SN Computer Science (2026) 7:527
https://doi.org/10.1007/s42979-026-05076-4

ORIGINAL RESEARCH

Hybrid Streaming Workflows for Scalable Visualization in Astronomy and Cosmology

Eva Sciacca[1] · Nicola Tuccari[1,2] · Marco Edoardo Santimaria[3] · Valentina Cesare[4] · Fabio Vitello[1] · Alberto Mulone[3] · Doriana Medić[3] · Iacopo Colonnelli[3]

Received: 20 September 2025 / Accepted: 7 May 2026

Abstract

Modern astrophysics and cosmology generate petabyte-scale datasets that demand advanced tools for scalable visualization and analysis. The Visualization Interface for the Virtual Observatory (VisIVO) provides multi-dimensional data exploration, but its integration with heterogeneous computing infrastructures and data-intensive workflows has remained challenging. Here we present a portable and reproducible approach that combines VisIVO with the StreamFlow workflow engine and the CAPIO middleware. StreamFlow enables hybrid execution across cloud-HPC environments, while CAPIO injects transparent I/O streaming into legacy file-based pipelines without modifying application code. Together, these technologies accelerate high–performance visualization, improve reproducibility, and reduce I/O bottlenecks. We demonstrate our approach on large-scale cosmological simulations produced with ChaNGa and GADGET-based codes, including studies of massive neutrinos in the DEMNUni suite. Our results show that streaming-enhanced workflows deliver up to 50% performance gains and enable interactive, scalable visualization across distributed infrastructures, paving the way for exascale-ready scientific discovery.

Keywords Hybrid workflows · Reproducibility · Exascale computing · I/O streaming · Scientific visualization · Astrophysics and cosmology · Data-intensive computing

Introduction

The exponential increase in data volume from astronomical observations and numerical simulations has confirmed the data-intensive nature of astrophysics and cosmology (A&C). Large-scale N-body and hydrodynamical simulations now routinely produce datasets on the order of petabytes, offering unprecedented opportunities to investigate phenomena such as dark matter distribution, galaxy formation, and the role of neutrinos in cosmic evolution. However, extracting knowledge from such data requires advanced visualization methods that go beyond traditional approaches, enabling researchers to interact with complex, high-dimensional, and time-evolving datasets.

✉ Eva Sciacca
eva.sciacca@inaf.it

Nicola Tuccari
nicola.tuccari@inaf.it

Marco Edoardo Santimaria
marcoedoardo.santimaria@unito.it

Valentina Cesare
valentina.cesare@inaf.it

Fabio Vitello
fabio.vitello@inaf.it

Alberto Mulone
alberto.mulone@unito.it

Doriana Medić
doriana.medic@unito.it

Iacopo Colonnelli
iacopo.colonnelli@unito.it

[1] Astrophysical Observatory of Catania, INAF, Via Santa Sofia 78, Catania 95123, Italy

[2] Department of Mathematics and Informatics, University of Catania, Viale Andrea Doria 6, Catania 95125, Italy

[3] Computer Science Department, University of Turin, Corso Svizzera 185, Torino 10149, Italy

[4] Radioastronomical Institute, Radioastronomical Station of Medicina, INAF, Via Fiorentina 3513, Medicina 40059, Italy

Published online: 04 June 2026

In Europe, initiatives such as the SPACE Centre of Excellence are preparing widely used astrophysical simulation codes for efficient use of pre-exascale and future exascale systems,[1] while the Italian National Research Centre for High-Performance Computing, Big Data and Quantum Computing (ICSC) is driving innovation in workflows, I/O, and cloud-HPC convergence[2] [1]. Within this ecosystem, high-performance visualization plays a central role by providing scalable and interactive methods to bridge the gap between data generation and scientific discovery.

The Visualization Interface for the Virtual Observatory (VisIVO) is a toolkit designed for multi-dimensional data analysis and knowledge discovery across heterogeneous astrophysical datasets. VisIVO has already been deployed on distributed computing infrastructures through Science Gateways [2], containerization, and integration with the European Open Science Cloud (EOSC) [3]. Building on this foundation, the present work introduces a new exploitation of VisIVO that combines hybrid workflows and streaming frameworks to address the challenges of large-scale visualization in the petascale era.

Specifically, we describe the integration of VisIVO with the StreamFlow [4] workflow engine[3], enabling flexible hybrid execution across cloud and HPC environments, and with CAPIO[4] [5], a middleware that injects streaming semantics into legacy file-based workflows. Together, these technologies enhance portability, reproducibility, and performance. We demonstrate their impact through applications to cutting-edge cosmological simulations, including galaxy mergers with the ChaNGa code, large scale cosmological simulations with the OpenGadget3 code, and GADGET-based large-volume neutrino cosmologies within the [6, 7] DEMNUni suite.

Related Works

The demand for scalable visualization and workflow solutions in astrophysics has led to the development of numerous frameworks across scientific computing.

In visualization, ParaView [8] has long been established as a scalable environment for analyzing massive datasets in distributed HPC systems, leveraging parallel rendering and client-server architectures for remote visualization. The yt [9] project provides a Python-based toolkit tailored for astrophysical simulations, supporting multiple codes (e.g., Enzo, GADGET) with flexible volumetric and particle-based rendering. While powerful, both ParaView and yt operate primarily as standalone analysis environments and lack native integration with workflow abstractions or transparent streaming mechanisms.

Workflow models and systems have long been used to orchestrate large-scale computational scientific experiments [10–13]. Their advantages arise primarily from two factors. First, the loose coupling between coordination semantics and the underlying execution environment promotes portability, reproducibility, and reusability. Second, the explicit decomposition of complex processes into discrete steps and their dependencies allows the workflow system to handle concerns such as fault tolerance [14], provenance collection [15], and automatic optimization of the execution graph [16].

For these reasons, workflows have been widely adopted across a broad range of scientific disciplines. In bioinformatics, systems such as Galaxy [17], Snakemake [18], and Nextflow [19] provide curated catalogs of hundreds of composable computational tools. In computational materials science, AiiDA [20] enables domain experts to automate complex numerical procedures and execute them at scale on HPC infrastructures. In the astrophysics domain, systems such as Pegasus [21] and the engines behind large astronomical surveys like the LSST pipeline [22] provide robust frameworks for automating multi-stage processing with an emphasis on reproducibility and provenance. However, these rely on file-based semantics and do not natively support in-memory streaming between workflow stages, which can become a bottleneck for data-intensive visualization tasks.

I/O performance optimization has also been widely studied. Staging services have been proposed to alleviate file system pressure by prefetching or buffering data [23], while cloud-aware scheduling frameworks have incorporated provenance and data locality [24]. These methods highlight the importance of I/O management, yet they do not target streaming integration within legacy pipelines.

Our work advances beyond these approaches by combining three dimensions into a unified framework: (i) workflow abstraction through StreamFlow, enabling dynamic assignment of workflow stages to heterogeneous environments; (ii) I/O streaming through CAPIO, which transparently injects in-memory data transfer into file-based pipelines without modifying application code; and (iii) a visualization toolkit, VisIVO, optimized for large-scale astrophysical data. This integration uniquely enables portable, reproducible, and scalable visualization pipelines tailored for astrophysics and cosmology in the exascale era.

[1] https://www.space-coe.eu

[2] https://www.supercomputing-icsc.it/

[3] https://streamflow.di.unito.it/

[4] https://capio.hpc4ai.it/

VisIVO Modular Applications

To render the visualization of A&C simulation outcomes, VisIVO Server employs three modules for the following tasks: data importing, filtering, and viewing.[5] The data importing task converts the supplied datasets (originally in different formats) into an internal binary format named VisIVO Binary Table (VBT), which is a highly-efficient data representation internally used by VisIVO Server. A VBT consists of a header file (extension .bin.head) containing all necessary metadata, and a raw data file (extension .bin) storing actual data values. E.g., the header may contain information regarding the overall number of fields and the number of points for each field (for point datasets) or the number of cells and relevant mesh sizes (for volume datasets). The raw data file is typically a sequence of values, e.g., all X followed by all Y values.

The data filtering module can perform several types of operations, for example data filtering operations (e.g. *Randomization*, *Decimation*) to reduce the final resolution, the application of mathematical or statistical operators, and cosmological post-processing. Among the latter, there are three commonly used mass assignment methods [25] that calculate the particle densities, i.e., the nearest grid point (NGP), the cloud-in-cell (CIC), and the triangular-shaped cloud (TSC) methods.

Finally, the visualization process creates multi-dimensional views from the data that must fit the available RAM. These kinds of visualization include data points, volumes, and vectors, and are based on the Visualization Toolkit (VTK).[6]

Figure 1 depicts the typical visualization pipeline of VisIVO for processing cosmological snapshots, consisting of the application of the three main modules: `VisIVOImporter` and `VisIVOFilter`, for calculating the particle densities, and `VisIVOViewer`, for final renderings.

