High Performance Visualization with VisIVO Across Cloud and HPC

Umer Arshad
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Eva Sciacca
INAF Astrophysical Observatory of Catania, Italy

Nicola Tuccari
Department of Mathematics and Informatics, University of Catania, Italy

Fabio Pitari
CINECA, Bologna, Italy

Giuseppa Muscianisi
CINECA, Bologna, Italy

Abstract

The rapid growth of data in Astrophysics and Cosmology creates significant challenges that require scalable computing and advanced visualization solutions. Cineca is Italy's largest supercomputing center and a leading global provider of high-performance computing (HPC) services. This paper shows that the integration of the VisIVO scientific visualization framework is with the cloud-based InterActive Computing (IAC) service at Cineca. This integration enables GPU-accelerated, real-time visualization on HPC resources via a Jupyter interface in their browser. A new dedicated Python wrapper and a custom Jupyter kernel enable VisIVO to run smoothly from interactive notebooks, avoid command-line operations, and visualize data directly on HPC compute nodes. Furthermore, we enabled cloud-oriented RESTful APIs, built with the Flask framework, to perform VisIVO operations remotely via simple web services. This setup hides the backend's complexity and simplifies connections with other applications. Our framework increases system accessibility, ensures reproducibility of results, and supports rapid data exploration for large astrophysical simulations. The system was evaluated using real-world cases, including visual analysis of cosmological simulations generated using the OpenGadget3 code. Results indicate that the system is scalable and reliable, and that it facilitates interactive scientific discovery on high-performance computing (HPC) infrastructures.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases High-performance computing, HPC, VisIVO, Scientific visualization, Interactive visualization, Cloud computing, Jupyter, Flask, REST API

Digital Object Identifier 10.4230/OASIcs.PARMA-DITAM.2026.3

Funding The work is supported by the EuroHPC JU under grant agreement No 101093441 (SPACE CoE) and by the Spoke 1 "Future HPC & BigData" of the ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by NextGenerationEU.
Umer Arshad: ICSC – Spoke 1.
Eva Sciacca: SPACE CoE, ICSC – Spoke 1.
Nicola Tuccari: ICSC – Spoke 1.
Fabio Pitari: ICSC – Spoke 1.
Giuseppa Muscianisi: ICSC – Spoke 1.

17th Workshop on Parallel Programming and Run-Time Management Techniques for Many-Core Architectures and 15th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM 2026).
Editors: Davide Baroffio, Paola Busia, Lev Denisov, and Nitin Shukla; Article No. 3; pp. 3:1–3:11

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Astrophysical observations and simulation codes on high-performance supercomputers generate petabytes of data [22]. Managing this data – storing, retrieving, and analyzing it – presents major challenges critical to scientific progress [17]. Pre-exascale systems enable new ways to scale High-Performance Computing (HPC) for Astrophysics and Cosmology. Researchers now need both powerful computation and interactive methods to visualize results [7]. VisIVO[1], the Visualization Interface for the Virtual Observatory is a suite of tools for high-performance, multidimensional data analysis and visualization in astrophysics [11]. It was first built for distributed systems like grids and clouds [5], and later updated for the European Open Science Cloud (EOSC) [20, 21] using containerized science gateways.

VisIVO supports interactive and portable workflows [19]. Most recently, it has been integrated with the cloud-oriented InterActive Computing (IAC[2]) service [4], which is available on the Galileo 100 cluster (G100[3]) [9], at Cineca. This IAC service enables users to access compute nodes on demand via Jupyter notebooks in a web browser. The IAC service fundamentally transforms the conventional batch-based HPC access model into a fully interactive, web-accessible environment. This platform enables users to initiate notebook sessions directly on compute nodes equipped with GPUs and high-speed interconnects, thereby facilitating interactive data analysis, real-time visualization, and dynamic simulation control [3]. By enhancing usability and reducing technical barriers, IAC addresses critical challenges in HPC accessibility and user engagement. The integration of VisIVO with IAC further empowers researchers to conduct 3D visualization workflows seamlessly within the HPC ecosystem, eliminating the need for complex command-line configurations or manual data transfer. This integration extends the scalability and computational power of HPC to Jupyter-based environments, promoting reproducible, accessible, and user-centric scientific workflows.

