A Survey of Big Data Pipeline Orchestration Tools from the Perspective of the DataCloud Project*

Mihhail Matskin[1], Shirin Tahmasebi[1], Amirhossein Layegh[1], Amir H. Payberah[1], Aleena Thomas[2], Nikolay Nikolov[2], and Dumitru Roman[2]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{misha,shirint,amlk,payberah}@kth.se
[2] SINTEF AS, Oslo, Norway
{*firstname.lastname*}@sintef.no

Abstract. This paper presents a survey of existing tools for Big Data pipeline orchestration based on a comparative framework developed in the DataCloud project. We propose criteria for evaluating the tools to support reusability, flexible pipeline communication modes, and separation of concerns in Big Data pipeline descriptions. This survey aims to identify research and technological gaps and to recommend approaches for filling them. Further work in the DataCloud project is oriented towards the design, implementation, and practical evaluation of the recommended approaches.

Keywords: Big Data pipeline · Orchestration tools · Reusability.

1 Introduction

The availability of massive amounts of data has tremendously changed the data collection process and analysis over the last few years. The concept of Big Data and the supporting solutions have allowed dealing with potentially unlimited heterogeneous data in different formats within a practically acceptable time. However, the growth of data has increased both opportunities and challenges. In terms of opportunities, data processing is being heavily invested in to empower the decision-making process of organizations in possession of Big Data. Furthermore, the data analytics process is becoming complex due to the characteristics of Big Data, the sophisticated tools and technologies involved, different interests among stakeholders, often changing business needs, and the lack of a standardized process for the lifecycle of *Big Data pipelines* [22].

Because of the complexity of Big Data analysis tasks, the software supporting such analysis requires a combination of a broad spectrum of trusted software components. Such a combination involves integrating components into pipelines that take care of the pipeline execution and data transfer. The design and usage of Big Data pipelines increase the efficiency of data analysis, while at the same time require support for designing and managing the pipelines. Many organizations recognize the significance of Big Data pipelines, however there are still critical challenges in their implementation, such as the heterogeneity of involved stakeholders and limited knowledge reuse [6]. In this paper, we refer to the tools that support the description and execution of Big Data pipelines as *Big Data pipeline orchestration tools*. Although there exist many orchestration tools for Big Data pipelines, they have not focused on crucial areas such as *reusability* and *separation of concerns*.

*This work is partly funded by the EC H2020 project "DataCloud: Enabling The Big Data Pipeline Lifecycle on the Computing Continuum" (Grant nr. 101016835).

The problem of combining different components into an executable process is not new. *Workflow systems* are systems that support the integration of steps of a semi- or fully automated procedure into a manageable process. *Business workflows* are workflow systems oriented towards automating business processes [6]. *Scientific workflows* are other types of workflow systems oriented towards automation of scientific experiments [5]. Recently, *Big Data workflows* are becoming prevalent and refer to modeling processes containing various Big Data analytic or processing steps [25]. The main characteristics of Big Data workflows are the dynamics and heterogeneity of data sources and processing components, which typically require different orchestration models. Big Data pipelines are special cases of Big Data workflows where workflows are more oriented towards the end-users. However, since there is no clear boundary between Big Data workflows and Big Data pipelines, in this work we do not make an explicit difference between them.

The approach in this paper is based on (1) extracting requirements for Big Data pipelines from the business cases of the DataCloud project as well as existing scientific literature and software tools around Big Date pipelines, and (2) analyzing which of the existing solutions (if any) can satisfy the identified requirements. For the requirements extraction, we defined the following Research Questions (RQ), which guided our work:

- RQ1. How to bridge the technological gap between different experts involved in the Big Data pipeline design, implementation, and management?
- RQ2. How to support the reuse of previously developed knowledge and solutions in designing and implementing Big Data pipelines?
- RQ3. How to support debugging of Big Data pipelines?

