




# 1 Cost-Effective Big Data Orchestration Using Dagster: 2 A Multi-Platform Approach

3 **Hernan Picatto** <sup>1\*</sup>, **Georg Heiler** <sup>1,2\*</sup>, and **Peter Klimek**<sup>1,2,3,4\*</sup>

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

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 23 September 2024

Published: unpublished

## License

Authors of papers retain copyright  
and release the work under a  
Creative Commons Attribution 4.0  
International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## 8 Summary

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

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

## 21 Statement of Need and Relevance

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

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

30 Also, this approach ensures consistency across development stages. It helps verify and replicate  
31 results, which is critical in scientific research. Using a tool like Dagster, researchers can create  
32 better workflows. It will foster a collaborative scientific environment. Their methods will be as  
33 open as their findings.

34 While data pipeline research is growing, existing works focus on different aspects. Anil et  
35 al. ([Mathew et al., 2024](#)) emphasize optimizing big data processing. Use energy-efficient  
36 scheduling to reduce consumption and latency in data centers. Daw et al. ([Daw et al., 2021](#))  
37 explore using predictive analytics to automate resource scaling in cloud environments. This  
38 aims to optimize cost and performance. Our multi-cloud strategy leverages open orchestration  
39 tools like Dagster. This approach bridges existing gaps, deftly managing data tasks across  
40 diverse PaaS.

41 **Relevance**

42 The proposed framework improves reproducibility by centralizing metadata management and  
 43 standardizing orchestration across diverse environments. This in turn reduces infrastructure  
 44 complexity and aids in consistently replicating experiments, supporting reliable research.  
 45 Notwithstanding the mounting interest in data pipelines, authors such as Mathew et al. (2024)  
 46 concentrate on the optimisation of big data processing through sophisticated scheduling  
 47 techniques that minimise energy consumption and latency. While their work also aims to  
 48 optimise resource utilisation in data centres, its core emphasis is on the algorithmic enhancement  
 49 of scheduling mechanisms, rather than on orchestration across different PaaS solutions or on  
 50 the promotion of coding practices within data pipelines. In their 2021 paper, Daw et al. (2021)  
 51 examine the creation of a framework for automated scaling of resources in cloud environments.  
 52 Their work focuses on aspects of resource allocation based on predictive analytics, with the goal  
 53 of optimising operational costs and performance. In contrast to the work presented here, these  
 54 approaches do not address the integration of multiple cloud platforms or the orchestration of  
 55 data processing tasks using open tools.

56 **Architecture Model**

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

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

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

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

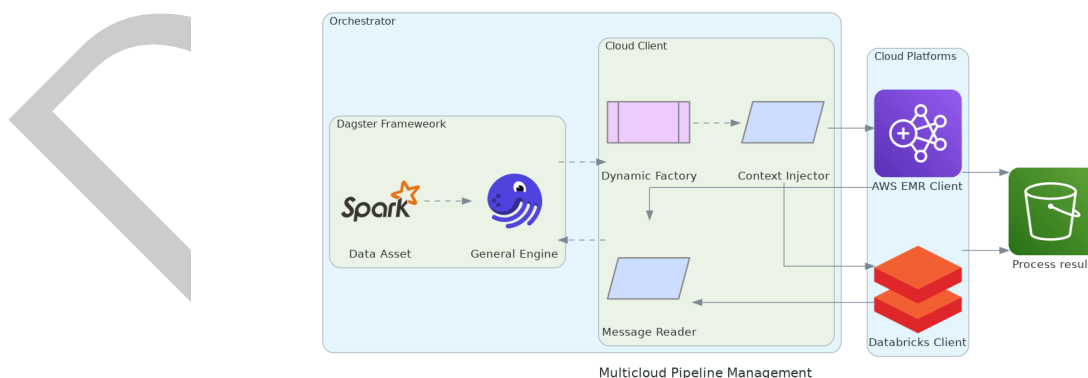


Figure 1: Diagram orchestrator behavior.

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

- 67 1. **Dagster Context Injector:** It manages general and job-specific settings. They are vital  
68 for efficient resource use and task segmentation.
- 69 2. **Message Reader Improvements:** It boosts telemetry support. It captures and processes  
70 messages for real-time monitoring and debugging.
- 71 3. **Cloud Client Innovations:** Introduces a generic cloud client for managing Dagster on  
72 various platforms, ensuring seamless AWS integration and secure environment customiza-  
73 tion.
- 74 4. **Automation and Integration:** Automates job definition uploads with the Databricks  
75 REST API and Boto3 clients. It streamlines setup and environment bootstrapping.
- 76 5. **Dynamic Factory for Cloud Client Management:** It picks the best execution environments  
77 based on changing needs or preferences.

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

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

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

### 88 Datasets

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

### 94 Pipeline Breakdown

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

- 99 1. **NodesOnly:** Extracts and preprocesses seed node information.
- 100 2. **Edges:** Extracts HTML content and hyperlinks from seed node URLs
- 101 3. **Graph:** Constructs a hyperlink graph by combining nodes and edges
- 102 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.

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

#### 109 Further Details

110 For detailed information on the implementation challenges encountered during the development  
111 of our framework, please refer to [Appendix 1](#).

112 For a comprehensive comparison of the platforms used in our study, please refer to [Appendix 2](#).

#### 113 Acknowledgments

114 This research was supported by [Supply Chain Intelligence Institute Austria \(ASCI\)](#).

#### 115 References

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

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

121 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/>

122 Kinne, J., & Axenbeck, J. (2020). Web mining for innovation ecosystem mapping: A framework  
123 and a large-scale pilot study. *Scientometrics*, 125(3), 2011–2041.

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

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

130 Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen,  
131 J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I.  
132 (2016). Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11),  
133 56–65. <https://doi.org/10.1145/2934664>

---

<sup>1</sup>Common Crawl was accessed between October 2023 and March 2024 from [Common Crawl](#).