We embed VisIVO within IAC via lightweight Python wrappers and a custom Jupyter kernel that automatically loads modules, configures environment variables, and transparently calls the underlying C++ binaries [22]. This design hides system complexity and delivers consistent execution across distributed nodes. Additionally, we have developed cloud-oriented RESTful API services that expand VisIVO's capabilities beyond the notebook interface. Built with lightweight Python frameworks, these APIs enable remote execution of visualization tasks and provide programmatic and web-based access for large-scale data rendering and analysis. This work complements the interactive IAC integration by enhancing automation and laying the foundation for future cloud–HPC hybrid environments.

The paper is organized as follows: Section 2 provides an overview of related works in interactive and HPC-driven visualization; Section 3 introduces the VisIVO Server modules and its architecture; Section 4 introduces the InterActive Computing (IAC) service; Section 5 describes the methodology used to integrate VisIVO into the IAC framework, while Section 6 focuses on the development of REST-based visualization APIs; Conclusions and future developments are discussed in Section 7.

[1] VisIVO, `https://visivo.readthedocs.io/`
[2] IAC, `https://jupyter.g100.cineca.it/`
[3] Galileo 100 infrastructure: `https://www.hpc.cineca.it/systems/hardware/galileo100/`

2 Related Works

Scientific visualization has progressed markedly at scale – especially in astrophysics and cosmology – where petascale simulations and surveys demand real-time analysis and visual insight. Although existing tools and platforms provide a solid foundation, just a few let you do data analysis and visualization in real-time, in one reproducible HPC workflow. Early pioneers in large-scale HPC visualization, VisIt [8] and ParaView [1] established powerful frameworks for parallel and remote rendering across distributed systems. Although both support HPC-friendly client–server models, they typically demand expert configuration and do not offer native web integration, which constrains usability for many domain scientists.

Interactive web-based platforms have become popular through science gateways and cloud-centric systems. For instance, [6] gateways have been shown to provide researchers access to sophisticated astrophysical analysis tools like VisIVO through web portals. Following this, the NEANIAS Visualization Gateway [21] was developed, incorporating VisIVO into the European Open Science Cloud (EOSC) using containerized services and Jupyter interfaces. This work demonstrated that deploying advanced astrophysical visualization on federated cloud resources is feasible, ensuring reproducibility and cross-platform portability. Nevertheless, these gateways generally do not deeply integrate with HPC schedulers or support real-time session control, operating primarily in batch mode or with limited interactivity.

As Jupyter Notebooks have become standard tools for reproducible scientific workflows, various projects have investigated combining them with high-performance computing resources. The work of [16] introduced Jupyter as a user interface for scientific computing, and newer efforts, such as the Fenix infrastructure [4], have adapted this approach specifically for neuroscience and brain studies. These platforms demonstrated that HPC systems could be accessed via notebook-driven web portals, though they primarily focused on static data analysis rather than interactive, GPU-powered visualization.

Here, combining VisIVO with Cineca's InterActive Computing (IAC) service, as demonstrated in this study, addresses an important shortfall. This method enables interactive 3D rendering on HPC nodes and packages VisIVO's functionality within a specialized Jupyter kernel via Python wrappers, making it easier for users to access high-performance GPU resources. Unlike [21], this solution removes the necessity for manual module management, SSH file transfers, or command-line interface rendering. Additionally, it offers live visualization through VisIVO Viewer and supports complex processing workflows by using pre-configured, scriptable notebooks. Spack [10] and Ansible [13] are commonly used tools for automating deployments in HPC software environments, and their application here to merge Python and C++ tools into unified notebook setups highlights an advanced degree of composability. This approach converts traditional static command-line workflows into interactive, reproducible sessions, thereby enhancing both the reproducibility and accessibility of scientific visualization methods in astrophysics. This method has been applied specifically to astrophysical simulations such as GADGET and to workflows involving density visualization and the examination of cosmological structures [22], but it might be generalized to any command-line-based backend tool. Additionally, it enables the use of cloud-based RESTful services and serverless visualization backends, simplifying HPC complexity without sacrificing efficiency.

