bwVisu: A Scalable Remote Service for Interactive Data Processing and Training for Scientists

Erik Schnetter[1], Carlo Antonio Beretta[2], Martin Baumann[1], Sabine Richling[1], Florian Heuschkel[1], Thomas Kuner[2]

[1]University Computing Centre, Heidelberg University;
[2]Department for Anatomy and Cell Biology, Heidelberg University

bwVisu is an easy-to-use, web-based platform for interactive, remote, GUI-based access to scientific work environments that require large compute and storage resources. We explain the technical architecture of bwVisu, the spectrum of available applications, and the typical way of working with the system. With use cases from biological/medical research and teaching we illustrate the advantages and user-friendliness of bwVisu. Finally, we give an outlook on further usage scenarios and possible future development of bwVisu.

1 Introduction

In the era of data sciences, researchers face various challenges that demand large storage capacities, powerful computing hardware, and interactive data exploration capabilities. However, traditional approaches often struggle to meet all these requirements, leading to issues such as lengthy data transfer times, inadequate local hardware capabilities, and limited interactivity in supercomputing environments, which typically rely on command-line skills.

To tackle these challenges, the bwVisu platform (Schridde, Baumann, and Heuveline 2017; Alpay et al. 2020) was developed as a comprehensive solution and gradually improved during the last years. bwVisu is a user-friendly, web-based platform that enables remote access to scientific work environments that require access to large compute und storage resources. It offers a collection of pre-built scientific applications designed to run on powerful hardware such as high-performance computing (HPC) systems which are typically connected to large storage systems. The bwVisu platform has a user interface that allows the selection of applications, the request of compute resources, and the direct web-based execution of graphical interactive applications. Users can interact with the deployed applications in real time through their web browser without the need for special software or hardware.

Publiziert in: Vincent Heuveline, Nina Bisheh und Philipp Kling (Hg.): E-Science-Tage 2023. Empower Your Research – Preserve Your Data. Heidelberg: heiBOOKS, 2023. DOI: https://doi.org/10.11588/heibooks.1288.c18073 (CC BY-SA 4.0)

The bwVisu platform hosted as a service at the University Computing Centre of Heidelberg University benefits from access to bwForCluster Helix[1] and Scientific Data Storage SDS@hd[2]. bwForCluster Helix offers high-performance computing resources and SDS@hd provides a robust storage infrastructure for handling large scientific datasets.

These services are available as "Landesdienste" to researchers at universities from Baden-Württemberg and are part of Baden-Württemberg's concept for High Performance Computing, Data Intensive Computing and Large Scale Scientific Data Management, see Schneider et al. (2019). bwVisu cannot be used as persistent storage services. Instead, bwVisu conceptionally integrates storage services like SDS@hd for this purspose. SDS@hd is one of Baden-Württembergs research storage service with connections and processes for research data management, e.g. transfer data to publication platforms or archives and to foster collaboration among researchers, see Richling et al. (2022). Thus, bwVisu enhances FAIR-based[3] principles in the open sciences by further removing barriers to research data that are due to required specialized IT-skill sets and thus promoting existing platforms for open data exchange.

This paper provides a detailed description of the bwVisu platform including its features and technical architecture (Section 2) as well as the specific implementation of bwVisu as a service at Heidelberg University, the bwVisu workflow, and the availiable applications (Section 3).

In this work we focus on two primary use cases: scientific work and teaching (Section 4). In the scientific work scenario, we highlight how bwVisu addresses challenges related to image analysis, modeling, and simulation. In this case, the bwVisu platform facilitates interactive exploration and efficient utilization of computing resources. In the teaching context, we demonstrate how bwVisu enhances educational environments by providing students with access to powerful scientific tools and fostering collaborative learning experiences.

Furthermore, we engage in a discussion of the presented features, evaluating their benefits and limitations (Section 5). We also outline future developments and planned enhancements, illustrating our ongoing efforts to improve the platform and to meet evolving user requirements (Section 6).

