Strategies for Flow-Based Deployment and Orchestration in Cloud-Edge Interactive Computing

Beniamino Di Martino, Salvatore D'Angelo, Gennaro Junior Pezzullo, Antonio Esposito, Gianmarco Spinatelli, Francesco Polzella, Andrea Carollo, Giacomo Corridori

Abstract This paper is intended as a preliminary analysis for defining and orchestrating workflows in a Cloud Continuum (CC) environment. Starting from an analysis of interactive computing platforms, in particular some modified kernels of Jupyter Notebook for executing code in heterogeneous and distributed environments, it moves towards integration with a source-to-source compiler based on code decomposition and annotation for parallelization and automatic distribution through templates. Finally, integration with tools for defining and executing workflows in heterogeneous environments is discussed. Within the project and in the paper, several different approaches are presented to integrate the two techniques at different levels (from notebook to modified kernel). The aim is to facilitate the definition and execution of workflows for fine-tuning basic artificial intelligence models and large language models.

B. Di Martino
Department of Engineering, University of Campania "Luigi Vanvitelli", Italy and
Department of Computer Science and Information Engineering, Asia University, Taiwan and
Department of Computer Science, University of Vienna, Austria
e-mail: beniamino.dimartino@unicampania.it

Salvatore D'Angelo and Antonio Esposito
Department of Engineering, University of Campania "Luigi Vanvitelli", Italy e-mail: salvatore.dangelo@unicampania.it, antonio.esposito@unicampania.it

G.J. Pezzullo
Department of Engineering, University of Campania "Luigi Vanvitelli", Italy and
Department of Engineering, University of Rome "Campus Bio-Medico", Italy
e-mail: gennaro.pezzullo@unicampus.it

Gianmarco Spinatelli, Francesco Polzella, Andrea Carollo, Giacomo Corridori
Zerodivision Systems , Pisa (PI), Italy
e-mail: g.spinatelli@zerodivision.it, f.polzella@zerodivision.it, a.carollo@zerodivision.it, g.corridori@zerodivision.it

1 Introduction

The convergence of cloud computing, high performance computing (HPC) and artificial intelligence (AI) represents one of the greatest challenges and opportunities in the modern scientific and industrial computing landscape. These technologies have penetrated many sectors, ranging from the military to e-health [1], from the preservation of cultural heritage [2] to public administration and justice [3] . Ultimately, they now impact virtually every sector, driving transformation and innovation on a global scale. The rapid evolution of these technologies requires the development of methodologies, tools and frameworks to ensure smooth convergence and interoperability between these domains.

This project aims to address this challenge by developing a framework that enables integration between rapid prototyping tools, such as Jupyter Notebook, and heterogeneous distributed computing architectures.

The main goal of the project is to provide the Jupyter Notebook rapid prototyping environment with mechanisms that allow workflows defined in this environment to be deployed on multiple target platforms, following a ”code once, run anywhere” logic. The project represents a significant contribution to the scientific progress in the field of artificial intelligence and workflow systems, introducing an innovative approach that integrates interactive computing tools, rapid prototyping and workflow definition on heterogeneous architectures. This integration requires the implementation of advanced techniques for code annotation and source-to-source compilation, helping to push the boundaries of current methodologies.

The work is divided into five main phases

- State of the art analysis: reviewing existing tools, frameworks and technologies for managing and orchestrating distributed workflows, highlighting their limitations and potential opportunities.
- Definition of key objectives: clarifying design objectives, such as the integration of heterogeneous platforms, and defining success criteria based on portability, scalability and optimisation.
- Proposed methodology: describing an approach to support efficient and portable workflows through annotation and code transformation techniques.
- Strategies: introducing a Jupyter extension for environment creation and selection, a code decorator system to enable intra-cell parallelization and dynamic adaptation of execution environments, and mechanisms for resource optimisation across remote platforms.
- Conclusions: summarising the results obtained, evaluating them against the objectives set, and discussing future prospects for improving the integration of advanced technologies.

2 Background

In the field of artificial intelligence and training of more or less complex models, sequences of steps, workflows, are often defined involving data collection and preparation, model design and training, validation and optimization [4, 5]. The breadth and complexity of data and models require significant computational resources, prompting many organizations to leverage High-Performance Computing (HPC) and Cloud resources to manage these operations efficiently. In this context, HPC environments provide the computational resources needed to accelerate the training of complex AI models, primarily through clusters of GPUs to parallelize mathematically intensive operations; Cloud Computing services offer easily scalable resources specifically for machine learning and deep learning, but also serverless services to perform individual workflow tasks without having to manage complex infrastructure [6].