3 VisIVO Server

VisIVO Server is a collection of software tools built to produce customized 3D visualizations using astrophysical datasets, handling very large amounts of data with no restrictions on data dimensions [23]. The primary modules available via the command line are:

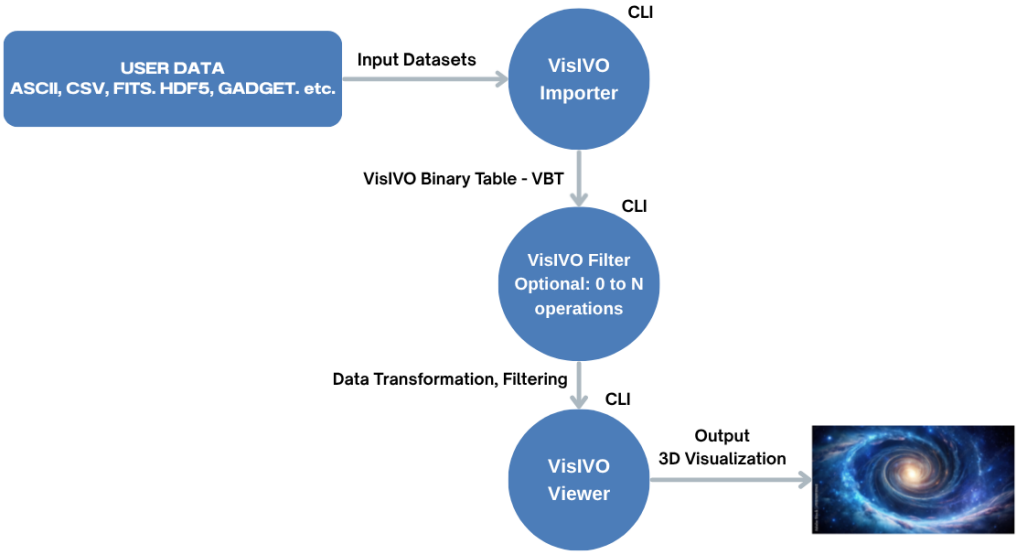

Figure 1 VisIVO Server modular architecture and basic workflow involving importer, filter and view operations.

- VisIVO Importer: transforms user datasets into an efficient internal format called VisIVO Binary Tables (VBT), supporting a wide range of formats from common data types like ASCII and CSV to specialized astronomy formats such as FITS, HDF5, and GADGET.
- VisIVO Filter: allows users to process VBT files by applying data transformations, performing filtering, and other modifications needed before visualization.
- VisIVO Viewer: creates interactive 3D graphics from VBTs, enabling users to visually analyze and explore datasets efficiently.

A standard VisIVO Server workflow progresses through three stages – data preparation, processing, and visualization (as shown in Figure 1).

4 InterActive Computing service (IAC)

As the demands of modern scientific workflows increase, there is a need for capabilities beyond traditional batch-based processing. Fenix[4] addresses this need for interactive access to large-scale computing resources by providing a federated set of infrastructure services that complement traditional batch-oriented HPC systems [2]. Fenix introduces Interactive Computing Services (IAC) as a core constituent of its architecture. IAC enables users to access a persistent, low-latency interactive environment, typically via web-based interfaces such as Jupyter. IAC has been developed with a focus on supporting neuroscience work in the Human Brain Project (HBP[5]) and European Brain Research INfrastructure (EBRAINS[6]) communities. This service lets researchers interact with data and run applications directly, facilitating rapid feedback and accelerating scientific progress. The development

[4] Fenix, `https://fenix-ri.eu/`
[5] Human Brain Project, `https://humanbrainproject.eu/`
[6] EBRAINS, `https://ebrains.eu/`

