A Web-Based Service Portal to Steer Numerical Simulations on High-Performance Computers

Alice Grosch[1], Moritz Waldmann[2,3], Jens Henrik Göbbert[1], and Andreas Lintermann[1,3]

[1] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Júlich, Germany
A.Grosch@fz-juelich.de
WWW home page: https://www.fz-juelich.de/jsc

[2] Institute of Aerodynamics and Chair of Fluid Mechanics, RWTH Aachen University, Wüllnerstr. 5a, 52062 Aachen, Germany

[3] JARA Center for Simulation and Data Science, Seffenter Weg 23, 52074 Aachen, Germany

Available online 30 November 2020.
The original article can be accessed via https://doi.org/10.1007/978-3-030-64610-3_8

Abstract. Benefiting and accessing high-performance computing resources can be quite difficult. Unlike domain scientists with a background in computational science, non-experts coming from, e.g., various medical fields, have almost no chance to run numerical simulations on large-scale systems. To provide easy access and a user-friendly interface to supercomputers, a web-based service portal, which under the hood takes care of authentication, authorization, job submission, and interaction with a simulation framework is presented. The service is exemplary developed around a simulation framework capable of efficiently running computational fluid dynamics simulations on high-performance computers. The framework uses a lattice-Boltzmann method to simulate and analyze respiratory flows. The implementation of such a web-portal allows to steer the simulation and represents a new diagnostic tool in the field of ear, nose, and throat treatment.

Keywords: High-performance computing, service portal, computational fluid dynamics, respiratory flows, lattice-Boltzmann method

1 Introduction

X-rays, computed tomography (CT), magnetic resonance imaging (MRI), sonography, etc. offer excellent possibilities for the immersive imaging of bone structures, organs, or blood flow. Unfortunately, there is no similar immersive technique that makes respiratory flow, i.e., the flow in the nasal cavity, throat, trachea, down to the lungs easily and reliably in-vivo visible. Punctual measurements, e.g., with a 4-phase rhinomanometer (4-PR) [8] already offer a good and quick orientation, but a precise view of the entire flow cannot be achieved in this

way either.
An effective method to overcome this issue is to calculate the flow on the basis of the measurable variables by means of numerical simulations. Therefore, CT image data can be used to generate realistic in-silico models of the nasal cavity [7]. 4-PR yields the temporally resolved pressure difference between the nares and the pharynx for complete respiration cycles. This information can be used to accurately simulate respiratory flow at high resolution. A high resolution in space and time is mandatory to account for the complexity of the internal flow. That is, it is necessary to accurately predict (i) the smallest eddies, (ii) heat and moisture transport, (iii) aeroacoustics, and (iv) shear layers and wall-shear stresses. This, however, comes at the price of high computational power that is required to yield meaningful and detailed analysis results.
In science, high-performance computing (HPC) is commonly employed to get answers to computationally intensive tasks. Access to HPC systems on the scale required for solving medical flow problems in a reasonable time is, however, limited to domain scientists with expertise in HPC and simulation science. To use such methods in everyday medical practice or in medical research, the necessary time-consuming preparatory work has to be overcome and a user-friendly interface has to be offered.

This manuscript presents a software-technical solution to ease HPC access and HPC usage to answer rhinological questions with the help of numerical simulations. In the following, first the state-of-the-art in interactive supercomputing is described in Sec. 2. Subsequently, Sec. 3 discusses the challenges to operate an HPC service. Finally, the technical implementations to realize such a service are described in Sec. 4 before some conclusions are drawn in Sec. 5.

2 State-of-the-art in interactive supercomputing

From the user's point of view, the classical access to supercomputers has continuously developed over the last decades. However, the basic form of a text-based terminal has been maintained in the HPC environment until today. Regardless of the field of application, this ensures the greatest possible flexibility and automation of processes for the HPC expert. A large number of commands can be combined in different programming languages, thus enabling workflows to be mapped and computing resources to be used in the best possible way. Despite its flexibility, the classic terminal access with its high number of commands and their very specific requirements leads to a high error rate, when used by an inexperienced user. That is, a steep learning curve challenges beginners in using such systems.
Significantly easier to use are portals that hide the complexity of the HPC environment to a certain extent and thus prevent or intercept faulty operation. Depending on the level of abstraction, these portals have been developed for different levels of knowledge. The more general the portal is, the larger the possible user group. The Jülich Supercomputing Centre (JSC) service UNICORE [1], for