2 bwVisu Architecture

This section explains the technical architecture of bwVisu. It provides an overview of the general structure and describes the individual components in detail. The focus is on the current implementation of bwVisu and the considerations for specific design decisions.

1 https://www.urz.uni-heidelberg.de/en/service-catalogue/high-performance-computing/bwforcluster-helix

2 https://www.urz.uni-heidelberg.de/en/service-catalogue/storage/sdshd-scientific-data-storage

3 https://www.go-fair.org/fair-principles

The technical architecture of bwVisu consists of three layers: frontend, middleware, and backend. The layers communicate via different Application Programming Interfaces (APIs) (Fielding 2000a), namely via RESTfull (Fielding 2000b) HTTP calls or backend specific API calls, see Figure 1. Each layer is discussed in more detail below.

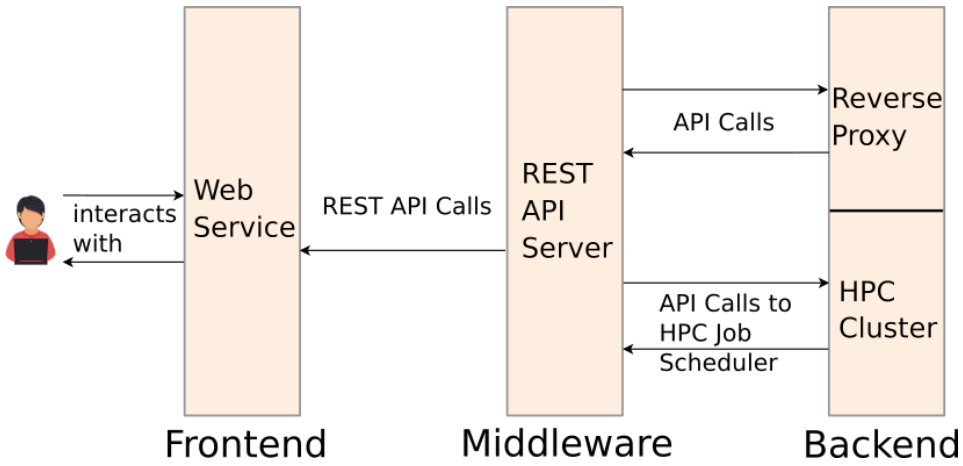

Figure 1: Technical architecture of bwVisu.

2.1 Frontend

The frontend of bwVisu is designed as a classic web application and implemented using the Python web framework Django[4]. It provides users with a web interface, including a login area. Users can start and stop applications offered by bwVisu through this interface. The frontend allows users to request dedicated backend resources for a selected application, such as the number of CPU cores and GPUs. The resource requests are communicated by the middleware to the backend. The frontend also displays links to the streaming endpoints provided by the middleware that allows users to interact with the graphical user interface of the started application in real time through their browser.

Django was chosen because of its suitability for implementing a classic web application in a small group of developers. Django has many built-in features such as user authentication or database management systems that make it possible to significantly reduce development time. Currently, authentication via the LDAP protocol is supported only, but due to the flexible authentication backend[5] of Django, different authentication methods can be easily implemented.

2.2 Middleware

The middleware is implemented as a REST API server using the Python web framework Flask[6]. Flask was chosen for its lightweight nature, which makes it an ideal choice for API

4 `https://www.djangoproject.com`
5 `https://docs.djangoproject.com/en/4.2/topics/auth/customizing`
6 `https://flask.palletsprojects.com/en/2.3.x`

servers that typically require only a few features of a full web development framework. The middleware manages the status of launched applications and provides access to them via JSON-based REST endpoints. Furthermore, it interacts with backend systems via their API interfaces to implement various features of bwVisu. The tasks of the middleware are as follows:

Launching an application: When an application is launched via the associated middleware API endpoint, a generic submit script is populated with application-specific data and submitted as a compute job to the scheduler of the backend HPC system. The submit script loads necessary software modules, starts the remote visualization tool Xpra[7], if the application cannot render its graphical output to the browser itself, and initiates the application within an isolated Singularity[8]/Apptainer[9] container which also provides access to storage systems and other resources needed by the application in said container. After that, the middleware assigns free ports from a pool of reserved ports to bwVisu and creates dynamic port forwarding rules on the reverse web proxy in the backend for the streaming endpoints opened by the application on the HPC system.