Methodology

The StreamFlow Methodology

We employed workflow abstractions to allow a portable representation of the VisIVO modular applications and their resource requirements. The integration of VisIVO with the StreamFlow workflow management system fosters reproducibility and maintainability, allowing users to take advantage of heterogeneous HPC facilities (including

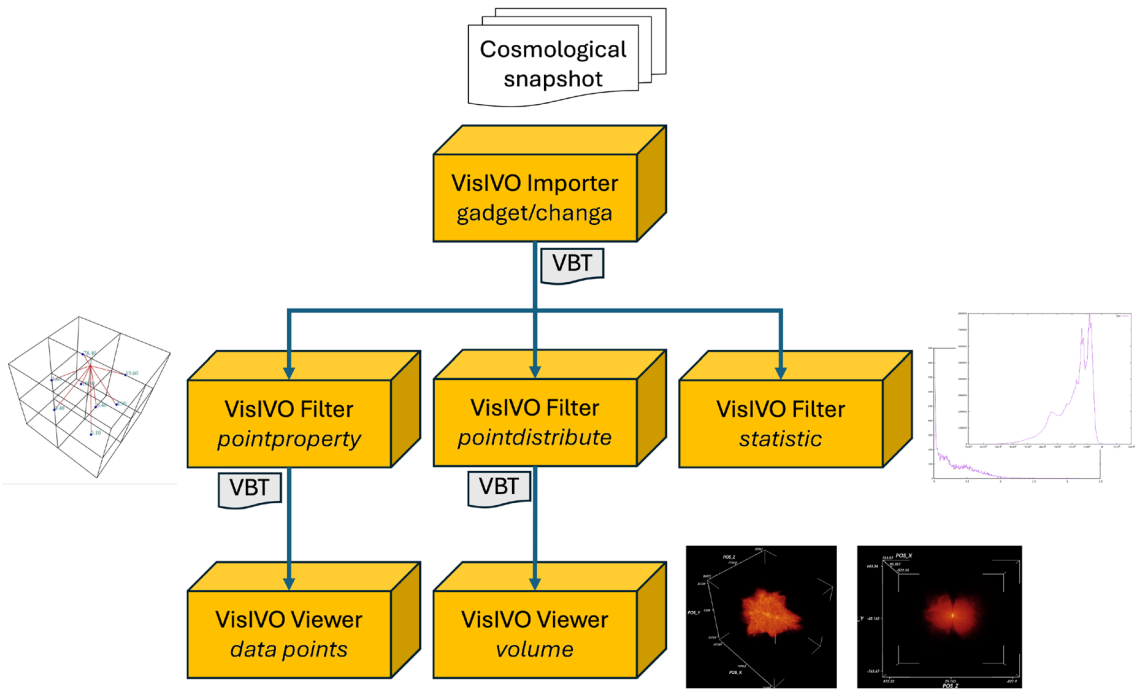


Fig. 1 Visualization pipeline of VisIVO. A schematic representation of the VisIVO pipeline applied to cosmological simulation snapshots (from ChaNGa and GADGET-based codes). The pipeline consists of three modules: (i) `VisIVOImporter`, which converts simulation outputs into the efficient VisIVO Binary Table (VBT) format; (ii) `VisIVOFilter`, which applies operations such as density computation or statistical analysis; and (iii) `VisIVOViewer`, which produces particle- and volume-based renderings. This modular structure enables scalable visualization of astrophysical simulations.

[5] https://visivo.readthedocs.io/en/latest/

[6] https://vtk.org/

mixed cloud-HPC resources) while minimizing data transfer overheads.

StreamFlow is a container-native Workflow Management System (WMS), developed in Python 3 and based on the Common Workflow Language (CWL) [26] open standard.[7] It has been designed with two key goals in mind: first, to enable the execution of tasks in *multi-container environments*, thereby supporting the concurrent execution of multiple communicating tasks within a multi-agent ecosystem; and second, to relax the requirement for a single shared data space, thus facilitating *hybrid workflow executions* across multi-cloud deployments or hybrid cloud/HPC infrastructures.

When targeting homogeneous execution environments (e.g., a single node, an HPC cluster, or a cloud orchestrator), a WMS can make simplifying assumptions about the underlying infrastructure. A common assumption is the availability of a shared data space, such as a local disk, a cluster-wide parallel file system, or a shared object store in the cloud. Additional assumptions may concern transport layers (e.g., SSH, TCP connections, or WebSocket APIs), resource allocation mechanisms, and authentication.

In heterogeneous workflow executions, these assumptions no longer hold, and the WMS must explicitly manage tasks such as data movement across environments. The StreamFlow WMS extends a standard workflow graph, composed of steps and their inter-dependencies, with a declarative description of the potentially complex topology of execution environments. This hybrid workflow model [27] enables the execution of different steps across multiple sites without requiring shared data spaces, authentication protocols, or bidirectional network channels. The StreamFlow control plane manages all aspects of workflow execution, including data transfers, fault tolerance, and provenance. This approach has proven sufficiently general to orchestrate large-scale workflows across heterogeneous cloud/HPC environments [4, 28], multiple HPC facilities [29, 30], and classical/quantum systems [31].

With the help of hybrid workflows, different VisIVO modules can run in a multi-container environment, seamlessly switching between different container runtimes (e.g., Docker or Singularity). Moreover, independent workflow steps can be executed concurrently on top of multi-agent ecosystems (cloud/HPC) [10]. Figure 2 shows an example of a complex deployment model for the VisIVO workflow. In this setting, the importer and filters modules of VisIVO are executed on the HPC platform, where cosmological simulations are produced, while the viewer modules are deployed on a Cloud infrastructure where a data or web service may be present to easily access the produced images.

In addition, using the CWL format for workflow design ensures the portability and reproducibility of VisIVO workflows while avoiding technology lock-in. CWL benefits from strong compatibility layers with a wide range of workflow systems (e.g., Galaxy and Airflow[8]), standards (e.g., the Workflow Run RO-Crate profiles for provenance collection [15]), and services (e.g., the WorkflowHub platform for publishing [32]).

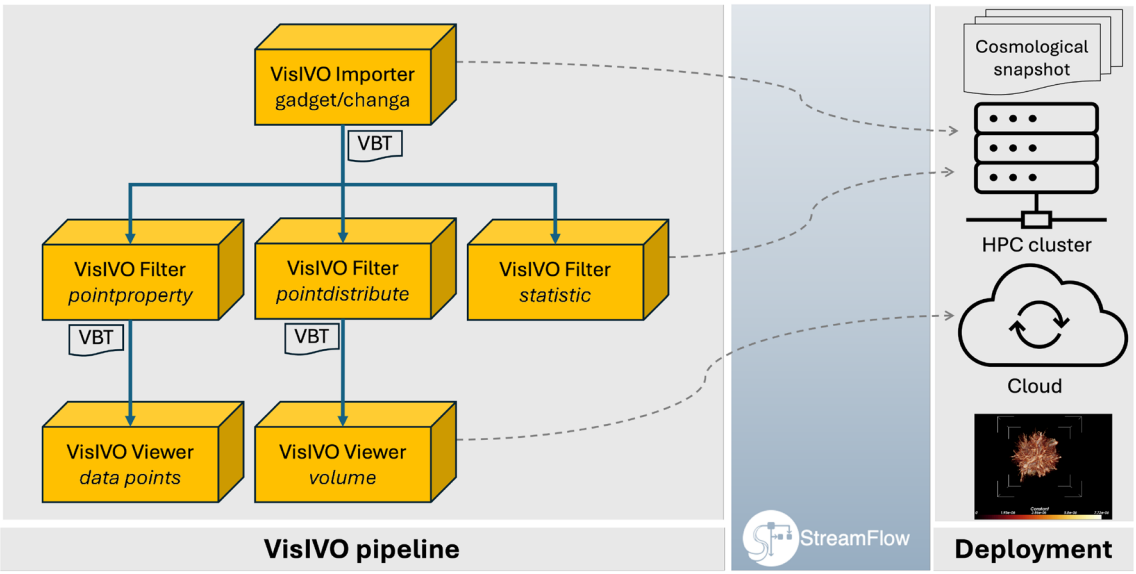


Fig. 2 Hybrid deployment model for VisIVO workflows. An example of how VisIVO modules can be distributed across heterogeneous infrastructures. Simulation data are imported and filtered on an HPC system, while rendering tasks are executed on cloud services, where visualization results can be accessed interactively. This deployment strategy minimizes data movement while enabling portability and reproducibility.

[7] CWL, https://www.commonwl.org/

[8] https://airflow.apache.org/

The CAPIO Methodology

The CAPIO methodology aims at improving the performance of file-based workflows without the necessity to modify the workflow step's source code. At its core, CAPIO is composed of two parts (see Figure 3): a coordination language, called CAPIO-CL [33, 34], which aims at annotating dependencies on files within file-based workflows, and a middleware, named CAPIO middleware [5], which is the reference implementation of CAPIO-CL. The CAPIO middleware improves the performance of file-based workflows by enforcing the semantics expressed in the CAPIO-CL configuration file (thus, injecting streaming capabilities), and by performing I/O operations in memory instead of in the file system (when possible).

CAPIO-CL

CAPIO-CL is a coordination language designed to manage file dependencies in file-based workflows. It uses JSON for its syntax and organizes its semantics into two main categories: **fire rules**, which determine when data produced by a workflow step can be consumed, and **commit rules**, which specify when a data stream is considered complete (i.e., the end-of-file or EOF is reached). CAPIO-CL operates exclusively on files and directories located within a predefined virtual mount point, `CAPIO_DIR`, ensuring that rules only apply to data within this directory.

Fire Rules

CAPIO-CL defines two types of fire rules: *Fire no Update* (FnU) and *Fire on Commit* (FoC). **Fire no Update** allows a consumer step to begin reading a file as soon as the producer writes any data. In contrast, **Fire on Commit** delays consumption until the file is fully written and explicitly committed by the producer. By choosing *FnU*, streaming between steps is enabled, with the commit rule serving only to signal the end of data. However, if a file is modified in place (i.e., previously written portions are changed), streaming is not permitted. In such cases, the consumer must wait for the file to be committed before accessing it.

Commit Rules

Commit rules define when a file is considered complete, which is essential for notifying consumer steps in streaming scenarios. CAPIO-CL provides three commit rules:

- **Commit on Close** (CoC): a file is committed when the producer calls `close()`; subsequent modifications are disallowed.
- **Commit on Termination** (CoT): used when a workflow step opens and closes a file multiple times. The file is committed once the producing step terminates.
- **Commit on File**: it establishes a dependency where file f_1 is committed only after file f_2 is committed.

Additional modifiers exist (e.g., specifying the number of times a file is closed), but are beyond the scope of this paper. For directories, CAPIO-CL defines the **Committed n-Files** (CnF) rule, which considers a directory committed once it contains at least *n* entries.