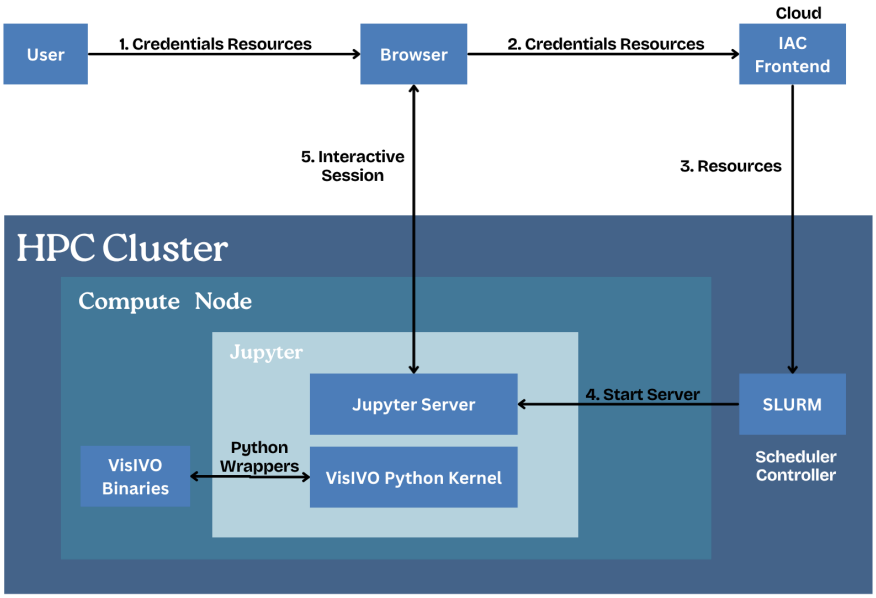

Figure 2 InterActive Computing VisIVO service implementation at Cineca.

and management of IAC services were a collaborative effort between Cineca and E4, using the ICE4HPC[7] software suite. Currently, IAC is available on Cineca's Galileo 100[8] supercomputing platform [22].

An interactive, browser-based interface to HPC delivers two key benefits.

1. Users can launch sessions directly on compute nodes at `https://jupyter.g100.cineca.it` with near-instant startup, bypassing the traditional SSH, batch-queue workflow and gaining seamless access to graphical visualization tools that are cumbersome in command-line-only environments.
2. It enables real-time control of running workflows – monitoring resource usage [15], inspecting intermediate results, and adjusting parameters on the fly – avoiding the unpredictable start times and slow feedback cycles inherent to batch processing.

The Interactive Access and Computing (IAC) framework delivers a secure, end-to-end interactive HPC experience by routing user authentication and resource requests from a local browser to a cloud-hosted frontend that isolates the web surface, which then forwards requests to Slurm to allocate a compute node and launch a Jupyter server with a VisIVO-enabled Python kernel, as shown in the UML diagram of Figure 2. Users work continuously in notebooks on that node – running heavy computations directly on HPC resources while monitoring usage and visualizing intermediate results in real time. Jupyter is chosen for its single-user server–client model, rich plugin ecosystem (e.g., monitoring tools, featured dashboards, and interface with other services via jupyter-server-proxy), and broad user familiarity. Security is enforced by executing orchestration on a dedicated VM in Cineca's cloud, running HPC operations as standard user-space Slurm jobs (no root privileges required on the cluster side), and applying two-factor authentication alongside continuous monitoring, vulnerability assessments, and penetration testing.

[7] ICE4HPC, `https://www.e4company.com/en/ice4hpc/`
[8] Galileo 100 infrastructure: `https://www.hpc.cineca.it/systems/hardware/galileo100/`

5 Methodology

Integrating a dedicated VisIVO Jupyter kernel into the IAC replaces the old ssh-and-module-load workflow on G100, eliminating command-line setup and extra steps to fetch PNG outputs. Users now launch an IAC session, select "VisIVO," and run commands immediately; outputs are saved to the session and viewable in the web interface. This streamlines visualization, improves accessibility, and reduces friction – even though VisIVO, written in C++, isn't a native Jupyter target – details of which are described next.

5.1 VisIVO wrappers

VisIVO is implemented in C/C++, and its compilation generates standalone executable binaries. Since Jupyter cannot directly embed such binaries in its interface; integration is achieved through a dedicated Python environment. To enable this, we developed a custom Python wrapper[9] that bridges Python with the VisIVO binaries, allowing users to execute VisIVO commands seamlessly within a Jupyter session. The Python wrapper is intended to execute a specific command while also handling essential tasks, such as invoking the CLI, logging activities, and verifying the function arguments.

The wrapper provides command-specific functions – VisIVOImporter, VisIVOFilter, and VisIVOViewer – that share a common design pattern (here below we will show the importer as an example), enabling seamless execution of VisIVO commands from a Python session.

```
def importer(input_file, *flags, mpi=False, tasks=None, fformat=None, out=None,
    volume=None, compx=None, [...]):
```

The design deliberately hides the VisIVO command-line from users delegating the VisIVO command line execution to an ad-hoc function (`_corefunction`):

```
command = "VisIVOImporter"
_corefunction(command, input_file, *flags, mpi=mpi, tasks=tasks, options=options)
```

The `_corefunction` validates the types of options passed from the outer wrapper, converts booleans into flag-only options and all other parameters into key–value options, then appends the input file to finalize the command; execution runs either with MPI or in serial mode based on the `mpi` and `tasks` settings. Additionally, integrating the *Pillow* library with the VisIVO kernel improves image handling – especially PNGs – by rendering outputs directly inside the Jupyter session, eliminating command-line viewers and manual file retrieval. This makes image generation and inspection faster and simpler, so users can stay focused on analysis rather than setup.