Aborting an application: When an application is aborted via the associated middleware API endpoint, the corresponding job on the HPC system is aborted, and the dynamically generated port forwarding rules on the reverse web proxy are removed.

Monitoring running jobs: The middleware regularly retrieves information about running bwVisu jobs on the backend HPC system and stores it in an internal SQLite database.

2.3 Backend

bwVisu uses as backend an HPC system that provides the powerful hardware necessary to run the scientific applications provided by bwVisu as well as a reverse web proxy to manage the port forwarding rules for the streaming endpoints. In this context, a reverse web proxy is a web server that retrieves resources for web clients from other servers on an internal network of the HPC system in order to keep the interface of the internal network to the internet as small as possible and thus protect internal resources from uncontrolled or unauthorized access from the internet.

An HPC system typically provides access to compute and storage resources via a job scheduler or workload manager (e.g. Slurm[10]). The middleware interacts with the job scheduler either by an exposed API or – as a fallback – by a generic shell script which invokes the required commands provided by the job scheduler.

The HPC system must also host the Singularity containers of the bwVisu applications and provide the Singularity and Xpra software.

7 `https://www.xpra.org`
8 `https://sylabs.io/singularity`
9 `https://apptainer.org`
10 `https://slurm.schedmd.com`

To enable dynamic generation of port forwarding rules, bwVisu needs a reverse web proxy that is capable of adapting its configuration on the fly, which is essential for the dynamic nature of the port forwarding rules generated by the middleware. Currently, only Traefik[11] is supported as reverse web proxy, however, support for other proxies can be implemented as well if required. Traefik was selected because its ability to dynamically change the firewall configuration is its key feature, which is critical for managing dynamic port forwarding rules.

3 bwVisu at Heidelberg University

bwVisu is successfully deployed as a service at Heidelberg University with bwForCluster Helix as backend HPC system. bwForCluster Helix is operated by the University Computing Centre as a service for Baden-Württemberg Universities mainly for research in the fields of Structural and Systems Biology, Medical Science, Soft Matter, Computational Humanities as well as method development in scientific computing. The cluster is equipped with about 20,000 AMD EPYC Milan CPU cores, around 200 NVIDIA Ampere Tensor Core GPUs (A100 and A40), and a high-performance parallel file system with a flash tier for high-speed data access.

As a special feature, bwForCluster Helix provides native access to SDS@hd, a storage service for large research data. SDS@hd is directly accessible from a mount point on bwForCluster Helix and is also available for use in bwVisu applications. Thus, the users of bwVisu can directly benefit from massive storage capacities of SDS@hd in the order 20 PB.

3.1 Workflow

To get a better idea of the features of bwVisu, we will now describe the typical workflow in bwVisu from a user's perspective (Figure 2). See also Beretta (2023a) and Beretta (2023b) for recorded user sessions.

Choosing applications and cluster resources: Users choose an application from a list of available options and specify the resources required to run it on the backend HPC system, such as time duration and the number of GPUs. The middleware receives the name of the application along with the resource request via a REST API call.

Starting applications and streaming graphical output: The chosen application is launched on the backend HPC system by the middleware, which launches a submit script corresponding to the chosen application and submits it to the job scheduler. Due to a resource reservation for bwVisu on bwForCluster Helix, a bwVisu job will start immediately, as long as the resources are available. After launching the application on the cluster, the middleware opens the assigned streaming ports via the reverse web proxy. Information regarding the job such as the public web address of the streaming endpoints