CAPIO-CL Syntax

CAPIO-CL semantics is expressed via JSON, in a configuration file referred to as the CAPIO-CL configuration file. This file contains five sections, two of which are mandatory: **name** and **IO_Graph**. The **name** section identifies the workflow, allowing CAPIO to distinguish among multiple workflows. The **IO_Graph** section describes the file dependencies between workflow steps, with the following fields:

- *name*: the name of the workflow step;
- *input_stream*: a list of file paths[9] consumed by the step;
- *output_stream*: a list of file paths produced by the step;
- *streaming*: it defines coordination rules, including the fire rule (via *mode*) and commit rule (via *committed*). This section uses *name* (for files) or *dirname* (for directories), both referring to entries in *output_stream*, and can optionally include *n_files* to specify a required number of directory entries.

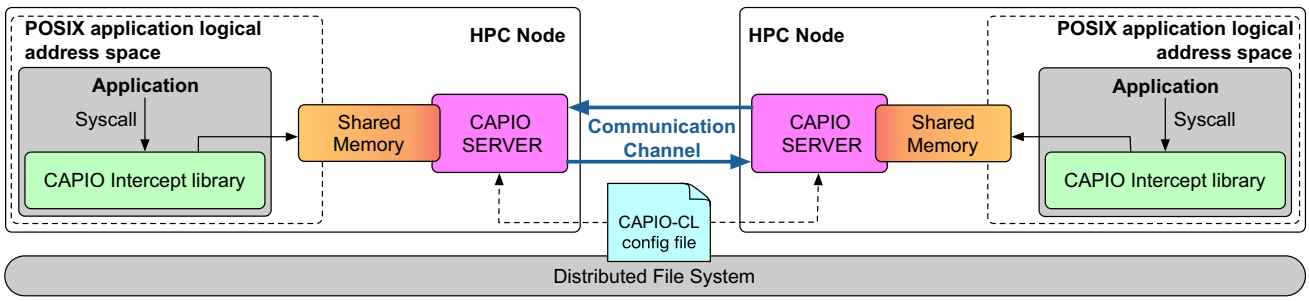


Fig. 3 Overview of the CAPIO middleware architecture.

[9] CAPIO-CL supports wildcards: * it matches any sequence of characters, ? matches a single character.

The three optional sections are as follows:

- **exclude**: it lists files to ignore within `CAPIO_DIR`;
- **permanent**: it specifies files that should persist after the workflow finishes;
- **home_node_policies**: it designates the node responsible for storing specific files.

For further details, the reader is referred to the official CAPIO-CL documentation.[10]

The CAPIO Middleware

The CAPIO middleware[11] is a userspace implementation of the CAPIO-CL coordination language. This means that there are no requirements for root permissions. The CAPIO middleware enhances the performance of file-based workflows through two primary mechanisms:

- **Streaming Injection**: CAPIO relaxes POSIX semantics based on the coordination rules defined in the CAPIO-CL configuration file;
- **In-Memory Filesystem**: CAPIO enables workflows to read and write files directly in node-local memory, thereby avoiding the distributed file system or at least bypassing shared network mounts.

These performance improvements are transparently achieved by intercepting system calls related to file I/O.

Figure 3 illustrates the CAPIO middleware architecture. The CAPIO system intercepts POSIX file I/O calls and transparently injects streaming semantics into file-based workflows. Each HPC node hosts a CAPIO server and a POSIX interception library, enabling applications to exchange data directly in memory rather than through the distributed file system. As illustrated in Figure 3, the CAPIO middleware consists of two main components:

- A *server component*, responsible for executing CAPIO-CL logic, managing storage, and handling inter-node communication;
- A *POSIX interception library*, built on a customized fork of `syscall_intercept`[12] which allows the interception and redirection of system calls before they enter kernel mode.

System calls that require handling are forwarded to the local server via a shared-memory queue. If necessary, the local server can query remote servers to resolve requests collaboratively.

CAPIO is designed to operate dynamically. Unlike traditional solutions (such as MPI-IO [35]), each server instance functions independently, eliminating the need for coordinated startup mechanisms such as MPI (though MPI remains compatible). This design facilitates seamless orchestration (and future transparent integration) by WMS such as StreamFlow [4] and DagonStar [36–38], which leverage HPC job schedulers such as Slurm. Finally, CAPIO supports multiple backends. It can operate in a file-system-based mode (CAPIO-FS), where only POSIX semantics are relaxed, or in a high-performance in-memory mode (CAPIO-MEM), where files are exchanged between servers using communication layers such as UCX/UCS, MQTT, TCP sockets, or MPI.

To apply the CAPIO methodology to the VisIVO workflow, given the size of the dataset, we had to use the CAPIO-FS backend, which only requires a shared file system mounted on each compute node, from which the workflow will read and write files.

VisIVO Workflows

We developed two CWL workflows to automate the visualization of astrophysical simulation data using the VisIVO suite. Each workflow consists of a sequence of three components: `VisIVOImporter`, `VisIVOFilter`, and `VisIVOViewer` (Figure 1).

Workflow$_1$ generates a point-based visualization of particles, colored according to their computed densities. Simulation snapshots from ChaNGa or GADGET-based simulation codes are first converted into VisIVO Binary Tables using the `VisIVOImporter`. A density field is then computed with the Cloud-in-Cell algorithm, merged with the original particle data, and rendered with the `VisIVOViewer` to produce density-colored particle images.

Workflow$_2$ produces a volumetric rendering of the particle density field. Following data import, the particle distribution is mapped onto a regular 3D grid using the *PointDistribute* filter, creating a volumetric density field. The workflow then computes basic statistics before generating volumetric renderings through the `VisIVOViewer`.

Both workflows are publicly available under the Apache 2.0 license in the VisIVOCWL GitHub repository,[13] where they are listed as workflows n.3 and n.5 in the `README.md`.

[10] https://capio.hpc4ai.it/docs/

[11] https://github.com/High-Performance-IO/capio

[12] http://github.com/alpha-unito/syscall_intercept/ forked from https://github.com/pmem/syscall_intercept

[13] https://github.com/VisIVOLab/VisIVOCWL

Workflow Components

The initial step, common to both workflows, is executed by `VisIVOImporter`, which converts a snapshot produced by ChaNGA or a GADGET-based code into VBTs.

In the first workflow, the `VisIVOImporter` step executes the script in Listing 1 for the ChaNGa code output:

```
$ VisIVOImporter --fformat changa --out NewTable --file snapdir/snap_name
```

Listing 1 VisIVOImporter step command for ChaNGA snapshots

The script above converts a ChaNGa snapshot in Tipsy format and produces a new VBT for each type of particle in the snapshot. In our specific case, it produces three files, NewTableDARK.bin, NewTableGAS.bin, NewTableSTARS.bin, and their corresponding header files, NewTableDARK.bin.head, NewTableGAS.bin.head, and NewTableSTARS.bin.head. The header files contain information about the dark matter, gas components, and stars of the input ChaNGa snapshot.

In the second case of the snapshots produced by a GADGET-based code, it runs in parallel using `AA` OpenMP threads and `BB` MPI processes, as shown in Listing 2.

```
$ export OMP_NUM_THREADS=AA
$ mpirun --np BB VisIVOImporter --fformat gadget --out NewTable --file snapdir/
    snap_name
```

Listing 2 VisIVOImporter for GADGET snapshots running with MPI and OpenMP

```
$ VisIVOFilter --op pointproperty --resolution X_RES Y_RES Z_RES  --points POS_X
    POS_Y POS_Z --out density.bin --outcol density --file NewTableHALO.bin
```

Listing 3 VisIVOFilter PointProperty operation command

```
$ VisIVOFilter --op merge --out NewTableHALOMerge.bin  --filelist tabselection.txt
```

Listing 4 VisIVOFilter merge operation commandThe final visualization step renders the particle distribution, as shown in Listing 5.

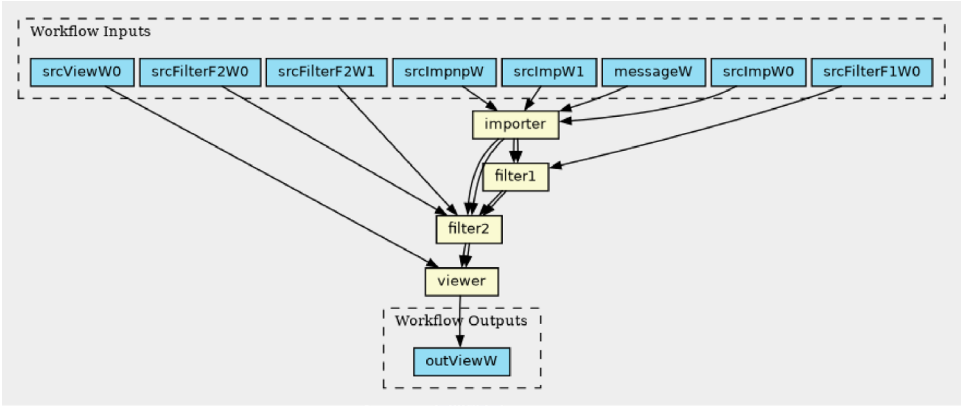
```
$ VisIVOViewer --x POS_X_tab_1 --y POS_Y_tab_1 --z POS_Z_tab_1 --color
    --colorscalar density_tab_2 --colortable volren_glow --logscale --out
    VisIVOServerImage NewTableHALOMerge.bin
```

Listing 5 VisIVOViewer particle rendering command

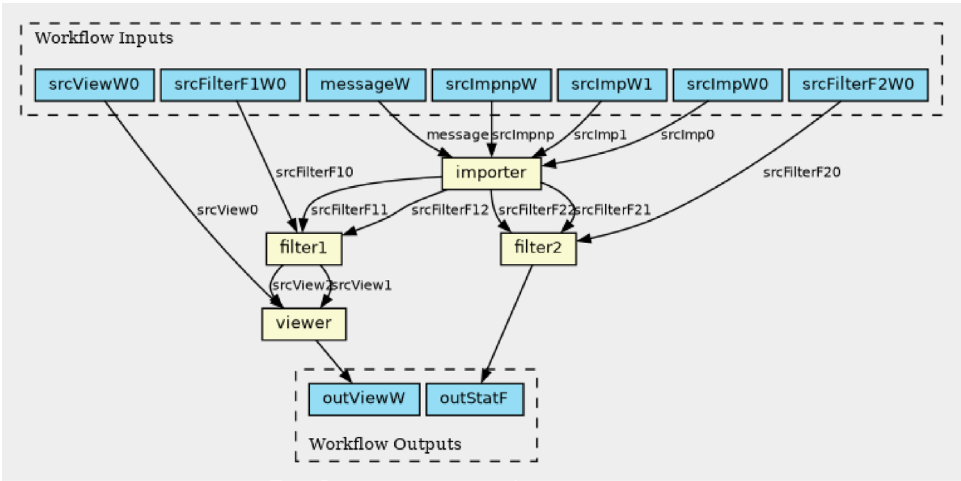
This command generates VBTs for each particle species, such as `NewTableHALO.bin` and `NewTableGAS.bin`, along with the corresponding header files. For the remainder of the workflow, we focus on the halo data (`NewTableHALO`).