5.2 IAC Deployment

To deploy the framework, VisIVO was first compiled system-wide, with a Spack [10]. A Spack module enforces the OSMesa backend across graphical libraries such as VTK and GLEW. This enforcement ensures correct image output in IAC through off-screen rendering with OSMesa, which is crucial because the system lacks an active windowing system and only the web front end is operational. Then, a *conda* environment was provisioned with *Ansible* [13] on the IAC backend nodes to host the VisIVO Python wrappers (described in the previous Section 5.1), optionally adding Pillow, NumPy, and Matplotlib for smoother

[9] VisIVO Python Wrapper: `https://github.com/VisIVOLab/VisIVOPythonWrapper`

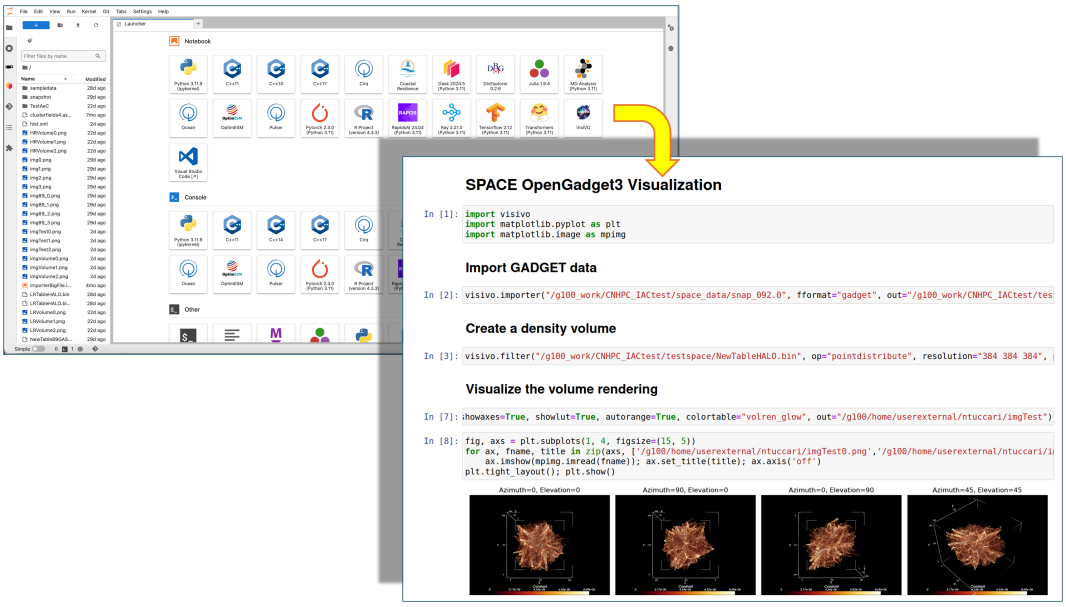

Figure 3 IAC VisIVO framework deployment and sample notebook to visualize OpenGadget3 data.

visualization. Ansible deploys the conda environment to each backend node's local storage (not the parallel file system) to avoid login slowdowns from many small file reads during Jupyter initialization; the inventory explicitly lists all service nodes, and the login node acts as the Ansible control host.

After that, the VisIVO Python wrappers were installed. With the wrappers in place, users can invoke VisIVOImporter, VisIVOFilter, and VisIVOViewer directly from Python, simplifying analysis and visualization workflows. Finally, the Jupyter ipykernel was customized to execute a bash prolog that loads the Spack VisIVO module and exports the necessary environment variables, making the VisIVO kernel immediately available in notebook and console sessions.

This deployment is being tested within the SPACE[10] project to interactively visualize the cosmological snapshots of two simulation codes, namely OpenGadget3 [12] and ChaNGa [14], in GADGET and Tipsy format respectively, as shown in Figure 3.