11 `https://traefik.io/traefik`

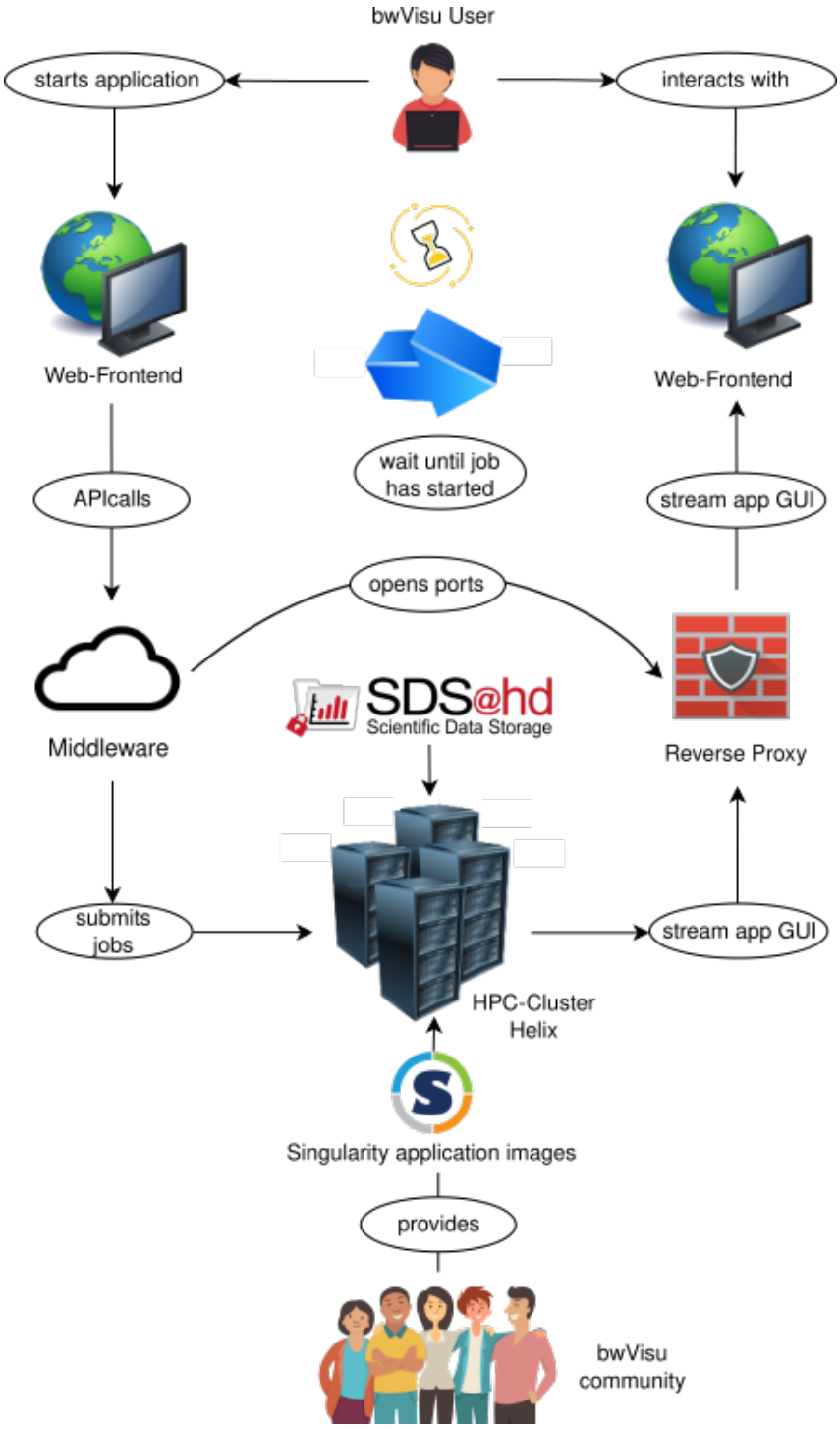

Figure 2: bwVisu workflow at Heidelberg University.

