Towards Cross-Facility Workflows Orchestration through Distributed Automation

TYLER J. SKLUZACEK, RENAN SOUZA, MARK COLETTI, FRÉDÉRIC SUTER, and RAFAEL FERREIRA DA SILVA, Oak Ridge National Laboratory, USA

Modern science relies on end-to-end workflows that incorporate experimental instruments and utilize edge, cloud, or high-performance computing and storage resources. These components are geographically dispersed across various user facilities and interconnected through high-speed networks. In this paper, we present Zambeze, an automated distributed framework designed to facilitate this new class of cross-facility workflows. Utilizing swarm intelligence principles, Zambeze orchestrates science campaigns by managing distributed autonomous agents. These agents can offer a suite of services, including computing, storage, and data management. We demonstrate the feasibility of Zambeze through a real-world application involving electron microscopy, enhanced with Artificial Intelligence capabilities.

CCS Concepts: • **Computing methodologies → Distributed computing methodologies**; *Parallel computing methodologies*; Artificial intelligence.

Additional Key Words and Phrases: workflows, distributed computing, heterogeneous computing

ACM Reference Format:
Tyler J. Skluzacek, Renan Souza, Mark Coletti, Frédéric Suter, and Rafael Ferreira da Silva. 2024. Towards Cross-Facility Workflows Orchestration through Distributed Automation. In *Practice and Experience in Advanced Research Computing (PEARC '24), July 21–25, 2024, Providence, RI, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3626203.3670606

1 INTRODUCTION

Modern science has become increasingly complex, both experimentally and computationally. It relies on end-to-end workflows that integrate experimental instruments with advanced computing and storage resources, including edge, cloud, and high-performance systems [3]. Automated orchestration of cross-facility workflows simplifies resource management, allowing practitioners to focus more on their research goals. However, the heterogeneity of research capital presents significant challenges in achieving this level of automation [5]. For example, emerging AI-coupled HPC workflows may require near real-time computing to manage large data volumes from instruments, stream them into an HPC system for model training, and then use the model to steer experiments [6]. Given the dynamic nature of distributed workflows, additional ad hoc procedures such as data transformation [8] or automated metadata extraction [7] may be necessary, typically guided by model inference. Consequently, orchestrating these workflows demands fine-grained coordinated resource management throughout their execution lifespan.

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher acknowledges the US government license to provide public access under the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Authors' address: Tyler J. Skluzacek; Renan Souza; Mark Coletti; Frédéric Suter; Rafael Ferreira da Silva, Oak Ridge National Laboratory, Oak Ridge, TN, USA.

In this paper, we introduce Zambeze, an automated distributed framework that enables automated cross-facility workflows. Leveraging swarm intelligence principles, Zambeze orchestrates science campaigns through the management of distributed autonomous agents that deliver an ensemble of services, including computing, storage, and data management. The effectiveness of Zambeze is demonstrated through its application in a real-world electron microscopy use case, augmented by Artificial Intelligence, showcasing its capability to streamline complex scientific processes.

2 A DISTRIBUTED FRAMEWORK FOR CROSS-FACILITY WORKFLOWS

Developing a system that can effectively deliver the multitude of advantages offered by flexible cross-facility workflow orchestration presents significant challenges, particularly when faced with dynamic resource constraints and a variety of heterogeneous and dispersed computing environments. This section introduces Zambeze, our pioneering automated distributed workflow orchestration system, which draws its name from Africa's fourth-largest river, renowned for its role in linking diverse biological ecosystems. The architecture of our proposed system (illustrated in Fig. 1) supports these functionalities by enabling seamless communication and data transfer between various resources, thus facilitating a robust and adaptable research environment. We next detail aspects of Zambeze.