Workflow$_1$ The first workflow proceeds with two filtering steps:

1. **Density Computation:** A temporary volumetric dataset is created using the Cloud-in-Cell (CIC) algorithm, and density values are assigned to particles with the *Point-Property* operation, following the script shown in Listing 3.
2. **Table Merge:** The original table is merged with the newly computed density field using the script as shown in Listing 4.



(a) **Workflow**$_1$: performing the particle rendering using the density, computed by the Filter step, as the color field.



(b) **Workflow**$_2$: computing particles statistics and rendering the volume visualization of the particle densities.

This produces four PNG images (`VisIVOServerImageXX.png`) showing the density-based point rendering.

Workflow$_2$ The second workflow uses a filter step to compute a volumetric density field, as shown in Listing 6.

The *PointDistribute* operation distributes density values onto a regular grid using the CIC algorithm (default), generating a volumetric representation stored in `densityvolume.bin`.

An additional filter step calculates the snapshot statistics as shown in Listing 7.

```
$ VisIVOFilter --op pointdistribute --resolution X_RES Y_RES Z_RES --points POS_X
    POS_Y POS_Z --out densityvolume.bin --file NewTableHALO.bin
```

Listing 6 VisIVOFilter PointDistribute operation command

```
$ VisIVOFilter --op statistic --histogram --out results.txt --file NewTableHALO.bin
```

Listing 7 VisIVOFilter statistic operation command

The final visualization step performs volume rendering, as shown in Listing 8.

```
$ VisIVOViewer --volume --vrendering --vrenderingfield Constant --color
    --colortable volren_glow --showlut --out img --file densityvolume.bin
```

Listing 8 VisIVOViewer volume rendering command

Like Workflow$_1$, this produces four PNG images using the specified rendering parameters.

Workflow Execution and Orchestration

The workflows are implemented as `.cwl` files, which declare a CWL `Workflow` composed of steps implemented as `CommandLineTool` classes. Docker containers are bound to each step using the `DockerRequirement` directive.

VisIVO workflows expressed in CWL are illustrated in Figure 4. Two workflows were developed to automate visualization of astrophysical simulation data: Figure 4a shows the Workflow$_1$ that generates particle-based renderings with density fields computed via the Cloud-in-Cell method; Figure 4b depicts the Workflow$_2$ which computes volumetric density fields and corresponding statistics, followed by volume rendering. Each workflow specifies inputs, outputs, and dependencies among VisIVO modules, enabling reproducibility and portability.

Results

This section reports the results of VisIVO workflows (described in Section VisIVO workflows) achieved using the computing infrastructures presented in Section Computing Infrastructure on the testing dataset described in Section Data description through the integration with CAPIO (Section CAPIO Configuration and Performance results) and StreamFlow (Section StreamFlow Hybrid Execution results) and, finally, the rendering results (Section Rendering results). This evaluation highlights the portability, scalability, and performance improvements of the overall methodology.

Computing Infrastructure

The VisIVO workflows have been tested on two computing infrastructures, namely the HPC4AI@UNITO[14] infrastructure [39] and the Galileo100 HPC cluster at CINECA.[15]

HPC4AI@UNITO

The VisIVO with CAPIO beneath is running on the C3S Broadwell partition hosted at the HPC4AI@UNITO infrastructure. The configuration of each node of the Broadwell partition is defined in the following way:

- CPU - Intel Xeon E5-2697 v4 @ 2.30GHz
- MEMORY - 125 GB
- NETWORK - InfiniBand @ 100Gb/s
- File System - BeeGFS

The execution of the VisIVO workflows with StreamFlow is deployed on the cloud-HPC continuum environment at the HPC4AI@UNITO infrastructure. The cloud partition consists of the Virtual Machine (VM) with 48 cores and 300 GB RAM . The VM establishes a connection to the HPC frontend via SSH over the public internet, achieving a transfer speed of 100 MB/s. In the hybrid execution, the HPC partition employed is the Broadwell partition defined above.

CINECA Galileo100

On the other hand, 16 nodes of the CINECA Galileo100 HPC facility (48 cores, 384 GB RAM, 2x NVIDIA V100 GPUs each) have been used to render the snapshots of the GADGET data presented in Section GADGET data from DEMNUni suite. The related data have been generated in the same facility and require 50 TB of storage, making it difficult to move them into another available infrastructure at our premises.

Data Description

For our tests we used ChaNGa[16] (Charm N-body GrAvity solver) and GADGET (GAlaxies with Dark matter and Gas) simulation snapshots (produced by the OpenGadget3[17] code and the DEMNUni suite).

[14] https://hpc4ai.unito.it/

[15] https://www.cineca.it/en/data-center/hpc-infrastructure

[16] https://github.com/N-BodyShop/changa

[17] https://gitlab.lrz.de/MAGNETICUM/Hydro-OpenGadget3

ChaNGa Data

ChaNGa is a code designed for collisionless *N*-Body simulations. It supports a range of applications, from cosmological simulations with periodic boundary conditions to the study of isolated stellar systems. The code incorporates hydrodynamics through the Smooth Particle Hydrodynamics [40] (SPH) method and performs gravity calculations using a Barnes-Hut tree algorithm [41]. This algorithm leverages hexadecapole node expansions and Ewald summation [42] to account for periodic forces. ChaNGa employs a leapfrog integrator for time evolution, allowing particles to have individual timesteps. The simulation outputs are stored in the TIPSY data format,[18] ensuring compatibility with standard analysis tools.

GADGET Data from OpenGadget3

OpenGadget3 is a collisionless N-Body/Lagrangian cosmological code that solves the Vlasov-Poisson equations in a cosmological expanding framework. It uses the SPH computational method to describe the motion of fluids in addition to the gravitational forces. The gravitational problem is numerically solved by coupling a Particle-Mesh algorithm for the average field and a tree-based Barnes&Hut algorithm for the interaction with the close neighborhood. The code allows running simulations in a full cosmological context, i.e., accounting for an expanding background and the presence of matter, both "dark" and baryonic (the ordinary matter), and dark energy. Although the full cosmological context is often the default choice, having a non-expanding background and a setup with only dark or baryonic matter is equally possible.

GADGET Data from DEMNUni Suite

Finally, the presented workflows have supported the studies of the effects of massive neutrinos on the evolution of the large-scale structures of the Universe from the so-called "Dark Energy and Massive Neutrino Universe" (DEMNUni) suite [43, 44]. The DEMNUni simulations have been performed using the tree particle mesh-smoothed particle hydrodynamics (TreePM-SPH) code GADGET-3[6], specifically modified by [7] to account for the presence of massive neutrinos. This modified version of GADGET-3 follows the evolution of Cold Dark Matter (CDM) and Hot Dark Matter (HDM) neutrino particles, treating them as two separated collisionless species.

For the particular case of this work, we have considered the DEMNUni subset of simulations with a starting redshift $z_{in} = 99$, and characterized by a comoving volume of $(500\ \ h^{-1}\ \text{Mpc})^3$ filled with 2048^3 dark matter particles and, when present, 2048^3 neutrino particles [45, 46]. Given the large amount of memory required by the simulations, baryon physics was not included. We consider two different simulations, with a baseline *Planck* cosmology [47], namely a flat ΛCDM model generalized to a ν ΛCDM framework by changing only the value of the sum of the three active neutrino masses $M_\nu = \sum_i m_{\nu,i} = 0.16$ eV, and keeping fixed $\Omega_{\rm m}$ and the amplitude of primordial curvature perturbations. This subset of simulations is characterized by a softening length $\varepsilon = 5\ \ h^{-1}$ Kpc, and has been run on the Marconi100 supercomputer at CINECA, Italy.[19] For each simulation, we have produced 64 outputs logarithmically equally spaced in the scale factor $a = 1/(1+z)$, in the redshift interval $z = 0 - 99$, 50 of which lay between $z = 0$ and $z = 10$. For each of the 64 output times, we have dumped on-the-fly a particle snapshot composed by both CDM and neutrino particles.

CAPIO Configuration and Performance Results

To execute the VisIVO pipelines with the CAPIO middleware, we first had to define the CAPIO-CL configuration file as detailed in the following section.

CAPIO-CL Configuration File to Execute VisIVO Pipelines

To execute the VisIVO pipelines with the CAPIO middleware, we first had to define the CAPIO-CL configuration file. This file was made generic, meaning that it is the same for all three different pipelines that were tested with the CAPIO middleware. Listing 9 shows the CAPIO-CL configuration file used to test the three VisIVO pipelines.

