the container system for secure high-performance computing. https://apptainer.org, accessed: 2023-04-11
3. Arvados workflow system. https://arvados.org, accessed: 2023-04-11
4. Azure batch. https://azure.microsoft.com/en-us/products/batch, accessed: 2023-04-11
5. Distributed computing environments (dice) team. https://dice.cyfronet.pl, accessed: 2023-04-11
6. Eudat collaborative data infrastructure. https://www.eudat.eu, accessed: 2023-04-18
7. Eurvalve: Personalized decision support for heart valve disease. https://eurvalve.sites.sheffield.ac.uk, accessed: 2023-04-11
8. Gsi-ssh. http://grid.ncsa.illinois.edu/ssh, accessed: 2023-04-11
9. Heappe middleware. https://heappe.eu/web/, accessed: 2023-04-18
10. In silico world: Lowering the barriers to a universal adoption of in silico trials. https://insilico.world, accessed: 2023-04-11
11. Keycloak open source identity and access management. https://www.keycloak.org, accessed: 2023-04-18
12. Model execution environment. https://mee.cyfronet.pl, accessed: 2023-04-11
13. Openstack. https://www.openstack.org, accessed: 2023-04-18
14. Primage, medical imaging, artificial intelligence, childhood cancer research. https://www.primageproject.eu, accessed: 2023-04-11
15. Rimrock: Robust remote process controller controller. https://rimrock.plgrid.pl, accessed: 2023-04-11
16. Slurm workload manager. https://slurm.schedmd.com, accessed: 2023-04-11
17. Software sustainability institute. https://software.ac.uk, accessed: 2023-04-11
18. Bhattacharya, P., Altai, Z., Qasim, M., Viceconti, M.: A multiscale model to predict current absolute risk of femoral fracture in a postmenopausal population. Biomechanics and Modeling in Mechanobiology **18**(2), 301–318 (2019). https://doi.org/10.1007/s10237-018-1081-0, http://link.springer.com/10.1007/s10237-018-1081-0
19. Brogi, A., Soldani, J., Wang, P.: Tosca in a nutshell: Promises and perspectives. In: Villari, M., Zimmermann, W., Lau, K.K. (eds.) Service-Oriented and Cloud Computing. pp. 171–186. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
20. Bubak, M., Czechowicz, K., Gubała, T., Hose, D.R., Kasztelnik, M., Malawski, M., Meizner, J., Nowakowski, P., Wood, S.: The eurvalve model execution environment. Interface Focus **11**(1), 20200006 (2021). https://doi.org/10.1098/rsfs.2020.0006, https://royalsocietypublishing.org/doi/abs/10.1098/rsfs.2020.0006
21. Crusoe, M.R., Abeln, S., Iosup, A., Amstutz, P., Chilton, J., Tijanić, N., Ménager, H., Soiland-Reyes, S., Gavrilović, B., Goble, C., Community, T.C.: Methods included: Standardizing computational reuse and portability with the common workflow language. Commun. ACM **65**(6), 54–63 (may 2022). https://doi.org/10.1145/3486897, https://doi.org/10.1145/3486897
22. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems **25**(5), 528–540 (2009). https://doi.org/https://doi.org/10.1016/j.future.2008.06.012, https://www.sciencedirect.com/science/article/pii/S0167739X08000861

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2

23. Deelman, E., Vahi, K., Rynge, M., Mayani, R., da Silva, R.F., Papadimitriou, G., Livny, M.: The evolution of the pegasus workflow management software. Computing in Science Engineering **21**(4), 22–36 (2019). https://doi.org/10.1109/MCSE.2019.2919690
24. García-Hernández, R.J., Golasowski, M.: Supporting keycloak in irods systems with openid authentication. presented at cs3—workshop on cloud storage synchronization and sharing services. `https://indico.cern.ch/event/854707/contributions/3681126`, accessed: 2023-04-18
25. Hachinger, S., Golasowski, M., Martinovič, J., Hayek, M., García-Hernández, R.J., Slaninová, K., Levrier, M., Scionti, A., Donnat, F., Vitali, G., Magarielli, D., Goubier, T., Parodi, A., Parodi, A., Harsh, P., Dees, A., Terzo, O.: Leveraging High-Performance Computing and Cloud Computing with Unified Big-Data Workflows: The LEXIS Project, pp. 159–180. Springer International Publishing, Cham (2022)
26. Jadczyk, T., Malawski, M., Bubak, M., Roterman, I.: Examining Protein Folding Process Simulation and Searching for Common Structure Motifs in a Protein Family as Experiments in the GridSpace2 Virtual Laboratory, pp. 252–264. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
27. Katz, D.S.: Fundamentals of software sustainability (2018), `https://danielskatzblog.wordpress.com/2018/09/26/fundamentals-of-software-sustainability/`
28. Kitowski, J., Wiatr, K., Dutka, Ł., Szepieniec, T., Sterzel, M., Pajak, R.: Domain-Specific Services in Polish e-Infrastructure, pp. 1–15. Springer International Publishing, Cham (2014)
29. Meizner, J., Nowakowski, P., Kapala, J., Wojtowicz, P., Bubak, M., Tran, V., Bobák, M., Höb, M.: Towards exascale computing architecture and its prototype: Services and infrastructure **39**, 860–880 (Jan 2021), `https://www.cai.sk/ojs/index.php/cai/article/view/2020_4_860`
30. Minerva, R., Lee, G.M., Crespi, N.: Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. Proceedings of the IEEE **108**(10), 1785–1824 (2020). https://doi.org/10.1109/JPROC.2020.2998530
31. Nowakowski, P., Bubak, M., Bartyński, T., Gubała, T., Harężlak, D., Kasztelnik, M., Malawski, M., Meizner, J.: Cloud computing infrastructure for the vph community. Journal of Computational Science **24**, 169–179 (2018). https://doi.org/https://doi.org/10.1016/j.jocs.2017.06.012, `https://www.sciencedirect.com/science/article/pii/S1877750317307330`
32. Nowakowski, P., Ciepiela, E., Harężlak, D., Kocot, J., Kasztelnik, M., Bartyński, T., Meizner, J., Dyk, G., Malawski, M.: The collage authoring environment. Procedia Computer Science **4**, 608–617 (2011). https://doi.org/https://doi.org/10.1016/j.procs.2011.04.064, `https://www.sciencedirect.com/science/article/pii/S1877050911001220`, proceedings of the International Conference on Computational Science, ICCS 2011
33. Venters, C., Jay, C., Lau, L., Griffiths, M., Holmes, V., Ward, R., Austin, J., Dibsdale, C., Xu, J.: Software sustainability: The modern tower of babel. CEUR Workshop Proceedings **1216**, 7–12 (01 2014)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_2