6 Automated pipeline using VisIVO

A second target for our codebase is pipeline-style automation, which is essential for high-throughput science (e.g., weather forecasting, tsunami alerts, satellite data management), where multiple servers interact in cascades that eventually trigger HPC jobs. Because initiating these interactions depends on site-specific HPC access policies, no standard method exists; in our earlier IAC use case, we overcame this by using Cloud–HPC coupling, which allows us to expose web services typically disallowed on HPC clusters. Extending that idea, we propose embedding standard web REST APIs into HPC workflows to provide secure, scalable, and reproducible automation from submission to result retrieval. REST makes

[10] SPACE project: `https://www.space-coe.eu/`

services client-agnostic, so requests can be sent from the command line (e.g., curl) or any web front end without server-side changes. Similar needs arise in projects like Terabit [24, 25] (rapid earthquake simulations) and GAIA [17] (post-processing large astronomical datasets), where pipelines coordinate data flows from remote sources (telescopes, seismic sensors) through interconnected steps but still lack a general approach – REST APIs are a strong candidate standard. To prototype this, we used our VisIVO codebases to simulate a pipeline that automatically post-processes simulation data using the cloud-hosted VisIVO REST API.

6.1 Deployment as a service

To achieve integration with pipelines via the cloud-based REST API, we integrated Flask [18] into our original code, i.e., the VisIVO Python wrappers described in Section 5.1. The choice of Flask is a perfect match for the implementation. Flask enables the rapid deployment of a web server using minimal Python code.

```
app = Flask(__name__)
if __name__ == "__main__":
app.run(host='0.0.0.0', port=5000, debug=True)
```

It processes requests and generates responses based on the application's configuration. Developers can specify the server's network address and port. In development mode, the application reloads automatically upon code modification, ensuring immediate updates, and provides detailed error messages through an interactive debugger.

```
@app.route('/some_endpoint', methods=['POST', 'GET'])
def some_endpoint():
```

Flask provides function decorators that make it easy to implement REST API endpoints. This feature aligns well with our starting code, which is organized around three core functions. Using Flask decorators, we can clearly map each function to a specific endpoint, simplifying the creation of REST APIs for the cloud-hosted server.

The REST API runs on a virtual machine in the OpenStack-based Cineca ADA cloud[11]; this is because bare-metal execution on the HPC cluster of web services is typically discouraged due to an higher probability of security issues that might affect such services, with the risk of compromising the security of the cluster. We set up VisIVO inside a Docker container dedicated to this service; while not strictly necessary, this cloud-friendly approach offers greater convenience and portability compared to a traditional installation. Alongside the container, we created a dedicated Python virtual environment to host the server, ensuring that REST API dependencies are cleanly managed. Together, these steps deliver an isolated, efficient backend layer specifically designed to support the reliable, scalable use of the application's REST API.

The primary goal is to provide a language-agnostic interface that abstracts system complexity via standard REST APIs. Currently, the "Import" action operates only on the cloud web server which serves as the system's control layer for handling user input, request validation, and secure REST interactions. To fully realize our aim, we plan to offload these operations to the HPC cluster. In the future, with shared storage, computation will be colocated with data – improving performance, scalability, and reliability on high-performance infrastructure.

[11] ADA cloud, `https://www.hpc.cineca.it/systems/hardware/ada-cloud/`

7 Conclusions and Future Works

This study describes how the VisIVO scientific visualization framework was integrated into Cineca's InterActive Computing (IAC) service to create an interactive, scalable, and reproducible visualization environment directly on HPC systems. The integration transformed VisIVO from a command-line, batch-based tool into a web-accessible, browser-driven platform that supports complex 3D visualizations within Jupyter notebooks using GPU nodes. Python wrappers and a dedicated Jupyter kernel make it easier for users to interact and run VisIVO commands manual configuration or module loading. These updates speed up visualization, provide clear HPC access, and help scientists analyze data more quickly. Cloud-based RESTful APIs built with Flask now let users run VisIVO remotely through web services, which simplifies the backend and makes it easier to connect with other applications. Overall, this paper showcases how conventional visualization software can be adapted into modern HPC-native solutions that combine high performance with user-friendly accessibility. The framework was tested on real astrophysics workloads, such as the ones from the SPACE project, and handled them smoothly, efficiently scaling and reliably visualizing very large, complex datasets. These trials demonstrated its ability to connect traditional batch supercomputing with real-time, interactive analysis.