[18] https://github.com/N-BodyShop/tipsy

[19] https://www.cineca.it/

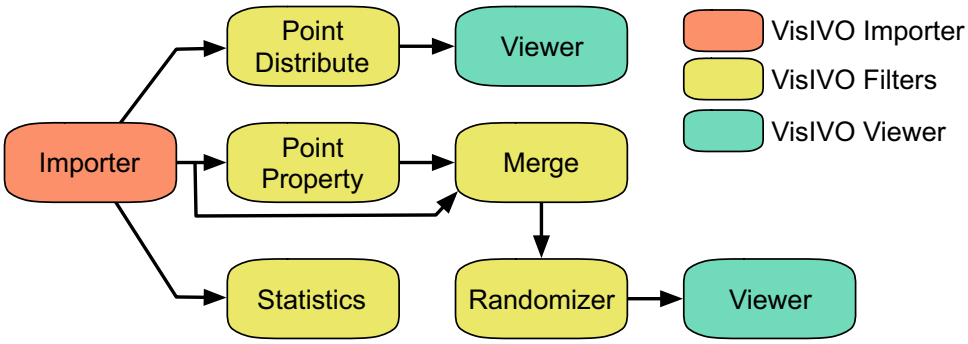


Fig. 5 The structure of the VisIVO workflow executed with CAPIO, consists of an initial Importer step, one or more Filter operations, and one or more Viewer renderings.

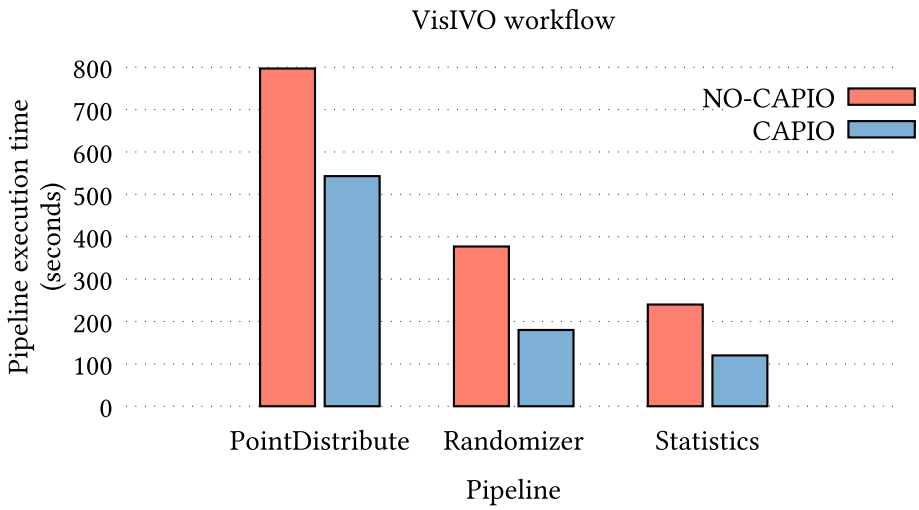


Fig. 6 Performance of VisIVO workflow when executing with different filter modules, with and without CAPIO. Average execution time for three representative filter operations (*PointDistribute*, *Randomizer*, and *Statistics*) applied to a 257 GB GADGET dataset from the DEMNUni suite. The CAPIO-FS backend delivers performance improvements of 15-50%, with the largest gains observed for the *Statistics* filter, which benefits most from streaming-based parallelization.

```
{
    "name":"visivo",
    "IO_Graph":[
        {
            "name" : "importer",
            "input_stream" : ["box_256/snap_*"],
            "output_stream" : ["GAS.bin*","HALO.bin*","STARS.bin*"],
            "streaming" : [
                {
                    "name" : ["GAS.bin*","HALO.bin*","STARS.bin*"],
                    "committed" : "on_close",
                    "mode" : "no_update"
                }
            ]
        },{
            "name" : "filter",
            "input_stream" : ["GAS.bin*","HALO.bin*","STARS.bin*"],
            "output_stream" : ["randomizer.bin*","statistics.bin*",
                               "pointdistribute.bin*"],
            "streaming" : [
                {
                    "name" : ["randomizer.bin*","statistics.bin*",
                              "pointdistribute.bin*"],
                    "committed" : "on_close",
                    "mode" : "no_update"
                }
            ]
        },{
            "name" : "viewer",
            "input_stream" : ["randomizer.bin*","statistics.bin*",
                              "pointdistribute.bin*"],
            "output_stream" : ["*.png"]
            "streaming" : []
        }
    ]
}
```

Listing 9 The configuration file for the VisIVO workflow, which higlights the mostimportant aspects needed by CAPIO to correctly add streaming functionality withinthe workflow

With the CAPIO-CL configuration shown in Listing 9, the VisIVO pipeline was executed with the CAPIO middleware beneath. All the scripts reported here are based on the Slurm queue manager. The Slurm script describing the submission of the importer is presented in Listing 10.

To execute the filter step, the bash script described in Listing 11 can be used. Note that we use the *PointDistribute* filter in our example; however, to execute the *randomizer* or *PointProperty* filter, users need only to change the filter step parameters.

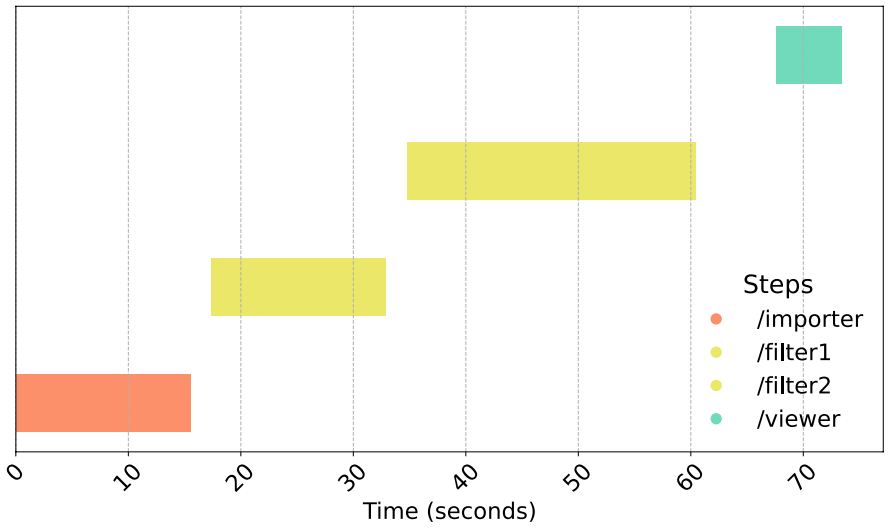
```
#!/bin/bash
#SBATCH [SLURM config params...]

export CAPIO_DIR=./visivo_workdir
export CAPIO_WORKFLOW_NAME=visivo
export CAPIO_APP_NAME=importer


srun --overlap --exact --export=ALL -N $SLURM_NNODES -n $SLURM_NNODES
    --ntasks-per-node=1 capio_server -c visivo-all.json > logs/server.log &
CAPIO_SERVER_PID=$!

srun --overlap -N 1  --export=ALL,LD_PRELOAD=libcapio_posix.so \
        VisIVOImporter --fields POS  --fformat gadget          \
        --out visivo_out_dir/halo.bin  box_256/snap_062.0

kill $CAPIO_SERVER_PID
```

Listing 10 Extract of the Slurm submission script to start the VisIVOImporter withCAPIO

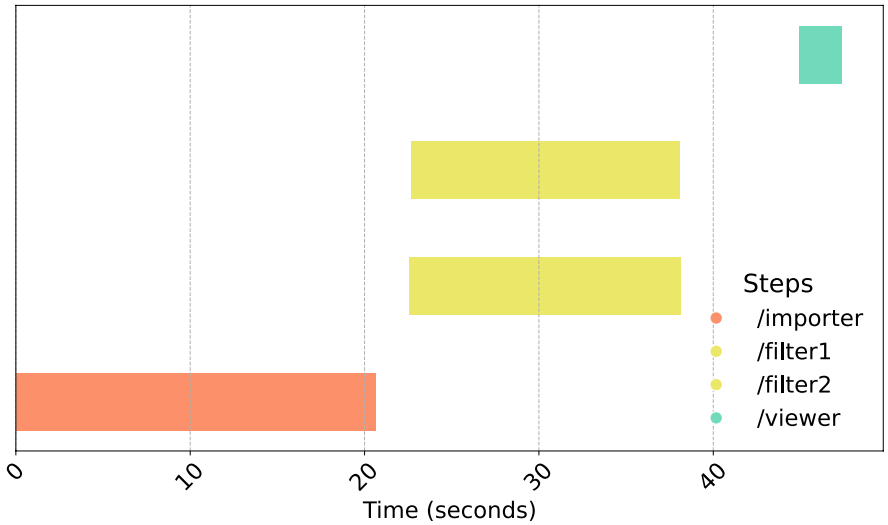
```
#!/bin/bash
#SBATCH [SLURM config params...]

export CAPIO_DIR=./visivo_workdir
export CAPIO_WORKFLOW_NAME=visivo
export CAPIO_APP_NAME=importer


srun --overlap --exact --export=ALL -N $SLURM_NNODES           \
        -n $SLURM_NNODES --ntasks-per-node=1                   \
        capio_server -c visivo-all.json > logs/server.log &
CAPIO_SERVER_PID=$!

srun --overlap -N 1  --export=ALL,LD_PRELOAD=libcapio_posix.so \
        VisIVOFilter --op pointdistribute --resolution 512 512 512 \
        --points POS_X POS_Y POS_Z --out visivo_out_dir/pointdistribute.bin \
        visivo_out_dir/HALO.bin

kill $CAPIO_SERVER_PID
```

Listing 11 Extract of the Slurm submission script to start the VisIVOFilter withCAPIO

Finally, to visualize the filter output data, the `VisIVOViewer` can be submitted as described in Listing 12.



(a) **Workflkow**$_1$: hybrid execution on cloud-HPC.



(b) **Workflow**$_2$: hybrid execution on cloud-HPC.