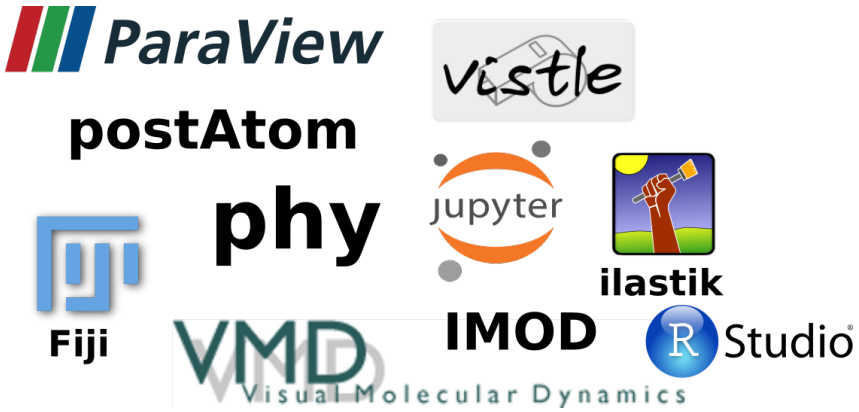

Figure 3: Available bwVisu applications at Heidelberg University.

is made available to the frontend through REST API endpoints. The streaming endpoints for accessing the application's graphical user interface is provided in the web interface as an URL. When this URL is opened by the user, the application's interface is streamed in real-time to the user's web browser, allowing direct interaction with the remote application.

Stopping applications: The application can be stopped either manually by the user or automatically when the specified time limit will be reached. If the user manually cancels the application, the frontend sends an API call to the middleware, which then cancels the associated job on the backend HPC system. In this case, or if the job scheduler reports a timeout for the associated job, the middleware closes the relevant ports via the reverse proxy.

3.2 Applications

bwVisu aims to facilitate the utilization of interactive software in the field of Data Science for scientists. To achieve this, it offers a diverse range of scientific applications commonly used across various disciplines. Figure 3 shows the currently available applications. Expansion of the application repertoire is planned. Deploying additional applications on bwVisu is straightforward due to the minimal requirements imposed by the platform. Essentially, any Linux container-compatible application can be deployed on bwVisu. However, it should be noted that applications designed for multi-user systems may require reconfiguration, as bwVisu exclusively initiates applications for individual users. Based on our experience, such reconfiguration is typically feasible and is usually a matter of just disabling the login mask of the application. Nonetheless, it is possible that certain programs may possess operating requirements that are incompatible with bwVisu.

It is worth reiterating the significant advantage provided by bwVisu in terms of instant availability of scientific applications for scientists who may not possess extensive IT expertise. Additionally, the utilization of powerful hardware without the need for installation, deployment, or complex command-line operations further enhances user productivity.

4 Use Cases

4.1 bwVisu for Science

Microscopy is one of the most common methods to investigate biological processes in life sciences and delivers images that can be challenging to quantify. Yet, image quantification is essential to reveal biological mechanisms, in particular when multimodal imaging approaches are used. In the last decade many open source tools have been developed to facilitate image analysis using conventional approaches (e.g. Imagej/Fiji; see Schindelin et al. 2012) and more recently, machine learning (e.g. ilastik pixel classification; see Berg et al. 2019) or deep learning methods (e.g. noise2void, CARE, and stardit; see Krull, Buchholz, and Jug 2019; Weigert et al. 2018; Schmidt et al. 2018, respectively). Image analysis requires computational resources and, when data analysis demands state-of-the-art tools, programming skills. The rapid and continuous development of artificial intelligence-based deep learning tools for image analysis leads to new powerful ways to precisely quantify images and to enable analyses of complex imaging data that was previously not accessible. In this study, we extensively tested bwVisu functionality using:

- StarDist Jupyter Notebook (Beretta 2023b)
- Ilastik Pixel Classification Workflow (Beretta 2023a)

We used on bwVisu a Jupyter Notebook developed to segment cell nuclei in three dimensional image stacks by training the StarDist deep learning model on GPU resources. bwVisu allows to run the training and the inference directly on cluster resources accessing large dataset of images from SDS@hd storage. The user can install the necessary dependencies to run any Jupyter Notebooks in a Conda environment. bwVisu is a user interactive tool that can be used to train a machine learning classifier on CPU resources with high memory consumption. In our study, we train the ilastik pixel classification workflow (Berg et al. 2019). bwVisu users can run the ilastik workflow on cluster resources and add labels on images in an interactive process. When the training is finished the user can directly batch process large data located on the SDS@hd storage. Both examples show how bwVisu can be used to run applications that require demanding CPU and GPU resources in a user-friendly environment.

