The Journal of Open Source Software

# Cost-Effective Big Data Orchestration Using Dagster: A Multi-Platform Approach

**Hernan Picatto** [1*], **Georg Heiler** [1,2*], **and Peter Klimek**[1,2,3,4*]

**1** Supply Chain Intelligence Institute Austria, Austria **2** Complexity Science Hub Vienna, Austria **3** Institute of the Science of Complex Systems, Center for Medical Data Science CeDAS, Medical University of Vienna, Austria **4** Division of Insurance Medicine, Department of Clinical Neuroscience, Karolinska Institutet, Sweden * These authors contributed equally.

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Rohit Goswami

**Reviewers:**

- @abhishektiwari
- @Midnighter

**Submitted:** 23 September 2024
**Published:** unpublished



## Summary

The rapid evolution of big data has amplified the need for robust and efficient data processing. Spark-based Platform-as-a-Service (PaaS) options, like Databricks and Amazon EMR, offer strong analytics, but at the cost of high operational expenses and vendor lock-in (Kumar & Kumar, 2022). Despite being user-friendly, their cost structures and opaque pricing can lead to inefficiencies.

This paper introduces a cost-effective, flexible orchestration framework leveraging Dagster (Dagster, 2018). Our solution reduces reliance on a single PaaS provider. It does this by integrating multiple Spark environments. We showcase Dagster's power to boost efficiency. It enforces coding best practices and reduce costs. Our implementation showed a 12% speedup over EMR. It cut costs by 40% compared to DBR, saving over 300 euros per pipeline run. This boosts productivity by permitting rapid prototyping on smaller datasets. This is key for continuous development and efficiency. It promotes a sustainable model for large-scale data processing.

## Statement of Need and Relevance

In large-scale data processing, Spark-based PaaS like Databricks are user-friendly and powerful. But, they have vendor lock-in and unpredictable costs (Zaharia et al., 2016). This convenience can lead to inefficient resource use, impacting productivity and increasing expenses.

Our solution uses Dagster's orchestration to integrate diverse Spark environments. This reduces reliance on a single provider. This mitigates lock-in risks, cuts costs, and promotes best coding practices. This boosts productivity by rapidly prototyping on smaller datasets. It cuts costs by optimizing resource use, without sacrificing performance. This approach is vital for organizations seeking agile, scalable, and cost-effective data operations.

Also, this approach ensures consistency across development stages. It helps verify and replicate results, which is critical in scientific research. The proposed framework improves reproducibility by centralizing metadata management and standardizing orchestration across diverse environments. This, in turn, reduces infrastructure complexity and aids in consistently replicating experiments, supporting reliable research. Using a tool like Dagster, researchers can create better workflows, fostering a collaborative scientific environment where methods are as open as findings.

While data pipeline research is growing, existing works focus on different aspects. For instance, authors such as (Mathew et al., 2024) concentrate on optimizing big data processing through sophisticated scheduling techniques that minimize energy consumption and latency in data centers. Their core emphasis is on the algorithmic enhancement of scheduling mechanisms, rather than on orchestration across different PaaS solutions or the promotion of coding practices

within data pipelines. Similarly, (Daw et al., 2021) explores the creation of a framework for automated resource scaling in cloud environments based on predictive analytics, aiming to optimize operational costs and performance.

In contrast to these approaches, which largely do not address the integration of multiple cloud platforms, our multi-cloud strategy leverages open orchestration tools like Dagster. This approach bridges existing gaps by deftly managing data tasks and orchestrating processing across diverse PaaS solutions. ## Architecture Model

We use Dagster, an open-source data orchestrator, in our framework. It builds, operates, and monitors data pipelines next to aligning with our cost and performance optimizations. That this pipeline can also significantly reduce resource use has been previously reported, see Heiler & Picatto (2024):

More specifically, we aimed to create a cloud-based management system offering

- Dynamic resource deployment with automatic scaling
- Virtual machine and network configuration management
- Comprehensive deployment and execution monitoring

To achieve these capabilities, several modifications to Dagster default clients were necessary.

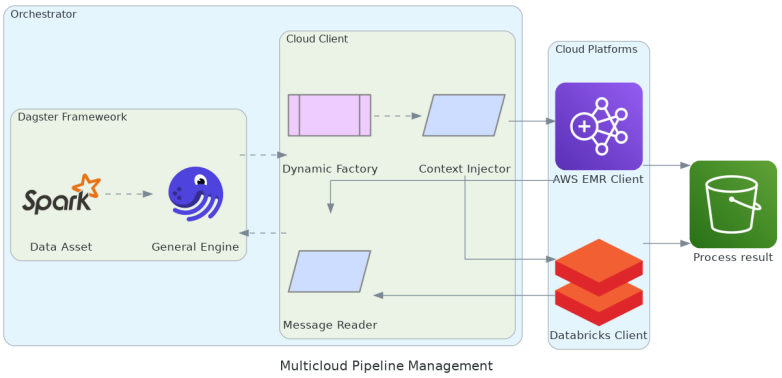


**Figure 1:** Diagram orchestrator behavior.

Our framework's core components, depicted in Figure 1, include:

1. **Dagster Context Injector:** Manages general and job-specific settings. They are vital for efficient resource use and task segmentation.
2. **Message Reader Improvements:** Boosts telemetry support. It captures and processes messages for real-time monitoring and debugging.
3. **Cloud Client Innovations:** Introduces a generic cloud client for managing Dagster on various platforms, ensuring seamless AWS integration and secure environment customization.
4. **Automation and Integration:** Automates job definition uploads with the Databricks REST API and Boto3 clients. It streamlines setup and environment bootstrapping.