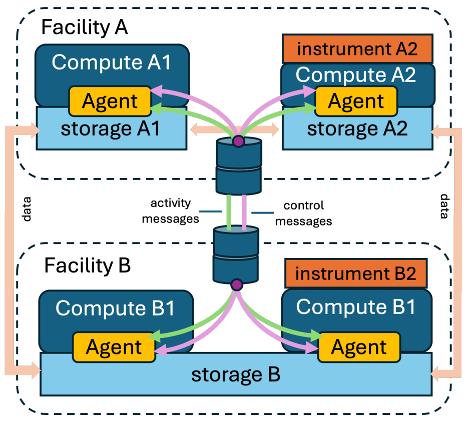

Fig. 1. **Architecture of a distributed workflow orchestration system on two facilities using Zambeze.** Each instrument and computational resource has an agent. Agents can autonomously initiate direct data movement between storage resources.

2.1 User Interface

The user interface is vital to any orchestration system, as it shapes how users manage their scientific processes. An ideal interface provides a standard, intuitive means for scientists to define orchestration at an abstract level, facilitating rapid development and improved experience. Conversely, it should also offer the flexibility to accommodate low-level configurations when needed. In our design, distributed workflows are conceptualized as a *science campaign* consisting of various *activities*. These activities can range from executing broad, complex actions like running an entire workflow to invoking simple, small-scale functions. Users can design and implement their campaigns using standard Python, enabling straightforward integration into their existing workflows. For instance, a user might set up a science campaign to train numerous machine learning (ML) models with data from two distinct resources, A and B, select the top-performing models, and then transfer these models to a repository on resource B. Zambeze simplifies the process by eliminating the need for users to specify absolute data paths, except where strictly necessary, such as locating stored datasets or determining the final destination for research artifacts. Automatically, the system identifies suitable computational resources and manages data transfers seamlessly in the background.

2.2 Compute Fabric

Once a user's science campaign initiates its activities, a mechanism is needed to dispatch and execute each activity to a suitable compute resource. To facilitate this level of processing in Zambeze, each eligible compute resource is equipped with its own autonomous *agent*. These agents connect to a distributed swarm of other independent agents, allowing them to receive task instructions, handle data transfers, capture and relay monitoring information, and ultimately execute tasks on underlying resources.

In Zambeze, agents come pre-configured with plugins that provide a standardized interface for accessing the underlying software and hardware necessary for executing activities. For example, our proof-of-concept implementation (Section 3) currently includes plugins to invoke a workflow management system (e.g., Dask and Parsl), execute shell commands, and facilitate data transfers (e.g., rsync and Globus). Our plugin system follows a modular design, allowing users to register and use custom plugins on their owned agents.

Each agent is assigned a unique address to make it accessible by every other agent in the swarm. Upon initialization, an agent automatically connects to a site-local communication server, which then links to a network of communication servers from other sites. An agent listens exclusively for activities that match the capabilities of the plugins installed on its resource. Once an activity is identified, the agent executes it and communicates the results and control information—such as execution status and file locations—to the rest of the swarm.

A significant challenge is the agent's need to execute tasks across the heterogeneous resources. To address this, our system design leverages previous advancements to abstract the complexities of machine-specific details from users. The agent's executor, which is responsible for deploying and monitoring activities on a compute resource, is built atop an internally-constructed API that interfaces with various cluster schedulers, such as Slurm, LSF, or Flux. This allows the executor, once started on a login node by the user, to autonomously manage the allocation, deallocation, and monitoring of compute nodes for any activity.

2.3 Scheduling

When a user launches a science campaign through Zambeze, the locally-running agent constructs a Directed Acyclic Graph (DAG) of the user-submitted activities, where each node is an activity and edges are dependencies between the activities. Additionally, a dedicated 'monitor' node is placed onto the front of the DAG to track and broadcast progress updates across the swarm. This storage structure allows us to leverage various optimizations, including topological sorting, which optimizes parallelization and concurrency ordering throughout the distributed system [9].