4.2 bwVisu for Teaching

Teaching image analysis can be a challenge in particular when GPU resources are needed. bwVisu can be used as centralized infrastructure for running workshops and courses to train scientists on traditional computer vision applications and AI tools overcoming local hardware limitations, for example. There is a large number of methods, tools and applications that could benefit from bwVisu as a teaching platform, since the necessary hardware resources can be reserved and made available on the powerful HPC system for the duration of a course. Additionally, there are processes for the creation of a working area for a course or series of events as well as for the user and group management includ-

ing a right-/role-concept. Course materials can be easily accessible directly from SDS@hd storage creating a common ground to share data during courses and workshops.

In the near future, bwVisu will be used to organize several workshops to train Heidelberg University scientists on state-of-the-art image analysis tools and it will also become a centralized infrastructure to share image analysis tools and knowledge. Furthermore, a course series is in preparation for students using Jupyter on GPUs for different applications via bwVisu. There is a strong interest in using bwVisu for further teaching events at Heidelberg University.

5 Discussion on other technologies

While bwVisu offers a valuable and comprehensive solution for teaching and computational based research, it is important to be aware that alternative platforms exist that serve similar purposes within the scientific community. These alternatives can provide additional features and functionalities that may complement or enhance the teaching experience in consideration of one or another feature.

Subsequently, we will give an overview of some alternative platforms and a comparison of features which might be interesting for operating sites that want to take an informed decision about which platform might meet their specific needs the best.

KASM[12] is a web-based streaming platform focused on providing containerized desktop applications. Notably, KASM introduces a custom open-source VNC server[13] to enhance streaming performance over conventional VNC-based alternatives[14]. Additionally, KASM offers built-in support for educational scenarios, featuring integrated chat functionality and session sharing capabilities tailored for educational contexts. However, it's important to note that Kasm Technologies Inc, the entity behind KASM, hasn't open-sourced all components, leading to licensing costs for adopting KASM. Moreover, KASM lacks integration with existing software in the HPC landscape. By design, there is no integration into an HPC infrastructure for KASM.

Apache Guacamole[15] is an open-source remote desktop gateway aiming to provide a unified access point for diverse server-side streaming technologies through an HTML client and API. A key feature of Apache Guacamole is its browser-based HTML client, enabling user access to stream backend server applications exclusively via web browsers. Additionally, it abstracts various streaming technologies, such as VNC and RDP[16], offering operating sites the flexibility to select their preferred streaming technology. Similar to KASM, Apache Guacamole lacks inherent integration into the HPC landscape.

12 `https://kasmweb.com`
13 `https://kasmweb.com/kasmvnc`
14 `https://en.wikipedia.org/wiki/Virtual_Network_Computing`
15 `https://guacamole.apache.org`
16 `https://en.wikipedia.org/wiki/Remote_Desktop_Protocol`

Open OnDemand[17] is an open-source web application designed to offer user-friendly access to HPC resources for scientific end-users. It provides a web-based interface to access resources and stream applications on HPC clusters, with its standout feature being seamless integration within the HPC landscape. Notably, there exists a wide community of HPC sites that operate Open OnDemand, allowing new adopters to tap into existing knowledge within this community. Furthermore, Open OnDemand's open-source code base has undergone multiple security audits, reducing potential security vulnerabilities.

As previously mentioned, bwVisu aims to offer user-friendly web-based access to HPC resources, allowing for the mapping of interactive scenarios with significant resource demands. When considering the aforementioned platforms in this context, the authors present the following evaluation results from their perspective:

The performance advantages of KASM's open-source VPC server over alternative streaming solutions, such as Xpra used by bwVisu, are not currently clear. Further investigation would be necessary if there is a demand for improvement in bwVisu. KASM's chat and session sharing features might enrich teaching scenarios. However, KASM's licensing model introduces cost considerations and might lead to potential limitations within its field of application.