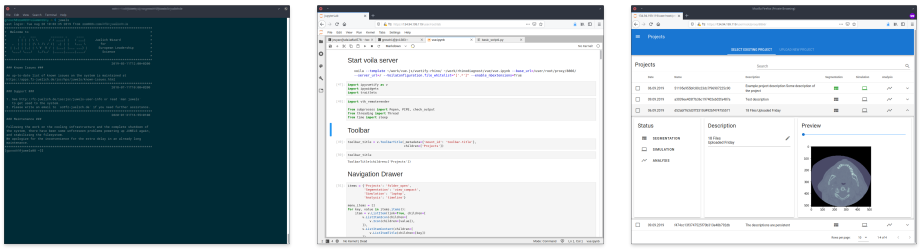

Fig. 1. The decreasing complexity of accessing an HPC environment. From the complex terminal (left) to the simplified interface of JupyterLab (middle) to a web application for even the most inexperienced user (right).

example, offers a web portal that provides general access to JSC's supercomputers. Nevertheless, in order to reach a large user community, a certain degree of HPC expertise is expected.

For the creation of web applications, several back-end web frameworks exist. The scripting language PHP is used for popular websites such as Google, Yahoo and Wikipedia [9]. It provides several popular frameworks such as Laravel[4] and CakePHP[5]. Since PHP was originally designed for web development, it remains largely unused in applications unrelated to web development. Although initially only implemented for client-side executions in a browsers, JavaScript now also provides runtime environments to execute code on the server side. Popular representatives are for example Node.js[6] and its framework Express[7]. Ruby on Rails[8] is a server-side web application framework written in Ruby and used, e.g., by Yahoo and Twitter.com. Other programming languages, which are also popular outside of the field of web development, include Java and Python. Java frameworks such as Spring[9] and the Google Web Toolkit[10] have made it easy to generate web applications from Java code. For Python, frameworks such as Django[11] and Flask[12] have made the creation of web applications possible without the need to dive into the depth of web development.

3 Challenges of service establishment and operation

Special application scenarios and a large number of users usually justifies the development of service web portals targeting specific tasks. However, the devel-

[4] Laravel https://laravel.com/
[5] CakePHP https://cakephp.org/
[6] Node.js https://nodejs.org/en/
[7] Express https://expressjs.com/
[8] Ruby on Rails https://rubyonrails.org/
[9] Spring https://spring.io/
[10] Google Web Toolkit http://www.gwtproject.org/
[11] Django https://www.djangoproject.com/
[12] Flask https://www.palletsprojects.com/p/flask/

opment can be very time-consuming and frequently requires further expertise such as in web development. Continuous adaptation and further development pose special challenges to the operation of such a service. A tight cooperation between web developers, HPC experts, and users must be present to guarantee successful service operation and advancement. Such a close cooperation is in research and development, due to different work focuses, project interests, and spatial separation of experts from HPC, web development, and medicine, a challenge in itself. To achieve a high flexibility and sustainability of the software, the sometimes conflicting requirements of these three expert groups were considered in the design and implementation phase of the respiratory flow web service.
From JSC's perspective, a simple and transparent integration into the existing infrastructure is desired. This integration needs to fulfill the requirements from the user administration, authentication, and authorization. Furthermore, the operating schemes and usage rules of the supercomputer facility, software environment, and data storage need to be adapted. These different areas of HPC are, however, subject to rapid and continuous changes. A service web portal must hence be designed in such a way that it resides several layers above fundamental supercomputing operation layers. This way it can flexibly adapt to and kind of changes on the other layers. Unlike the HPC expert, the web developer has different requirements. A maximum freedom in the design and in the choice of the underlying web technologies of the portal is a key to a successful implementation. It is the objective to create a modern, intuitive interface that can quickly be adapted to changing specifications. A close coupling to the underlying HPC layers should be avoided under all circumstances. From the point of view of the user / medical professional a clear, intuitive interface is also of high importance. This is, however, not so much a question of modern design but rather an issue of usability, i.e., a smooth operation and fast operability needs to be ensured.
In the context of the specific application considered here, it is the aim to establish an open service web portal that can be used by both the clinical and research community and includes experts and newcomers at the same time. Experienced users from HPC and physicians both need to have the opportunity to adapt and extend the web portal independently with new functions matching their requirements.