Fig. 7 VisIVO workflows hybrid (cloud-HPC) execution orchestrated by StreamFlow.

```
#!/bin/bash
#SBATCH [SLURM config params...]

export CAPIO_DIR=./visivo_workdir
export CAPIO_WORKFLOW_NAME=visivo
export CAPIO_APP_NAME=importer


srun --overlap --exact --export=ALL -N $SLURM_NNODES           \
        -n $SLURM_NNODES --ntasks-per-node=1                   \
        capio_server -c visivo-all.json > logs/server.log &
CAPIO_SERVER_PID=$!

srun --overlap -N 1  --export=ALL,LD_PRELOAD=libcapio_posix.so \
        VisIVOViewer  --volume --vrendering                    \
                      --vrenderingfield Constant               \
                      --autorange --colortable volren_glow     \
                      visivo_out_dir/pointdistribute.bin

kill $CAPIO_SERVER_PID
```

Listing 12 Extract of the Slurm submission script to start the VisIVOViewer withCAPIO.

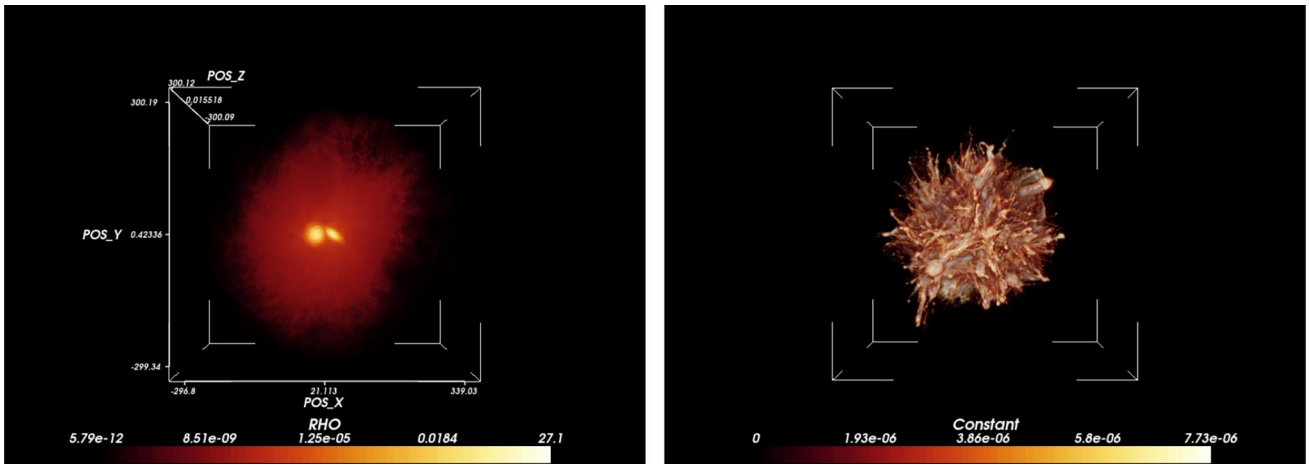


Fig. 8 Example particle and volume renderings from VisIVO workflows. Left: particle-based rendering of gas distributions from ChaNGa galaxy merger simulations, with density computed using the *PointProperty* filter. Right: volumetric rendering of large-scale structure evolution from OpenGadget3 simulations, with voxel density computed via the *PointDistribute* filter. Colors indicate particle or voxel density, enabling intuitive exploration of astrophysical processes.

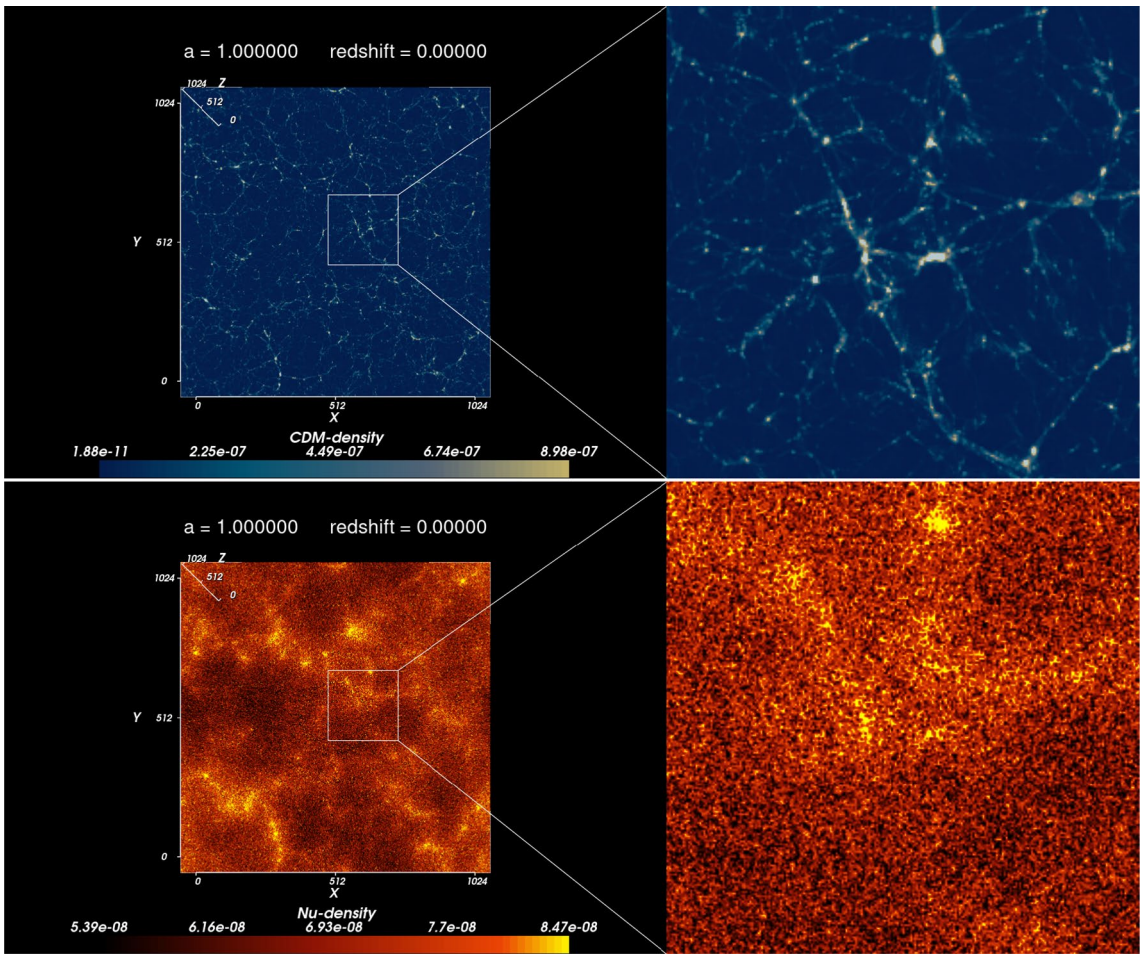


Fig. 9 Volume renderings of DEMNUni simulations. Top panel: cold dark matter (CDM) density distribution at redshift z = 0 from the DEMNUni cosmological suite. Bottom panel: corresponding rendering of neutrino particle density from simulations that include massive neutrinos. Both visualizations were produced using Workflow$_2$ (*PointDistribute* + volume rendering). These outputs illustrate the ability of the framework to capture the impact of neutrinos on the growth of cosmic structures.

```
{
    "name": "visivo",
    "IO_Graph": [{
        "name": "importer",
        "input_stream": ["box_256/snap_*"],
        "output_stream": ["workdir/HALO.bin*"],
        "streaming": [{
            "name": ["workdir/HALO.bin*"],
            "committed": "on_close",
            "mode": "no_update"
        }]
    },{
        "name": "filter",
        "input_stream": ["workdir/HALO.bin*"],
        "output_stream": ["workdir/pointdistribute.bin*"],
        "streaming": [{
            "name": ["workdir/pointdistribute.bin*"],
            "committed": "on_close",
            "mode": "no_update"
        }]
    },{
        "name": "viewer",
        "input_stream": ["workdir/pointdistribute.bin*"],
        "output_stream": ["workdir/*.png"]
    }]
}
```

Listing 13 The configuration file used to execute the PointDistribute VisIVO pipeline.When running other pipelines, the configuration is extremely similar, and the onlyrequirement is to add/replace files in the input and output stream sections of thevarious steps

Listing 13 shows the CAPIO-CL configuration file used to execute the VisIVO workflow with the CAPIO-FS backend.

CAPIO Performance Results

Figure 5 shows the logical representation of the VisIVO pipeline (Importer → Filter(s) → Viewer(s)), highlighting the points where CAPIO streaming injection can replace traditional file-based communication.

Figure 6 reports the performance of three `VisIVOFilter` modules (with and without the CAPIO-FS backend) on a 257 GB dataset in GADGET format of a large box-sized cosmological N-body simulation of CDM produced by the DEMNUni suite [44]. The dataset is composed of 512 snapshot files, each of ∼512 MB. In detail, the *PointDistribute* filter is a parallel program that creates a volume from a selected subset of the input table's fields, the *Randomizer* filter creates a random subset from the original data table, and the *Statistics* filter computes the average, minimum, and maximum value of the given fields and creates a histogram of the fields in the input table. Regarding the synchronization semantics for the set of the considered VisIVO workflows, CAPIO can execute all steps with enabled streaming.

As shown in Figure 6, the best computational speed-up is achieved in the last two pipelines when *Randomizer* and *Statistics* filter operations are tested. This is because the computation of the statistics in each field is independent of the others, thus allowing for the full exploitation of the benefits that CAPIO provides through streaming injection. On the other hand, the *PointDistribute* filter requires information about particle positions to compute the final volume; therefore, it requires multiple non-consecutive reads, making the streaming injection less effective than in the *Statistics* filter. Overall, CAPIO-FS delivers performance improvements ranging from 15% (with the *PointDistribute* filter) to nearly 50% in the best-case scenario with the *Randomizer* and *Statistics* filters.