Each agent in the swarm operates using an asynchronous 'pull worker' model that only pulls activities from its communication server's queue of activities that match the available plugins properties on that specific device. Upon receiving an activity, the agent checks for any unmet data dependencies, such as an input file generated by an upstream activity. If a dependency is missing, the activity is locally queued until a broadcast control message is received that indicates that the data are available. During this awaiting period, the activity remains idle. Once the necessary data are retrieved, the activity proceeds to execute. Subsequently, the resulting status and activity output information are broadcast back to the control queue on its communication server.

2.4 Data Fabric

In Zambeze, we have adopted a lazy-transfer model where data objects (e.g., files) are only retrieved when activities are initialized on their executing agent. When a transfer is needed, the system sends a control message to its communication server, broadcasting the need for a specific data object. The appropriate agent hosting the data object responds with its available transfer information (e.g., location, authorization headers, and available transfer software). Finally, the requesting agent will identify a common transfer software from this information and ultimately execute the transfer.

We designed the data fabric to accommodate both absolute and relative object paths in the distributed system. Absolute paths are necessary when targeting specific data objects that are created outside the context of a science campaign. However, if data are generated as a result of previous workflow steps, users can refer to directories within the campaign definition. These directories include the agent's default working directory, a user-updated working directory, and the output directories of the current activity. To refer to these directories, users can employ file URIs with specific escape sequences when notating their activities in the original science campaign (e.g., '@agentdir', '@workingdir' and '@outputdir', respectively).

3 PRELIMINARY EVALUATION

In this section, we evaluate Zambeze's efficacy in facilitating user convenience throughout the various stages of a science campaign. To illustrate its practical application, we implemented Zambeze as an open source proof-of-concept framework [10] and applied it to a real-world use case involving electron microscopy. Here, we define the campaign, discuss the steps taken by users to launch the campaign, and illuminate what a user sees as the campaign executes.

3.1 Use case: Electron Microscopy and AtomAI

Electron microscopy techniques play a vital role in various scientific fields, including condensed matter physics, biology, materials science, chemistry, catalysis, and nanotechnology. The integration of AI models, such as deep convolutional neural networks, has the potential to revolutionize electron microscopy by enabling scientists to rapidly identify atomic species, defects in materials, and track their evolution over time. AtomAI [11] is an open source software suite designed specifically for this purpose. It combines instrument-specific Python libraries, deep learning capabilities, and simulation tools, simplifying the application of deep convolutional neural networks for semantic segmentation of atomic and mesoscopic images. AtomAI accurately predicts atomic species and positions in atomically-resolved imaging.

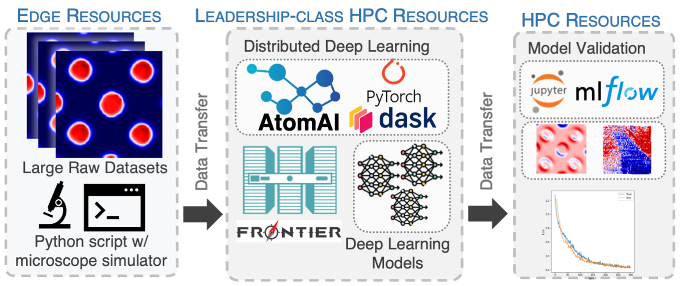

Fig. 2. **Electron microscopy use case**, where the scientists tune the next position of the microscope based on results generated by a Deep Learning model, showing a human-in-the-loop aspect.

The overall use case is depicted in Fig. 2, illustrating a distributed workflow comprising three separate environments: the Edge, a leadership-class HPC system, and a commodity cluster.

Workflow 1 (*Edge*). A Python script, simulating use of the microscope driver API, controls the electron microscope and captures raw imagery data, on the order of gigabytes.

Workflow 2 (*leadership-class HPC*). The raw image files are then transferred to a leadership-class HPC system (e.g., OLCF's Frontier). Here, users define hyperparameter ranges, such as model layers, optimizers, learning rates, number of epochs, and batch sizes, along with data sets for training and testing. The combination of tools, including Dask, AtomAI, and PyTorch, parallelizes hyperparameter search by training multiple deep convolutional neural networks for image segmentation. A subset of these models, selected based on accuracy and loss metrics, is further analyzed and validated on a commodity cluster.