4 An HPC-based web service

In order to build a suitable and user-friendly interface to an HPC environment, corresponding software needs to be built around the core of an application – in this case a simulation framework. The framework considered here is the Zonal Flow Solver (ZFS), developed at RWTH Aachen University, which will subsequently be described in Sec. 4.1. Details on the technical implementations of the service environment follow this in Sec. 4.2.

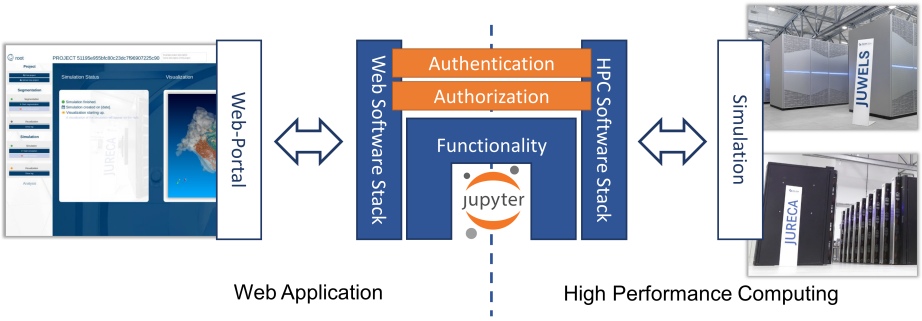

Fig. 2. The web-based service portal (left) enables the use of scientific simulation codes on state-of-the-art supercomputers (right) without special HPC knowledge via a multi-layer software stack based on Jupyter, Unity IdM and UNICORE.

4.1 Simulation method

The simulation framework implements a massively parallel grid generator [4] that is able to generate hierarchical Cartesian computational meshes with hundreds of billion of elements on $\mathcal{O}(10^5)$ HPC cores in a short amount of time. It solves the governing equations of fluid motion using a thermal lattice-Boltzmann (TLB) method [5–7], which is highly scalable and can compute complex flows in intricate geometries efficiently on HPC machines.

In fluid mechanics, the complexity of a flow is commonly characterized by the Reynolds number Re, which represents the ratio of inertial to viscous forces. In general, the higher the Reynolds number, the more it is necessary to increase the resolution to fully capture all necessary flow phenomena. For turbulent flows, the number of mesh elements scales with Re^3. Despite the Reynolds number is in respiratory flows quite small, i.e., for normal respiration at rest it is in general $Re < 2,500$, it is the complex geometry of the nasal cavity, which necessitates HPC computations. For example, Lintermann et al. [3] have shown, that meshes consisting of a minimum of approx. $100 \cdot 10^6$ elements with a sub-millimeter resolution of $\delta x < 0.1\ mm$ are necessary to achieve grid convergence. Especially the accurate prediction of flow separation, recirculation zones, shear layers, wall-shear stress, and heat-flux distributions necessitate high spatial and temporal resolutions. In particular the latter two, which are necessary to evaluate the forces acting on the tissue and to simulate the heat distribution correctly need fine meshes in the vicinity of the tissue surface in the nasal cavity.

Without any algorithmic parallelization, such simulations would be unfeasible. Fortunately, the TLB method can easily be parallelized, i.e., the problem can be split into many sub-problems, each being computed individually on a core of an HPC cluster. Transient simulations easily need computing resources in the range of $\mathcal{O}(10^4)$ compute cores to derive at a solution in a suitable amount of time. It should be noted that with code scalability at hand, the time-to-solution can drastically be reduced. To this end, what comes at the price of high

computational costs turns out to deliver reliable results that have the potential to be integrated into clinical diagnostics.