Over the years, numerous tools have been developed for defining, executing, and monitoring complex workflows, often including task automation and resource orchestration mechanisms. Initially designed to be generic, some tools have evolved to specifically manage workflows for use in the lifecycle of AI-based applications; others have evolved to support precise target platforms (Apache AirFlow, Kubeflow, Arvados [7], Nextflow [8], Streamflow [9]). The spread of paradigms such as microservices and Function-as-a Service architectures based on containerization technologies facilitate portability, deployment, and orchestration of tasks across different platforms (Docker, Singularity, Kubernetes), including multi-cloud, hybrid cloud, and edge environments. To address this complexity, some methodologies and tools are emerging that aim to simplify the code decomposition and distribution process, like the source-to-source compiler approach, which allows the automatic parallelization and distribution of code through templates, or the integration of code annotation techniques to define the parallelization and distribution of code in a more flexible way [10].

In the field of machine learning and deep learning, Jupyter Notebooks [11] are widely used, and are considered almost a de facto standard. A Jupyter notebook is a development environment that allows source code and text (including formatting) to be mixed together, effectively producing an interactive document that implements even complex logic. This approach makes it possible to greatly speed up the prototyping phase of a workflow for analyzing and training models, allowing models to be optimized and tested interactively, performing tests and visualizing results extremely quickly and immediately.

To bring the application expressed in a Notebook into production, however, it is necessary, once the prototyping phase is over, to proceed with exporting the Notebook into source code and refining the latter manually to make it suitable for execution on the various target platforms. Interactive computing tools such as Jupyter Notebook have also been widely integrated in the HPC domain for interactive workflow execution and have also become part of the offerings of several commercial cloud providers. There is, in addition, also work on the integration of Jupyter Notebook with workflow orchestrators, presented in [12], which strongly influenced us

in defining the goals because we consider it to be the best way to combine different worlds that year much in common: HPC, Cloud and AI.

3 Objective

As mentioned earlier in the section, 1 project's main purpose is to enrich the Jupyter Notebook rapid prototyping environment with tools that allow a workflow developed in this context to be transferred to different target platforms, following the principle "code once, run anywhere" [13]. In other words, the project seeks to make workflow nodes developed in Jupyter Notebook independent of architectural specifications, thus simplifying both prototyping and execution on HPC, Cloud, and even Edge platforms. Doing so would then allow nodes to be allocated on distributed HPC-Cloud architectures, enabling the creation of optimized workflows without requiring significant modifications or code rewrites after the initial prototyping phase. This project therefore represents a step forward in scientific programming on HPC and distributed platforms, fostering standardization and interoperability between HPC and Cloud Computing. On the other hand, the integration with Jupyter Notebook aims to involve the large user community, simplifying the adoption of the developed technologies. This will enable greater deployment and concrete impact on the life cycle management of scientific applications.

4 Methodology

After reviewing existing methodologies and identifying best practices for the integration of interactive computing and prototyping tools in the field of artificial intelligence, the project is structured in four phases:

- **State of the art analysis**: the main focus here is to study the latest advancements in interactive computing for the AI domain and what heterogeneous architectures support it.
- **Integration with interactive computing**: current workflow tools executable on heterogeneous architectures are being integrated with interactive computing tools
- **Integration of a source-to-source compiler with workflow tools**: workflow tools executable on heterogeneous architectures are then integrated with a decorator and skeleton based source-to-source compilation techniques to ensure optimal synergy between tools
- **Integration interactive computing with annotation techniques**: interactive computing tools and code annotation techniques are integrated here to integrate workflow tools executable si heterogeneous architectures

The activities in each methodological block contribute synergistically to the success of the project, ensuring a logical progression from state-of-the-art analysis to

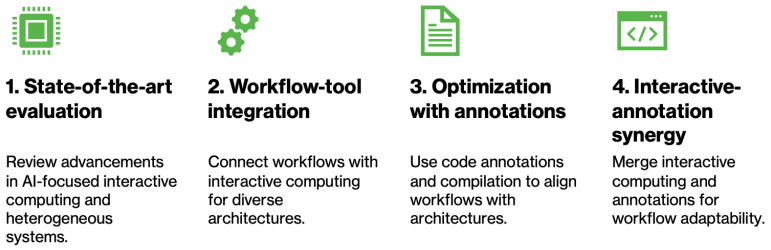

Fig. 1 Representation of the methodology used

implementation and validation of the PoC. The adoption of appropriate scientific methodologies in each phase of the project will ensure the consistency, robustness and relevance of scientific findings in the integration of artificial intelligence tools on heterogeneous architectures.

5 Strategies

Jupyter Notebook is a rapid prototyping environment widely used in the fields of data science, machine learning, and artificial intelligence. This tool enables the integration of executable code, interactive visualizations, and textual documentation within a single dynamic document, facilitating iterative workflows for data analysis, model design, and algorithm optimization. Due to its versatility and support for programming languages such as Python, R, and Julia, Jupyter Notebook has become a de facto standard among professionals such as data scientists, researchers, software engineers, and academics, who leverage it to explore complex datasets, experiment with computational solutions, and validate scientific hypotheses in a collaborative and efficient manner.