StreamFlow Hybrid Execution results

The VisIVO workflows presented in the previous Section are executed using the StreamFlow command shown in Listing 14, where the `.yml` file specifies all the information required to execute the workflow and defines how its steps are mapped to the execution environment.

```
$ streamflow run streamflow.yml
```

Listing 14 Executing VisIVO with StreamFlow

The content of the `streamflow.yml` file for Workflow$_1$ is depicted in Listing 15.

```
1   version: v1.0
2   workflows:
3       visivo:
4           type: cwl
5           config:
6               file: docker_VisIVO_ImpFilterF1F2View_Workflow_GADGET.cwl
7               settings: docker-job_VisIVO_ImpFilterF1F2View_Workflow_GADGET.yml
8           bindings:
9               - step: /
10                target:
11                  deployment: c3s
12                  service: broadwell
13              - step: /viewer
14                target:
15                  deployment: locally
16  deployments:
17      locally:
18          type: local
19          config: {}
20          workdir: /home/ubuntu/volume/tmp
21      c3s-ssh:
22          type: ssh
23          config:
24              nodes:
25                  - c3sfr1.di.unito.it
26              username: myuser
27          workdir: VISIVO/VisIVOCWL/tmp/ssh
28      c3s:
29          type: slurm
30          config:
31              maxConcurrentJobs: 10
32              services:
33                  broadwell:
34                      file: templates/broadwell.sh
35          workdir: VISIVO/VisIVOCWL/tmp/wf3
36          wraps: c3s-ssh
```

Listing 15 TheStreamFlowconfigurationfileforVisIVOWorkflow$_1$

Rows 6 and 7 specify the CWL description of the workflow (`.cwl`) and the input configuration (`.yml`), respectively. The binding of the steps to the most appropriate execution environment is specified in lines 8-15. In line 13, for example, the viewer step is deployed locally, which in this context means on the VM (cloud partition), since the StreamFlow driver itself runs on the VM. Conversely, line 9 uses the notation `- step: /` as a shorthand to indicate that all other steps, except the viewer, are deployed on C3S (HPC partition), described in Section Computing Infrastructure. Starting from line 16, the configuration defines the deployment environment in detail, including all the necessary parameters.

The execution times of the two workflows are shown in Figure 7. In particular, on Figure 7a , the importer and the two filter steps (represented by the blue, red, and green bars, respectively) are mapped to the HPC environment and the viewer (violet bar) is mapped to the Cloud. The gap between the final filter step (green) and the viewer step (violet) reflects the time required to transfer the data. On the other side, on Figure 7b , the two filter steps are executed in parallel.

Rendering Results

This section showcases the rendering results produced by the last workflow's step, which employs the `VisIVOViewer` module.

Figure 8 shows some exemplary renderings obtained from the VisIVO workflows depicted in Figure 4 on SPACE simulations output of the ChaNGA and the OpenGadget3 code on data described in Section Data description.

The left panel of Figure 8 results from Workflow$_1$. A particle rendering shows two galaxy mergers simulated with ChaNGa and the color bar indicates the density of particles calculated using the *PointProperty* filter module. The right panel of Figure 4 results from Workflow$_2$. A volume rendering shows a cosmological evolution simulated with OpenGadget3 and the color bar indicates the density of voxels calculated using the *PointDistribute* filter module.

The Figure 9 was produced using the Workflow$_2$, where the images are generated from the `VisIVOViewer` volume rendering implementation on the volume generated by the *PointDistribute* filter. The final rendering produced for the DEMNUni simulations shows the time-evolution of the

density of CDM particles as a movie of the different snapshots described in the previous Section Data description.[20]

The top panel of Figure 9 shows the results of the volume rendering of the CDM particles in the final snapshot of the simulation (corresponding to redshift $z = 0$). For a visual comparison, the lower panel of Figure 9 shows a rendering of the density of the neutrinos ν in the simulation that includes neutrino particles.

Conclusions

The rapid growth of data volumes in astrophysics and cosmology poses critical challenges for data access, analysis, and visualization. This work introduces a novel framework that integrates VisIVO with workflow abstractions (via StreamFlow) and transparent I/O streaming (via CAPIO). The resulting system enables portable, reproducible, and high-performance visualization pipelines that seamlessly span cloud and HPC infrastructures.

Our experiments on large-scale cosmological simulations highlight three key advances: (i) the ability to orchestrate VisIVO workflows across heterogeneous environments, (ii) substantial performance improvements from CAPIO-based streaming, with speed-ups of up to 50% in data-intensive filters, and (iii) practical deployment on state-of-the-art simulations of galaxy mergers and neutrino cosmologies.

Looking ahead, we plan to couple this framework with interactive Jupyter Workflow environments [48], further bridging batch-oriented HPC pipelines with notebook-driven data analysis. This integration will empower researchers to interactively explore complex astrophysical datasets at scale, strengthening reproducibility and accelerating discovery in the exascale era.

Adopting a hybrid workflow system allows seamless transitions across heterogeneous infrastructures. Building on this capability, we plan to explore the execution of the VisIVO workflow in a physically distributed environment. In such a setting, individual components of the workflow would be deployed across multiple HPC facilities, with StreamFlow serving as the orchestration layer to manage execution, coordinate dependencies, and oversee data transfers between sites.

Acknowledgements The authors thank C. Carbone for providing the DEMNUni simulations, which were carried out in the framework of "The Dark Energy and Massive-Neutrino Universe" project, using the Tier-0 IBM BG/Q Fermi machine and the Tier-0 Intel OmniPath Cluster Marconi-A1 of the Centro Interuniversitario del Nord-Est per il Calcolo Elettronico (CINECA).

Author Contributions E. Sciacca conceived and designed the project, defined the scientific use cases, and structured the manuscript. E. Sciacca, N. Tuccari, F. Vitello and V. Cesare developed the VisIVO workflows. M. E. Santimaria, A. Mulone, D. Medić, and I. Colonnelli contributed to the integration of VisIVO workflows with StreamFlow and CAPIO. All authors participated in writing, reviewing, and revising the manuscript.

Funding Open access funding provided by Istituto Nazionale di Astrofisica within the CRUI-CARE Agreement. The work has received funding from the European High Performance Computing Joint Undertaking (JU) and Belgium, Czech Republic, France, Germany, Greece, Italy, Norway, and Spain under grant agreement No 101093441 (SPACE CoE). Also, it is supported by the Spoke 1 "FutureHPC & BigData" and the Spoke 3 "Astrophysics and Cosmos Observations" of the ICSC - Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing - and hosting entity, funded by European Union - NextGenerationEU.

Data Availability Raw data were generated at EuroHPC JU infrastructures by the SPACE CoE and the DEMNUni projects. Derived data supporting the findings of this study are available from the corresponding author ES on request.

Declarations

Conflict of Interest Not Applicable.

Ethical Approval Not Applicable.

Informed Consent Not Applicable.

References

1. Aldinucci M, Baralis EM, Cardellini V, Colonnelli I, Danelutto M, Decherchi S, et al. Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis. 2023;2065–76 (**A systematic mapping study of italian research on workflows**).
2. Becciani U, Sciacca E, Costa A, Massimino P, Pistagna C, Riggi S, et al. Science gateway technologies for the astrophysics community. Concurrency and Computation Practice and Experience. 2015;27(2):306–27.
3. Sciacca E, Krokos M, Bordiu C, Brandt C, Vitello F, Bufano F, et al. Scientific visualization on the cloud: the NEANIAS services towards EOSC integration. Journal of Grid Computing. 2022;20(1):7.
4. Colonnelli I, Cantalupo B, Merelli I, Aldinucci M. StreamFlow: cross-breeding cloud with HPC. IEEE Trans Emerg Top Comput.