4.2 Technical implementation of a service portal

The web-based service portal provides a user interface to simulation pre-processing, the numerical simulation itself, and to the post-processing of the results and their visualization. In more detail, the service portal is responsible for the following tasks, which are made available to the user in a clearly arranged dashboard:

(i) upload of a CT data set and subsequent visualization
(ii) segmentation and visualization of a generated 3D geometry
(iii) configuration of a simulation
(iv) starting and monitoring a simulation on a supercomputer
(v) evaluation and visualization of the results in 2D and 3D

The upload mechanism was implemented in Python and JavaScript as the JupyterLab extension DICOM-upload[13]. It allows the CT data to be in the digital imaging and communications in medicine (DICOM) format. The connection from the client to the server side is encrypted using the hypertext transfer protocol secure (HTTPS). The data is pseudo-anonymized by a JavaScript algorithm on the client side and separately encrypted according to state-of-the-art encryption technologies. The data is subsequently uploaded to the storage system of the supercomputer. In addition, the files are protected against unauthorized access by the regulations of the user administration of the operating system. The encryption and decryption keys are stored on a separate system with advanced access control.
After the upload process finishes, the DICOM files are forwarded to a separate application for segmentation. The segmentation does not need HPC resources. But in order to avoid sending sensitive data over the network to other systems, the segmentation is, however, carried out on the same system as the subsequent simulation.
Usually, simulations are configured by HPC and/or simulation experts. Here, the service portal takes care of this step and provides the workflow with a suitable configuration file. A pre-defined job script is then sent to the scheduler SLURM[14] for batch submission on the supercomputer[15]. The portal's web interface is designed to avoid unnecessary queries and to allow intuitive operation and usage. HPC expertise is no longer necessary. Depending on the load of the supercomputer, the simulation might start with a time delay. The monitoring functionalities provide the user with additional features such as checking the estimated job execution time, the convergence of a simulation, or the visualization of simulation snapshots.

[13] DICOM-upload https://pypi.org/project/dicom-upload
[14] SLURM Workload Manager https://slurm.schedmd.com
[15] The scheduler manages the resources on an HPC system and takes care of priorities and job execution.

After simulation the results are evaluated and/or visualized. Some data processing methods require HPC resources and need to operate on large amounts of data, e.g., operations that need to be executed at each simulation time step. Temporal averaging or the computation of the Reynolds stress tensor are examples of such operations. In such cases, it is reasonable to utilize the simulation framework itself for data processing, which can directly benefit from having the data already in memory. That is, data can be accessed and processed fast in parallel instead of taking a detour over disk. Processing in this case also means the possibility to view the simulation results highly parallel by means of in situ visualization and the help of ParaView/Catalyst[16]. The Python library pvlink[17] was developed to integrate this technology in the Service Portal.
Finally, the service portal presents the results in the form of classic two-dimensional plots, cross-sections, or three-dimensional visualizations. A special implementation allows the user to interactively render three-dimensional high-resolution simulation results on an HPC systems. This renders the visualization mostly independent from the performance of the user's local computer. It also allows for full sharpness of detail even with limited network bandwidth.
The service portal is based on the open-source software JupyterLab[18], which provides both the backend on the HPC system side and the frontend on the web browser side. At JSC, JupyterLab, JupyterHub[19], Unity IdM[20], and UNICORE are combined to yield full flexibility. The authenticated and authorized user has direct access to a JupyterLab via his web browser. Data security is a key element of the service portal. The Identity Manager Unity Idm is used for authentication. The authorized access the software stack and to data on the HPC systems is controlled via UNICORE. The communication over network between the individual services is encrypted. The trust between the services is secured by certificates. Details on the technical implementation are presented in [2].
JupyterLab runs in the realm of the user on the HPC systems, which allows for individual accounting of the consumed compute resources. It provides a general environment for experts who need the greatest possible freedom to develop efficient workflows. The design and functionalities of the service portal is based on the Jupyter extension voilà[21] to provide the user with a professional dashboard-like desktop. This solution allows the portal developer to develop frontend functionalities in the programming language Python, which is widely used in HPC. That is, adaptation and extension of the portal can quickly and securely be implemented by HPC experts or experienced users.