5. **Dynamic Factory for Cloud Client Management:** Picks the best execution environments based on changing needs or preferences.

These changes aim at creating a user-friendly interface that shields users from the complexities of cloud resource management. This shielding significantly reduces overhead and lets organizations focus on strategic goals. To minimize inconsistencies and configuration issues, we further dockerized the implementation to ensure a controlled development and production environment, facilitating reliability and replicability in production.

### Example Use Case: Mining web-based interfirm networks from Common Crawl

We show our framework by making a web-based map of company ecosystems, as Kinne & Axenbeck (2020). The research aim in such work is to find relationships between companies. To this end, company websites are searched for hyperlinks to other company websites, often revealing collaborative innovation efforts.

Datasets

- Common Crawl CC-MAIN: This dataset comprises WARC (Web ARChive) files containing raw web crawl data, and WAT files storing computed metadata.
- Seed Nodes: A subset of URLs (e.g., langing pages of company websites) identified as starting points for our analysis. These nodes are processed to ensure they are relevant and free of common problems.

Pipeline Breakdown Existing data extraction methods only work on text or graph data. However, to understand which kind of collaborations companies are forming, our use case requires the extraction of both text and graph data simultaneously. We therefore developed a custom data extraction method as follows. Our pipeline consists of four key assets:

1. **NodesOnly**: Extracts and preprocesses seed node information.
2. **Edges**: Extracts HTML content and hyperlinks from seed node URLs
3. **Graph**: Constructs a hyperlink graph by combining nodes and edges
4. **GraphAggr**: Aggregates the graph to the domain level for broader analysis

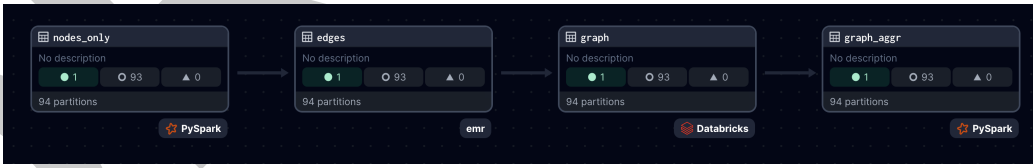


**Figure 2:** Detailed dagster pipeline showcasing how execution environments can be chosen as needed between local, EMR and DBR.

Figure 2 hows assets that prove our framework's adaptability and efficiency. The framework can handle diverse computing needs across various platforms. Data partitioning occurs along two dimensions: time and domain. The temporal partitioning matches the Common Craw[1] dataset. It streamlines data management and access. Domain-based partitioning, on the other hand, enables parallel processing of different research queries. This approach allows varied filtering in data analysis. It optimizes resources and enables task submission to the best platforms.

### Further Details

For detailed information on the implementation challenges encountered during the development of our framework, please refer to Appendix 1.

For a comprehensive comparison of the platforms used in our study, please refer to Appendix 2.

[1] Common Crawl was accessed between October 2023 and March 2024 from Common Crawl.

## Acknowledgments

This research was supported by Supply Chain Intelligence Institute Austria (ASCII).

## References

Dagster. (2018). Dagster | cloud-native orchestration of data pipelines. In *GitHub repository*. GitHub. https://github.com/dagster-io/dagster

Daw, N., Bellur, U., & Kulkarni, P. (2021). Speedo: Fast dispatch and orchestration of serverless workflows. *Proceedings of the ACM Symposium on Cloud Computing*, 585–599. https://doi.org/10.1145/3472883.3486982

Heiler, G., & Picatto, H. (2024). *Cost efficient alternative to databricks lock-in*. https://georgheiler.com/2024/06/21/cost-efficient-alternative-to-databricks-lock-in/

Kinne, J., & Axenbeck, J. (2020). Web mining for innovation ecosystem mapping: A framework and a large-scale pilot study. *Scientometrics*, *125*(3), 2011–2041. https://doi.org/10.1007/s11192-020-03726-9

Kumar, P., & Kumar, P. (2022). Vendor lock-in situation and threats in cloud computing. *International Journal of Innovative Science and Research Technology*, *7*(9), 1437–1441. https://doi.org/10.5281/zenodo.7196590

Mathew, A., Andrikopoulos, V., Blaauw, F. J., & Karastoyanova, D. (2024). Pattern-based serverless data processing pipelines for function-as-a-service orchestration systems. *Future Generation Computer Systems*, *154*, 87–100. https://doi.org/10.1016/j.future.2023.12.026

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache spark: A unified engine for big data processing. *Commun. ACM*, *59*(11), 56–65. https://doi.org/10.1145/2934664