Apache Guacamole's streaming abstraction results in a high flexibility, which might be useful for some operators. In bwvisu, streaming performance has not been rated as needing improvement thus far. Despite the strengths of KASM and Apache Guacamole, their lack of HPC integration necessitates extra effort and tailored solutions for that use-case.

This is where Open OnDemand emerges. Both bwVisu and Open OnDemand offer distinct advantages for integration with, or utilization within, HPC environments. While we think bwVisu's simple architecture may offer benefits in security and long-term maintainability, Open OnDemand's mature code base, larger feature set, and community support are noteworthy. This platform represents the most promising technology for the authors, which will be further evaluated in the future (see next section).

Overall, determining the "best platform" is likely something that can only be evaluated on a case-by-case basis, using specific and precise use cases.

6 Summary and outlook

The utilization of bwVisu for teaching classes in the field related to data science can be justified based on several key reasons. Firstly, it allows instructors and students to easily bring their own custom applications, enabling tailored and specific educational experiences. Additionally, the availability of ready-made applications eliminates the need for complex installations, reducing technical barriers and enabling efficient use of class time. Moreover, bwVisu leverages powerful hardware resources, such as bwForCluster Helix, which eliminates the burden of managing computing environments and provides students with access to high-performance computing capabilities. Furthermore, the platform's

17 `https://openondemand.org`

integration with large cloud or network storage, such as SDS@hd, facilitates seamless collaboration and data sharing, promoting a collaborative and interactive learning environment. By offering these features, bwVisu offers a compelling rationale for its adoption in data science education, enhancing teaching effectiveness and fostering a conducive learning experience.

While the core features of bwVisu are already implemented, there are specific areas that could benefit from improvements, the introduction of new features, and the exploration of potential alternative approaches. To enhance debugging capabilities and to support the typical workflow of scientists, there are plans to make the outputs of bwVisu job executed on the backend system available to users in the frontend. This functionality would not only aid the operators of the bwVisu service in implementing new applications but also facilitate better debugging capabilities. Additionally, in order to foster the growth of the bwVisu user community and the development of bwVisu applications, users will have the opportunity to locally test and customize new applications for bwVisu using newly provided developer tools. These applications can then be directly uploaded in the frontend for review by the bwVisu operators. Furthermore, access control will be implemented, granting users the opportunity to assume responsibility by becoming maintainers for specific bwVisu applications. In addition to the introduction of new features, it is essential to conduct further benchmarking, tests, and the collection of metrics for typical use-cases. This evaluation aims to assess the robustness of the current implementation when faced with high network latency or a high volume of concurrent user requests and to provide further steps for improvement.

As motivated in the previous section, it is crucial to explore alternative platforms and technologies for remote visualization and remote desktop software in the scientific community. These may offer unique features and capabilities that can complement or enhance the functionality of bwVisu. By examining and testing the integration of these alternative approaches with bwVisu, or exploring how bwVisu can contribute to existing projects, a climate of mutual collaboration can be fostered. When undertaking this process, it is important to consult use cases from both research and teaching domains. This ensures that any subsequent developments are appropriate and contribute to the enhancement of bwVisu's value for research and teaching.

Acknowledgements

The authors acknowledge support by the state of Baden-Württemberg through bwVisu and bwHPC as well as the bwForCluster Helix and the storage service SDS@hd supported by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK) and the German Research Foundation (DFG) through grants and INST 35/1503-1 FUGG and INST 35/1597-1 FUGG. Furthermore, the authors gratefully acknowledge the Federal Ministry of Education and Research (BMBF) and the MWK within the framework of the Excellence Strategy of the Federal and State Governments of Germany and the DFG in form of a CRC1158 that granted bwVisu hardware.