[16] ParaView/Catalyst https://www.paraview.org/insitu
[17] pvlink https://pypi.org/project/pvlink
[18] JupyterLab https://jupyterlab.readthedocs.io
[19] JupyterHub https://jupyterhub.readthedocs.io
[20] Unity IdM https://www.unity-idm.eu/
[21] voilà https://voila.readthedocs.io

5 Conclusion

Numerical computations that use HPC systems to solve complex flow physics have the potential to tremendously advance clinical diagnostics in the field of rhinology. They come, however, with the disadvantage that interdisciplinary expertise in HPC and simulation science is required to run them. To overcome this issue, service portals that hide the complexity of supercomputing from non-experts such as physicians need to be developed. Such portals must be able to follow the rapidly changing and continuously developing field of HPC. At the same time they need to offer a clearly structured web interface tailored to the application scenario. The service portal developed in this manuscript shows that the different requirements of HPC and medicine can be mapped flexibly and in a future-oriented manner. Therefore, it features a consistent modular structure based on voila, JupyterLab, JupyterHub, Unity Idm, and UNICORE. This way, HPC, numerical simulations, and pre- and post-processing tools are made easily accessible for the physician. Obviously, the development also requires an intensive involvement of medical experts.

Disclosure statement. The authors declare that there is no conflict of interest.

Acknowledgments. The work presented in this manuscript has been performed within the project Rhinodiagnost, which is funded as a ZIM (Zentrales Innovationsprogramm Mittelstand - Central Innovation Programme for small and medium-sized enterprises)[22] project by the Federal Ministry for Economic Affairs and Energy (BMWi)[23] in Germany. The Austrian partner Angewandte Informationstechnik Forschungsgesellschaft mbH is funded by COIN (Cooperation and Innovation)[24], Federal Ministry of Science, Research and Economy (BMWFW)[25]. The project runs under the auspices of IraSME (International research activities by SMEs)[26].

References

[1] Erwin, D.W., Snelling, D.F.: UNICORE: A Grid Computing Environment. pp. 825–834 (2001)
[2] Göbbert, J.H., Kreuzer, T., Grosch, A., Lintermann, A., Riedel, M.: Enabling interactive supercomputing at jsc lessons learned. In: High Performance Computing. pp. 669–677. Springer International Publishing (2018)
[3] Lintermann, A., Meinke, M., Schröder, W.: Fluid mechanics based classification of the respiratory efficiency of several nasal cavities. Computers in Biology and Medicine 43(11), 1833–1852 (2013)

[22] ZIM https://www.zim.de/ZIM/Navigation/DE/Meta/Englisch/englisch.html
[23] BMWi https://www.bmwi.de/Navigation/EN/Home/home.html
[24] COIN https://www.ffg.at/coin-cooperation-innovation
[25] BMWFW https://www.bmbwf.gv.at/en.html
[26] IraSME https://www.ira-sme.net

[4] Lintermann, A., Schlimpert, S., Grimmen, J., Günther, C., Meinke, M., Schröder, W.: Massively parallel grid generation on hpc systems. Computer Methods in Applied Mechanics and Engineering 277, 131–153 (2014)
[5] Lintermann, A., Eitel-Amor, G., Meinke, M., Schröder, W.: Lattice-Boltzmann Solutions with Local Grid Refinement for Nasal Cavity Flows. In: New Results in Numerical and Experimental Fluid Mechanics VIII, pp. 583–590. Springer (2013)
[6] Lintermann, A., Meinke, M., Schröder, W.: Investigations of the Inspiration and Heating Capability of the Human Nasal Cavity Based on a Lattice-Boltzmann Method. In: Proceedings of the ECCOMAS Thematic International Conference on Simulation and Modeling of Biological Flows (SIMBIO 2011). Brussels, Belgium (2011)
[7] Lintermann, A., Schröder, W.: A Hierarchical Numerical Journey Through the Nasal Cavity: from Nose-Like Models to Real Anatomies. Flow, Turbulence and Combustion 102(1), 89–116 (jan 2019)
[8] Vogt, K., Bachmann-Harildstad, G., Wernecke, K.D., Garyuk, O., Lintermann, A., Nechyporenko, A., Peters, F.: The new agreement of the international RIGA consensus conference on nasal airway function tests. Rhinology 56(2), 133–143 (2018)
[9] Wikipedia contributors: Programming languages used in most popular websites — Wikipedia, the free encyclopedia (2020), https://en.wikipedia.org/w/index.php?title=Programming_languages_used_in_most_popular_websites&oldid=935675978, [Online; accessed 31-January-2020]