Frequently, the solutions developed and tasks described within the code executed in Jupyter Notebooks require substantial computational resources, which can only be provided by high-performance remote environments, such as HPC clusters or dedicated cloud infrastructures. However, deploying and executing code on such platforms is not a straightforward process, as it demands advanced technical skills to configure access, manage dependencies, and optimize the utilization of distributed resources.

To address these challenges, strategies have been proposed involving the development of additional modules for Jupyter Notebook capable of integrating and presenting the concept of heterogeneous computational environments to the user. A user-friendly interface has been designed to facilitate the creation and management of distributed execution environments, thereby simplifying interactions with complex infrastructures. Additionally, a decorator and skeleton based source-to-source compiler for automatic code decomposition and deployment, will be integrated to dynamically configure execution environments based on task requirements and to parallelize code execution within individual cells, significantly enhancing computational efficiency.

5.1 The Jupyter Extension

The developed component of the Jupyter Notebook extension focuses on simplifying the management of execution environments and improving usability for diverse users.

The interactive UI is designed to be straightforward and adaptable, providing a personalized experience based on the user's technical expertise. This ensures that beginners can use the system without difficulty, while more experienced users can access advanced customization options.

The UI includes the ability to select the execution environment for individual cells. This allows users to dynamically assign tasks to different computing resources, enabling heterogeneous execution environments. These environments can include both local and remote setups, allowing code to run on diverse platforms and technologies such as Docker containers, Kubernetes clusters, or over SSH connections. The

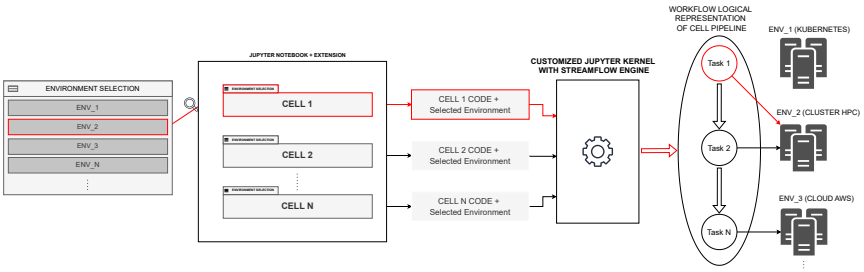

Fig. 2 Jupyter Extension for Environment Management

extension features a form for creating and customizing execution environments or templates. A key distinction is that a template serves as a blueprint, while an environment is built based on a template. Both templates and environments are represented as JSON files that define the metadata for individual notebook cells. This structure ensures consistency and flexibility in configuring computational resources.

To support collaborative work in labs or office settings, an Admin/Group/Users logic is implemented. Administrators can create templates and distribute them to specific groups or users, enabling consistent and efficient workflows within teams.

These features collectively provide a system for managing and optimizing execution environments within Jupyter Notebooks, balancing ease of use with flexibility and resource optimization.

5.2 Decorator: Cell Parallelization and Environment Tweaking

The integration of a decorator and skeleton based source-to-source compiler represents a paradigm shift in how Jupyter Notebooks handle code execution. Traditionally, a cell in Jupyter has been treated as an atomic block of code, executed sequentially in a preconfigured environment. With the integration of source-to-source compiler, cells can now be parallelized within themselves, enabling independent sections of code in a single cell to run concurrently. This approach significantly improves the execution efficiency of computationally intensive tasks.

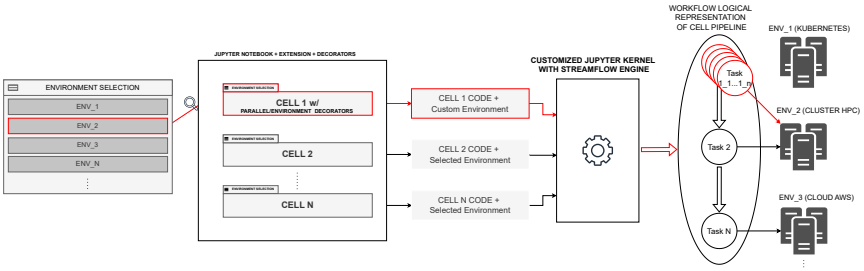

Fig. 3 Decorator for Cell Parallelization and Environment Tweaking

Another strategies that uses the decorators and skeleton based source-to-source compiler allow dynamic customization of execution environments at the cell level. A key concept introduced is the notion of a ”sub-environment/environment instance” which refers to further customization performed dynamically at runtime, based on the starting environment. For example, a decorator can generate a lightweight Docker image containing only the packages and dependencies required for the execution of a specific cell. This ensures that each cell operates in an optimized and isolated environment tailored to its specific needs.