Workflow 3 (*HPC*). In this workflow, the scientist performs an in-depth analysis of the models generated in Workflow 2. They create evaluation metric plots, compute additional evaluation metrics (e.g., mean squared error) against a validation dataset, and record these steps using the MLFlow data tracking API. Based on the model results, scientists can deploy the models on an actual microscope for real-time atom identification or use them to guide the selection of the next measurement point in a problem-specific parameter space (e.g., determining the next (x, y) coordinates on a sample).

In a conventional setup that relies on state-of-the-art workflow management systems and data management frameworks, the execution of the three above workflows, as well as the coordination of data transfers across facilities (e.g., experimental to computing facilities), often requires manual orchestration. This entails significant effort from users, who either have to manually manage these processes or rely on scripts to partially automate certain aspects.

3.2 Distributed Workflow Orchestration

Users simply need to install (`pip install zambeze`) and start an agent (`zambeze agent start`) on their local machine to enable communication with Zambeze's larger distributed agent swarm. They next create a science campaign in a Python script, with one activity for each of Workflows 1–3 as well as a fourth final data transfer activity, illustrated in Fig. 3.

Observe that different activities have both unique and shared parameters. Unique parameters are those specific to an activity type; for instance, a `ShellActivity` requires a “cmd" to be executed in the shell, while a `PythonActivity` requires a path to a Python script. Shared parameters are those useful to any type of activity, such as URIs to data, a human-readable name, or working/output paths.

Once users dispatch their activities to the distributed system for execution, they may poll for the status of all individual activities. In doing so, the originating agent finds and queries the agent responsible for monitoring campaign execution, which provides a stream of status events back to the originating agent, such as the the initiation and completion of activity execution, requests for data dependencies, and finally, whether the distributed workflow has successfully completed.

Note that the distributed workflow orchestration system allocates resources non-deterministically. That is, the components of a science campaign can be executed on a distinct set of machines, which enhances the flexibility and scalability of the system, accommodating varying workloads, and optimizing resource utilization.

```
# Create empty science campaign
campaign = Campaign("My Science Campaign!")

# Activity 1: Microscope simulation
act_1 = ShellActivity(
           name="create micrscope images",
           script_uri="B://mscope_sim.sh",
           cmd="./microscope_sim.sh"
           output_dir="@agentdir/images")

# Activity 2: Train machine learning models
act_2 = WFActivity(
           name="train ml models",
           wf_manager="dask",
           wf_config = ...,
           wf_uri="A://train_script.py",
           # Data from machines A and B
           data_uris=[act1.outputdir,
                      "B://file_2"],
           working_dir="@agentdir/tmp",
           output_dir="@workingdir/models")

# Activity 3: Select best-performing models
act_3 = PythonActivity(
           name="Model validation",
           data_uris=[act_2.output_dir],
           script_uri="A://mod_select.py",
           args=["-metric", "loss_2"],
           output_dir="@agentdir/models")

# Activity 4: Transfer best models
act_4 = TransferActivity(
          source=act_2.outputdir,
          dest="C://user/repository")

# Load the activities into the campaign
campaign.add_activities([act_1, act_2, act_3])

# Dispatch the campaign and poll!
campaign.dispatch()
campaign.poll(blocking=True)
```

Fig. 3. **Example campaign and activity definitions** for a simple ML distributed workflow orchestration that trains and validates ML models.

Once the client application receives a status message indicating completion of the science campaign, the user can navigate to the destination directory specified in their final transfer activity. There, they will discover the selected models along with the corresponding validation files, readily available for further analysis and evaluation.