In the future, the next step is to exploit the cloud-hosted REST API with VAST's multi-protocol shared storage so the web front end can read cluster-generated data directly – no data movement – possibly via VAST's object-storage interface for extra flexibility. The REST API will require proper authentication, likely token-based. Once shared storage is active, a REST workflow similar to IAC can submit VisIVO jobs to the cluster's batch system, moving computation to the data. This should boost throughput and I/O efficiency while keeping security tight by not exposing any web server from the cluster.

Declaration on Generative AI

During the preparation of this work, the authors made use of OpenAI's ChatGPT and Google's Gemini for paraphrasing, rewording, and checking grammar and spelling. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

1 James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. In *Visualization Handbook*, pages 717–731. Elsevier, 2005. doi:10.1016/B978-012387582-2/50038-1.

2 Sadaf Alam, Javier Bartolome, Sanzio Bassini, Michele Carpene, Mirko Cestari, Frederic Combeau, Sergi Girona, Stefano Gorini, Giuseppe Fiameni, Björn Hagemeier, Andreas Herten, Nikoleta Kiapidou, Wouter Klijn, Dorian Krause, Jacques-Charles Lafoucriere, Cerlane Leong, Thomas Leibovici, Thomas Lippert, Colin McMurtrie, Pavel Mezentsev, Anne Nahm, Boris Orth, Dirk Pleiter, Thomas Schulthess, Benedikt von St. Vieth, Debora Testi, and Gilles Wiber. Fenix: Distributed e-infrastructure services for ebrains. In Katrin Amunts, Lucio Grandinetti, Thomas Lippert, and Nicolai Petkov, editors, *Brain-Inspired Computing*, pages 81–89, Cham, 2021. Springer International Publishing.

3 Sadaf R Alam, Javier Bartolome, Sanzio Bassini, Michele Carpenè, Mirko Cestari, Frederic Combeau, Sergi Girona, Stefano Gorini, Giuseppe Fiameni, Björn Hagemeier, et al. Archival data repository services to enable hpc and cloud workflows in a federated research e-infrastructure. In *2020 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud)*, pages 39–44. IEEE, 2020.

4 Shafqat Alam, Juan Bartolome, Stefano Bassini, Massimo Carpene, Marco Cestari, Franck Combeau, Santi Girona, Stefano Gorini, Giuseppe Fiameni, Benedikt Hagemeier, et al. Fenix: Distributed e-infrastructure services for ebrains. In Katrin Amunts, Lucio Grandinetti, Thomas Lippert, and Nikolay Petkov, editors, *Brain-Inspired Computing*, pages 81–89. Springer International Publishing, 2021. doi:10.1007/978-3-030-70710-8_7.

5 Ugo Becciani, Eva Sciacca, Alessandro Costa, Piero Massimino, Costantino Pistagna, Simone Riggi, Fabio Vitello, Catia Petta, Marilena Bandieramonte, and Mel Krokos. Science gateway technologies for the astrophysics community. *Concurrency and Computation: Practice and Experience*, 27(2):306–327, 2015. doi:10.1002/CPE.3255.

6 Ugo Becciani, Eva Sciacca, Antonio Costa, Paola Massimino, Caterina Pistagna, Salvatore Riggi, Fabio Vitello, Claudio Petta, Massimo Bandieramonte, and Michalis Krokos. Science gateway technologies for the astrophysics community. *Concurrency and Computation: Practice and Experience*, 27(2):306–327, 2015. doi:10.1002/cpe.3280.

7 Anthony GA Brown, Antonella Vallenari, TJDBJH Prusti, Jos HJ De Bruijne, Carine Babusiaux, Coryn AL Bailer-Jones, Michael Biermann, Dafydd Wyn Evans, Laurent Eyer, Femke Jansen, et al. Gaia data release 2-summary of the contents and survey properties. *Astronomy & astrophysics*, 616:A1, 2018.

8 Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. Visit: An end-user tool for visualizing and analyzing very large data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Chapman and Hall/CRC, October 2012. doi:10.1201/b12985.