By enabling both intra-cell parallelization and sub-environment creation, the decorator system transforms Jupyter Notebooks into a more powerful and flexible tool for managing complex workflows.

6 Conclusion and Future Works

The features and strategies developed in this work represent a significant advancement for Jupyter Notebooks, the de facto standard for AI training, creation and data-driven workflows. By introducing mechanisms such as cell-level parallelization, dynamic environment customization, and heterogeneous execution, this approach addresses critical challenges faced by labs and organizations handling large-scale calculations.

These improvements provide a new level of granularity in resource allocation and execution logic, enabling users to adapt their computational setups dynamically to the specific requirements of each task. This paradigm shift enhances both the usability and efficiency of Jupyter Notebooks, making it a more powerful tool for scientific and industrial applications.

While the current work introduces foundational improvements, there are several opportunities for future development:

- **Support for writing and executing code in multiple programming languages** within the same notebook to expand the scope of workflows.
- **Execution reporting capabilities**, providing users with detailed feedback on resource utilization and performance metrics.
- **Integration with embedded systems and FPGA programming**, in collaboration with **Università di Siena**, to extend the applicability of the platform to specialized hardware.
- **Subcell creation to enable finer-grained execution** splitting within a cell, offering even greater flexibility and efficiency.

These future directions aim to further expand the capabilities of the platform, addressing emerging needs in heterogeneous computing and workflow optimization.

Acknowledgements This work was supported by project FLUENDO subgrantee of National HPC, Big Data and Quantum Computing Center - ICSC, under the Italian NRRP MUR program funded by European Union - Next Generation EU, Mission 4 Component 1, CUP J33C22001170001.

References

1. G. J. Pezzullo, A. Esposito, and B. Di Martino, "Federated learning of predictive models from real data on diabetic patients," in *International conference on advanced information networking and applications*. Springer, 2023, pp. 80–89.
2. L. Colucci Cante, B. Di Martino, M. Graziano, D. Branco, and G. J. Pezzullo, "Automated storytelling technologies for cultural heritage," in *International Conference on Emerging Internet, Data & Web Technologies*. Springer, 2024, pp. 597–606.
3. B. Di Martino, L. C. Cante, S. D'Angelo, A. Esposito, M. Graziano, F. Marulli, P. Lupi, and A. Cataldi, "A big data pipeline and machine learning for uniform semantic representation of data and documents from it systems of the italian ministry of justice," *International Journal of Grid and High Performance Computing*, vol. 14, no. 1, 2022.

4. B. Di Martino, A. Esposito, G. J. Pezzullo, and T.-H. Weng, "Evaluating machine and deep learning techniques in predicting blood sugar levels within the e-health domain," *Connection Science*, vol. 35, no. 1, p. 2279900, 2023.
5. B. Di Martino, S. D'Angelo, A. Esposito, and P. Lupi, "Anomalous witnesses and registrations detection in the italian justice system based on big data and machine learning techniques," *Lecture Notes in Networks and Systems*, vol. 451 LNNS, p. 183 – 192, 2022.
6. R. B. Roy, T. Patel, V. Gadepally, and D. Tiwari, "Mashup: making serverless computing useful for hpc workflows via hybrid execution," in *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2022, pp. 46–60.
7. P. Amstutz, N. Tijanić, S. Soiland-Reyes, J. Kern, L. Stojanovic, T. Pierce, J. Chilton, M. Mikheev, S. Lampa, H. Ménager *et al.*, "Portable workflow and tool descriptions with the cwl," in *Bioinformatics Open Source Conference*, 2015.
8. P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature biotechnology*, vol. 35, no. 4, pp. 316–319, 2017.
9. I. Colonnelli, B. Cantalupo, I. Merelli, and M. Aldinucci, "Streamflow: cross-breeding cloud with hpc," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1723–1737, 2020.
10. C. Bandirali, S. Lodi, G. Moro, A. Pagliarani, C. Sartori, S. D'Angelo, B. Di Martino, and A. Esposito, "Parallel primitives for vendor-agnostic implementation of big data mining algorithms," vol. 2018-January, 2018, p. 396 – 401.
11. H. Shen, "Interactive notebooks: Sharing the code," *Nature*, vol. 515, no. 7525, pp. 152–152, 2014.
12. I. Colonnelli, M. Aldinucci, B. Cantalupo, L. Padovani, S. Rabellino, C. Spampinato, R. Morelli, R. Di Carlo, N. Magini, and C. Cavazzoni, "Distributed workflows with jupyter," *Future Generation Computer Systems*, vol. 128, pp. 282–298, 2022.
13. B. Di Martino, S. D'Angelo, A. Esposito, I. Martinez, J. Montero, and T. Pariente, "Parallelization and deployment of big data algorithms: The toreador approach," p. 408 – 412, 2018.