4 RELATED WORK

Numerous tools have been developed to enhance cross-facility workflow orchestration. For instance, Globus automation services [4] provide a platform for streamlined workflow management across multiple facilities. This platform supports semi-automation of tasks, efficient resource management, and enhanced collaboration, adapting to diverse operational requirements. Our architecture extends the functionalities of Globus Flows to automate further and integrate these workflows into a cohesive framework that abstracts the complexities of underlying infrastructure. Regarding data interoperability, the Superfacility API by NERSC [2] enhances the seamless transfer of machine-readable data and services between facilities, thereby simplifying complex procedures and boosting operational efficiency.

Our earlier work [1] established a foundational framework for managing cross-facility workflows in distributed settings. This framework improves resource allocation, task scheduling, and data management, and incorporates fault tolerance mechanisms to enhance the reliability of workflows. Building on this groundwork, this paper leverages swarm intelligence principles for building an automated distributed orchestration framework that expands these capabilities.

5 CONCLUSION

Workflow execution is crucial in many scientific fields, leading to a growing library of scientific workflows and workflow management systems. Scientists are increasingly collaborating on a wide range of these workflows and research infrastructures, with the aim of achieving FAIRness in their data and workflows. This trend makes manual execution of research tasks more complex, necessitating simplified and automated methods. This paper underscores the need for institutions to adopt distributed workflow orchestration techniques. We demonstrated the benefits of such techniques through a proof-of-concept framework, Zambeze, that improved scientific throughput in a real-world electron microscopy example (AtomAI). Although these benefits are achievable, considerable research and development work in cross-facility networking and security; workflow co-scheduling on heterogeneous architectures; and human- and machine-in-the-loop workflow patterns is needed to make distributed workflow orchestration ubiquitous.

REFERENCES

[1] Katerina B. Antypas, Deborah Bard, Johannes P. Blaschke, et al. 2021. Enabling Discovery Data Science through Cross-Facility Workflows. In *2021 IEEE International Conference on Big Data*. IEEE.

[2] Deborah J. Bard, Mark R. Day, Bjoern Enders, et al. 2021. Automation for Data-Driven Research with the NERSC Superfacility API. In *High Performance Computing: ISC High Performance Digital 2021 International Workshops*. Springer.

[3] Benjamin L Brown, William L Miller, et al. 2023. *Integrated Research Infrastructure Architecture Blueprint Activity.* Technical Report. US Department of Energy (USDOE), Washington, DC (United States). Office of Science.

[4] Ryan Chard, Jim Pruyne, Kurt McKee, Josh Bryan, Brigitte Raumann, Rachana Ananthakrishnan, Kyle Chard, and Ian T. Foster. 2023. Globus Automation Services: Research Process Automation across the Space-Time Continuum. *Future Generation Computer Systems* (2023).

[5] Rafael Ferreira da Silva, Rosa M. Badia, Venkat Bala, Debbie Bard, et al. 2023. *Workflows Community Summit 2022: A Roadmap Revolution.* Technical Report. Oak Ridge National Laboratory. https://doi.org/10.5281/zenodo.7750670

[6] Shantenu Jha, Vincent R. Pascuzzi, and Matteo Turilli. 2022. AI-coupled HPC workflows. *arXiv preprint arXiv:2208.11745* (2022).

[7] Tyler J. Skluzacek. 2022. *Automated Metadata Extraction Can Make Data Swamps More Navigable.* Ph. D. Dissertation. The University of Chicago.

[8] Renan Souza, Leonardo Azevedo, et al. 2019. Efficient Runtime Capture of Multiworkflow Data Using Provenance. In *2019 15th International Conference on eScience*. IEEE, 359–368.

[9] Min-You Wu, Wei Shu, and Jun Gu. 2001. Efficient Local Search for DAG Scheduling. *IEEE Transactions on Parallel and Distributed Systems* 12, 6 (2001), 617–627.

[10] Zambeze 2024. Zambeze Framework. https://github.com/ornl/zambeze.

[11] Maxim Ziatdinov, Ayana Ghosh, et al. 2022. AtomAI Framework for Deep Learning Analysis of Image and Spectroscopy Data in Electron and Scanning Probe Microscopy. *Nature Machine Intelligence* (2022).