9 Antonio Corradi, Giuseppe Di Modica, Luca Evangelisti, Anna Fiorini, Luca Foschini, and Luca Zerbini. Hs-autofit: A highly scalable autofit application for cloud and hpc environments. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020. doi:10.1109/ISCC50000.2020.9219556.

10 Todd Gamblin, Matthew LeGendre, Michael R Collette, Gregory L Lee, Adam Moody, Bronis R De Supinski, and Scott Futral. The spack package manager: bringing order to hpc software chaos. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–12, 2015. doi:10.1145/2807591.2807623.

11 C Gheller, M Comparato, and U Becciani. Visivo: interoperable visualization and data analysis for the vo. In *Astronomical Data Analysis Software and Systems XV*, volume 351, page 29, 2006.

12 Frederick Groth, Ulrich P Steinwandel, Milena Valentini, and Klaus Dolag. The cosmological simulation code opengadget3–implementation of meshless finite mass. *Monthly Notices of the Royal Astronomical Society*, 526(1):616–644, 2023.

13 Lorin Hochstein and Rene Moser. *Ansible: Up and Running: Automating configuration management and deployment the easy way.* " O'Reilly Media, Inc.", 2017.

14 Pritish Jetley, Filippo Gioachin, Celso Mendes, Laxmikant V Kale, and Thomas Quinn. Massively parallel cosmological simulations with changa. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–12. IEEE, 2008.

15 Morris A Jette and Tim Wickberg. Architecture of the slurm workload manager. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 3–23. Springer, 2023. doi:10.1007/978-3-031-43943-8_1.

16 Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks, 2016.

17 Timo Prusti, JHJ De Bruijne, Anthony GA Brown, Antonella Vallenari, C Babusiaux, CAL Bailer-Jones, U Bastian, M Biermann, Dafydd Wyn Evans, L Eyer, et al. The gaia mission. *Astronomy & astrophysics*, 595:A1, 2016.

18 Kunal Relan. Building rest apis with flask. *Building REST APIs with Flask*, 2019.

19 Eva Sciacca, Valentina Cesare, Nicola Tuccari, Fabio Vitello, Iacopo Colonnelli, Camelita Carbone, Emiliano Tramontana, and Ugo Becciani. Toward a portable and reproducible high-performance visualization for astronomy & cosmology, September 2024. doi:10.5281/zenodo.13858498.

20 Eva Sciacca, Mel Krokos, Cristobal Bordiu, Carlos Brandt, Fabio Vitello, Filomena Bufano, Ugo Becciani, Mario Raciti, Giuseppe Tudisco, Simone Riggi, et al. Scientific visualization on the cloud: the neanias services towards eosc integration. *Journal of Grid Computing*, 20(1):7, 2022. doi:10.1007/S10723-022-09598-Y.

21 Eva Sciacca, Mario Raciti, Mel Krokos, Benjamin Kyd, et al. Science gateways in eosc: The neanias visualisation gateway. In *Proceedings of the 14th International Workshop on Science Gateways*, 2023.

22 Eva Sciacca, Nicola Tuccari, Umer Arshad, Fabio Pitari, Giuseppa Muscianisi, and Emiliano Tramontana. Interactive high-performance visualization for astronomy and cosmology. *arXiv preprint arXiv:2510.04665*, 2025. doi:10.48550/arXiv.2510.04665.

23 Nicola Tuccari, Eva Sciacca, Fabio Vitello, Iacopo Colonnelli, Yolanda Becerra, Enric Sosa Cintero, Guillermo Marin, Milan Jaros, Lubomir Riha, Petr Strakos, et al. High performance visualization for astrophysics and cosmology. In *2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 443–450. IEEE, 2025.

24 Elisa Zuccolo, Giorgio Bolzon, Fabio Pitari, Ileana Elizabeth Monsalvo Franco, Chiara Scaini, Valerio Poggi, Chiara Smerzini, Stefano Salon, et al. Urgentshake: An hpc system for near real-time physics-based ground shaking simulations to support emergency response efforts. In *EGU2025*. Copernicus Meetings, 2025.

25 Elisa Zuccolo, Giorgio Bolzon, Fabio Pitari, Lucía Rodríguez Muñoz, Chiara Scaini, Manuela Vanini, Valerio Poggi, and Stefano Salon. Advancing rapid response to earthquakes with tiered physics-based ground-shaking simulations: The urgentshake system. *Seismological Research Letters*, 2025.