[20] https://youtu.be/dleV82zuabI?si=1Q2OMe7AKidVXDke

2021;9(4):1723–37. https://doi.org/10.1109/TETC.2020.3019202.
5. Santimaria ME, Colonnelli I, Cantalupo B, Torquati M, Medić D, Tuccari N, et al. Dynamic transparent streaming in file-based workflows with CAPIO. Futur Gener Comput Syst. 2025;108159. https://doi.org/10.1016/j.future.2025.108159.
6. Springel V. The cosmological simulation code GADGET-2. Mon Not R Astron Soc. 2005;364(4):1105–34. https://doi.org/10.1111/j.1365-2966.2005.09655.x. arXiv:astro-ph/0505010 [astro-ph]..
7. Viel M, Haehnelt MG, Springel V. The effect of neutrinos on the matter distribution as probed by the intergalactic medium. J Cosmol Astropart Phys. 2010;2010(6):015. https://doi.org/10.1088/1475-7516/2010/06/015. arXiv:1003.2422 [astro-ph.CO].
8. Ahrens J, Geveci B, Law C. Paraview: An end-user tool for large data visualization. Visualization Handbook. 2005;717(8):
9. Turk MJ, Smith BD, Oishi JS, Skory S, Skillman SW, Abel T, et al. yt: A multi-code analysis toolkit for astrophysical simulation data. Astrophys J Suppl Ser. 2011;192(1):9. https://doi.org/10.1088/0067-0049/192/1/9.
10. Da Silva RF, Filgueira R, Pietri I, Jiang M, Sakellariou R, Deelman E. A characterization of workflow management systems for extreme-scale applications. Futur Gener Comput Syst. 2017;75:228–38.
11. Wratten L, Wilm A, Göke J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nat Methods. 2021;18(10):1161–8. https://doi.org/10.1038/s41592-021-01254-9.
12. Diercks P, Gläser D, Lünsdorf O, Selzer M, Flemisch B, Unger JF. Evaluation of tools for describing, reproducing and reusing scientific workflows. ing.grid 1(1);2023 https://doi.org/10.48694/inggrid.3726
13. Suter F, Coleman, T.a., Altintaş, Badia, R.M., Balis, B., Chard, K., Colonnelli, I., Deelman, E., Di Tommaso, P., Fahringer, T., Goble, C., Jha, S., Katz, D.S., Köster, J., Leser, U., Mehta, K., Oliver, H., Peterson, J.-L., Pizzi, G., Pottier, L., Sirvent, R., Suchyta, E., Thain, D., Wilkinson, S.R., Wozniak, J.M., Ferreira da Silva, R. A terminology for scientific workflow systems. Futur Gener Comput Syst. 2026;174:107974. https://doi.org/10.1016/j.future.2025.107974.
14. Mulone A, Medić D, Colonnelli I, Aldinucci M. A formal framework for fault tolerance in hybrid scientific workflows. Futur Gener Comput Syst. 2025;176:108188. https://doi.org/10.1016/j.future.2025.108188.
15. Leo S, Crusoe MR, Rodríguez-Navas L, Sirvent R, Kanitz A, Geest PD, et al. Recording provenance of workflow runs with RO-Crate. PLoS ONE. 2024;19(9):1–35. https://doi.org/10.1371/journal.pone.0309210.
16. Colonnelli I, Medić D, Mulone A, Bono V, Padovani L, Aldinucci M. Introducing SWIRL: An intermediate representation language for scientific workflows. In: Formal Methods. FM 2024. Lecture Notes in Computer Science, 14933:2024;226–244. Springer, Milano, Italy https://doi.org/10.1007/978-3-031-71162-6_12.
17. The Galaxy Community. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. Nucleic Acids Res. 2022;50(W1):345–51. https://doi.org/10.1093/nar/gkac247.
18. Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch C, Sochat VV, Forster J, Lee S, Twardziok S, Kanitz A, Wilm A, Holtgrewe M, Rahmann S, Nahnsen S, Köster J. Sustainable data analysis with snakemake. F1000Research 10;2021:33 https://doi.org/10.12688/F1000RESEARCH.29032.1
19. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017;35(4):316–9. https://doi.org/10.1038/nbt.3820.
20. Huber SP, Zoupanos, S, Uhrin M, Talirz, L, Kahle L, Häuselmann R, Gresch D, Müller T, Yakutovich AV, Andersen CW, Ramirez FF, Adorf, CS, Gargiulo F, Kumbhar S, Passaro E, Johnston C, Merkys A, Cepellotti A, Mounet N, Marzari N, Kozinsky B, Pizzi G. AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. Scientific Data 7(1);2020:300 https://doi.org/10.1038/s41597-020-00638-4.
21. Deelman E, Singh G, Su M-H, Blythe J, Gil Y, Kesselman C, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Sci Program. 2005;13(3):219–37. https://doi.org/10.1155/2005/528967.
22. Liew C, Lim K-T, Becla J, Dubois-Felsmann GP, Wang D, Daniel SF, et al. Scientific data management, analysis and visualization for the lsst. IEEE Computing in Science & Engineering. 2016;18(5):50–60. https://doi.org/10.1109/MCSE.2016.85.
23. Isaila F, Latham R, Carretero J, Ross R. Scalable data staging services for data-intensive computing. Concurrency and Computation Practice and Experience. 2015;27(2):344–64. https://doi.org/10.1002/cpe.3207.
24. Zhao Y, Hwang K, Fox GC, Qiu J, Zhang M, Zhou W. Improving scientific workflow performance using cloud-aware provenance-enabled scheduling. Futur Gener Comput Syst. 2015;46:3–16. https://doi.org/10.1016/j.future.2014.01.004.
25. Cui W, Liu L, Yang X, Wang Y, Feng L, Springel V. An ideal mass assignment scheme for measuring the power spectrum with fast fourier transforms. Astrophys J. 2008;687(2):738.
26. Crusoe MR, Abeln S, Iosup A, Amstutz P, Chilton J, Tijanic N, et al. Methods included: Standardizing computational reuse and portability with the Common Workflow Language. Commun ACM. 2022;65(6):54–63. https://doi.org/10.1145/3486897.
27. Colonnelli I. Workflow models for heterogeneous distributed systems. In: CEUR WORKSHOP PROCEEDINGS, vol. 3606. CEUR-WS; 2023:1–4.
28. Colonnelli I, Casella B, Mittone G, Arfat Y, Cantalupo B, Esposito R, Martinelli AR, Medić D, Aldinucci M. Federated learning meets HPC and cloud. In: Astrophysics and Space Science Proceedings, 60;2023:193–199. Springer, Catania, Italy. https://doi.org/10.1007/978-3-031-34167-0_39.
29. Colonnelli I, Birke R, Malenza G, Mittone G, Mulone A, Galjaard J, et al. Cross-facility federated learning. Procedia Computer Science. 2024;240:3–12. https://doi.org/10.1016/j.procs.2024.07.003.
30. Taborsky P, Colonnelli I, Kurowski K, Sarma R, Pontoppidan NH, Jansík B, et al. Towards a european HPC/AI ecosystem: a community-driven report. Procedia Computer Science. 2025;255:140–9. https://doi.org/10.1016/j.procs.2025.02.269.
31. Rocco R, Rizzo S, Barbieri M, Bettonte G, Boella E, Ganz F, et al. Dynamic solutions for hybrid quantum-hpc resource allocation. In: 2025 IEEE International Conference on Quantum Computing and Engineering (QCE). Piscataway, NJ, USA: IEEE; 2025:34–40. https://doi.org/10.1109/QCE65121.2025.10289.
32. Goble C, Soiland-Reyes S, Bacall F, Owen S, Williams A, Eguinoa I, et al. Implementing FAIR digital objects in the EOSC-Life workflow collaboratory. Zenodo. 2021. https://doi.org/10.5281/zenodo.4605654.
33. Santimaria ME, Martinelli AR, Colonnelli I, Cantalupo B, Torquati M, Aldinucci M. CAPIO-CL: The CAPIO coordination language. Int J Parallel Prog. 2025;53(2): https://doi.org/10.1007/s10766-025-00789-0.
34. Santimaria ME, Filgueira R, Medić D, Colonnelli I, Aldinucci M. Overcoming dynamic I/O boundaries: a double-sided streaming methodology with dispel4py and CAPIO. In: Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC Workshops '25. New York, NY, USA: Association for Computing

Machinery; 2025:2269–80. https://doi.org/10.1145/3731599.3767577.
35. Corbett P, Feitelson D, Fineberg S, Hsu Y, Nitzberg B, Prost J-P, Snirt M, Traversat B, Wong P. In: Jain, R., Werth, J., Browne, J.C. (eds.) Overview of the MPI-IO Parallel I/O Interface, pp. 127–146. Springer, Boston, MA 1996. https://doi.org/10.1007/978-1-4613-1401-1_5.
36. Montella R, Di Luccio D, Kosta S. Dagon*: Executing direct acyclic graphs as parallel jobs on anything. In: IEEE/ACM Workflows in Support of Large-Scale Science (WORKS). 2018:64–73. https://doi.org/10.1109/WORKS.2018.00012.
37. Perrotta S, Giuseppe De Vita C, Mellone G, Edoardo Santimaria M, Salvi G, Lapegna M, Torquati M, Ciaramella A. Extending a scientific workflow engine with streaming I/O capabilities: DAGonStar and CAPIO. In: Euro-Par 2024: Parallel Processing Workshops, pp. 129–140. Springer, Cham, Switzerland 2025. https://doi.org/10.1007/978-3-031-90203-1_12.
38. Perrotta S, De Vita CG, Mellone G, Santimaria ME, Torquati M, Garcia Blas J, et al. Streaming i/o for scientific workflow engine acceleration. Futur Gener Comput Syst. 2025;107978: https://doi.org/10.1016/j.future.2025.107978.
39. Aldinucci M, Rabellino S, Pironti M, Spiga F, Viviani P, Drocco M, et al. HPC4AI: an ai-on-demand federated platform endeavour. In: Proceedings of the 15th ACM International Conference on Computing Frontiers. 2018;279–86.
40. Price DJ. Smoothed particle hydrodynamics and magnetohydrodynamics. J Comput Phys. 2012;231(3):759–94. https://doi.org/10.1016/j.jcp.2010.12.011.
41. Barnes J, Hut P. A hierarchical O(N log N) force-calculation algorithm. Nature. 1986;324(6096):446–9. https://doi.org/10.1038/324446a0.
42. Ewald PP. Die Berechnung optischer und elektrostatischer Gitterpotentiale. Ann Phys. 1921;369(3):253–87. https://doi.org/10.1002/andp.19213690304.
43. Carbone C, Petkova M, Dolag K. DEMNUni: ISW, Rees-Sciama, and weak-lensing in the presence of massive neutrinos. J Cosmol Astropart Phys. 2016;2016(7):034. https://doi.org/10.1088/1475-7516/2016/07/034. arXiv:1605.02024 [astro-ph.CO].
44. Parimbelli G, Carbone C, Bel J, Bose B, Calabrese M, Carella E, et al. DEMNUni: comparing nonlinear power spectra prescriptions in the presence of massive neutrinos and dynamical dark energy. Journal of Cosmology and Astro particle Physics. 2022;2022(11):041. https://doi.org/10.1088/1475-7516/2022/11/041. arXiv:2207.13677.
45. Hernández-Molinero B, Carbone C, Jimenez R, Peña Garay C. Cosmic Background Neutrinos Deflected by Gravity: DEMNUni Simulation Analysis. arXiv e-prints. 2023;2301–12430 (**astro-ph.CO**).
46. Verza G, Carbone C, Pisani A, Porciani C, Matarrese S. The universal multiplicity function: counting halos and voids. arXiv e-prints. 2024;2401–14451 (**astro-ph.CO**).
47. Planck Collaboration Ade PAR. Planck 2013 results. XVI. Cosmological parameters. Astronomy & Astrophysics 571;2014:16 arXiv:1303.5076 [astro-ph.CO].
48. Colonnelli I, Aldinucci M, Cantalupo B, Padovani L, Rabellino S, Spampinato C, et al. Distributed workflows with Jupyter. Futur Gener Comput Syst. 2022;128:282–98. https://doi.org/10.1016/j.future.2021.10.007.