References

Alpay, Aksel, Karsten Hanser, Egzon Miftari, Dennis Schridde, Sabine Richling, Martin Baumann, Filip Sadlo, and Vincent Heuveline. 2020. "bwVisu: A Scalable Remote Visualization Service and its Application to Flow Visualization". In *E-Science-Tage 2019: Data to Knowledge,* edited by Vincent Heuveline, Fabian Gerhart, and Nina Bisheh, 173–184. heiBOOKS. DOI: https://doi.org/10.11588/heibooks.598.c8426.

Beretta, Carlo Antonio. 2023a. "ilastik Pixel Classification Workflow". Visited on September 25, 2023. DOI: https://doi.org/10.11588/heidicon/1747437. https://heidicon.ub.uni-heidelberg.de/#/detail/1747436.

———. 2023b. "StarDist Jupyter Notebook". Visited on September 25, 2023. DOI: https://doi.org/10.11588/heidicon/1747437. https://heidicon.ub.uni-heidelberg.de/#/detail/1747438.

Berg, Stuart, Dominik Kutra, Thorben Kroeger, Christoph N. Straehle, Bernhard X. Kausler, Carsten Haubold, Martin Schiegg, et al. 2019. "ilastik: interactive machine learning for (bio)image analysis". *Nature Methods.* ISSN: 1548-7105. DOI: https://doi.org/10.1038/s41592-019-0582-9.

Fielding, Roy Thomas. 2000a. *Architectural styles and the design of network-based software architectures.* Chapter 6.5.1: Advantages of a Network-based API. University of California, Irvine.

———. 2000b. *Architectural styles and the design of network-based software architectures.* Chapter 5: Representational State Transfer (REST). University of California, Irvine.

Krull, Alexander, Tim-Oliver Buchholz, and Florian Jug. 2019. "Noise2void-learning denoising from single noisy images". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2129–2137. DOI: https://doi.org/10.48550/arXiv.1811.10980.

Richling, Sabine, Sven Siebler, Alexander Balz, and Martin Kühl Robert und Baumann. 2022. "Managing large research data with SDS@hd". In *E-Science-Tage 2021: Share Your Research Data,* edited by Vincent Heuveline and Nina Bisheh, 421–427. heiBOOKS. DOI: https://doi.org/10.11588/heibooks.979.c13759.

Schindelin, Johannes, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. 2012. "Fiji: an open-source platform for biological-image analysis". *Nature methods* 9 (7): 676–682. DOI: https://doi.org/10.1038/nmeth.2019.

Schmidt, Uwe, Martin Weigert, Coleman Broaddus, and Gene Myers. 2018. "Cell Detection with Star-Convex Polygons". In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II,* 265–273. DOI: https://doi.org/10.1007/978-3-030-00934-2_30.

Schneider, Gerhard, Vincent Heuveline, Karl-Wilhelm Horstmann, Bernhard Neumair, Petra Hätscher, Josef Kolbitsch, Simone Rehm, Michael Resch, Thomas Walter, Stefan Wesner, et al. 2019. "Umsetzungskonzept der Universitäten des Landes Baden-Württemberg für das High Performance Computing (HPC), Data Intensive Computing (DIC) und Large Scale Scientific Data Management (LS^2 DM)". In *Proceedings of the 5th bwHPC Symposium.* Universität Tübingen. DOI: https://doi.org/10.15496/publikation-29040.

Schridde, Dennis, Martin Baumann, and Vincent Heuveline. 2017. "Skalierbare und flexible Arbeitsumgebungen für Data-Driven Sciences". In *E-Science-Tage 2017: Forschungsdaten managen,* edited by Jonas Kratzke and Vincent Heuveline, 153–166. heiBOOKS. DOI: https://doi.org/10.11588/heibooks.285.c3887.

Weigert, Martin, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Jain Akanksha, Benjamin Wilhelm, et al. 2018. "Content-aware image restoration: pushing the limits of fluorescence microscopy". *Nature Methods:* 1090–1097. DOI: https://doi.org/10.1038/s41592-018-0216-7.