In this paper, we survey existing Big Data pipeline orchestration tools, identify research and technological gaps, and suggest approaches to filling the gaps. This work is done in the context of the HORIZON 2020 project DataCloud[3]. The rest of the paper is organized as follows. Section 2 provides a brief overview of the DataCloud project, including its objectives and expected results. Section 3 identifies the requirements and the classifiers for building a comparison table of the existing tools relevant to the DataCloud project perspective, and includes the comparison tables to identify gaps in the current solutions. The final Section 4 summarizes the main findings and refers to ongoing work in the DataCloud project to solve the identified gaps.

2 DataCloud Project Perspective

In this section, we provide a brief overview the DataCloud project and present some requirements for Big Data pipeline orchestration tools identified while working on the project.

2.1 The DataCloud project

The DataCloud project, which runs between 2021 and 2023, aims to develop a novel paradigm for Big Data processing over heterogeneous resources, including the Cloud/Edge/Fog Computing Continuum. The core concept of the project is Big Data pipelines, whose complete lifecycle is supported by several processing capabilities. The DataCloud project utilizes this paradigm to solve issues for a broad set of business cases coming from Small and Medium-sized Enterprises (SMEs) and large organizations with difficulties in capitalizing on Big Data due to the lack of technical expertise and suitable processing capabilities.

[3] `https://datacloudproject.eu`

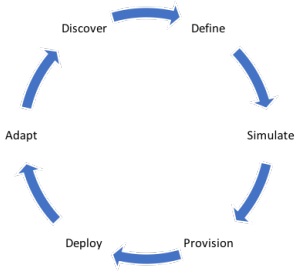

Fig. 1. Big Data pipeline lifecycle.

In the DataCloud project, we develop a set of new languages, methods, infrastructures, and software prototypes for discovering, simulating, deploying, and adapting Big Data pipelines on heterogeneous and untrusted resources. The project underlines the separation of concerns and separates the design from the run-time aspects of their deployment. This separation allows domain experts without significant technical/programming knowledge to participate in the definition and management of Big Data pipelines. Moreover, the DataCloud solutions allow the incorporation of Big Data pipelines in organizations' business processes more efficiently and make them more accessible to a broader set of stakeholders regardless of the hardware infrastructure. DataCloud assumes a typical Big Data pipeline lifecycle involving a set of high-level processing steps executed in a loop (see Figure 1). The project aims to deliver its solutions to support the Big Data pipelines lifecycle as a set of interoperable tools that form the *DataCloud toolbox*. The toolbox includes the following components (see Figure 2):

- *DIS-PIPE*: Provides integration of process mining techniques and Artificial Intelligence (AI) algorithms to learn the structure of Big Data pipelines. The learning is based on extracting, processing, and interpreting huge amounts of event data collected from heterogeneous data sources.
- *DEF-PIPE*: Provides support for the visual design and description of Big Data pipelines based on a Domain Specific Language (DSL). The tool includes the means to store and load the pipeline definitions that enable the reuse of previously developed solutions. It also enables data experts and domain experts to define the content by configuring individual steps and injecting code or customizing generic predefined step templates.
- *SIM-PIPE*: Simulates the enactment of Big Data pipelines and provides pipelines testing functionality, including a sandbox for evaluating individual pipeline step performance. Furthermore, SIM-PIPE provides a simulator to analytically predict the performance of the overall Big Data pipelines across the Computing Continuum resources.
- *R-MARKET*: Deploys a decentralized backbone resource network based on a hybrid permissioned and permissionless blockchain. This component provides a marketplace for resources and enables transparent provisioning of resources that increase the overall trust.
- *DEP-PIPE*: Enables elastic and scalable deployment of Big Data pipelines with real-time event detection and automated decision making.
- *ADA-PIPE*: Provides a data-aware algorithm for intelligent and adaptive provisioning of resources and services across the Computing Continuum as well as intelligent resource reconfiguration.

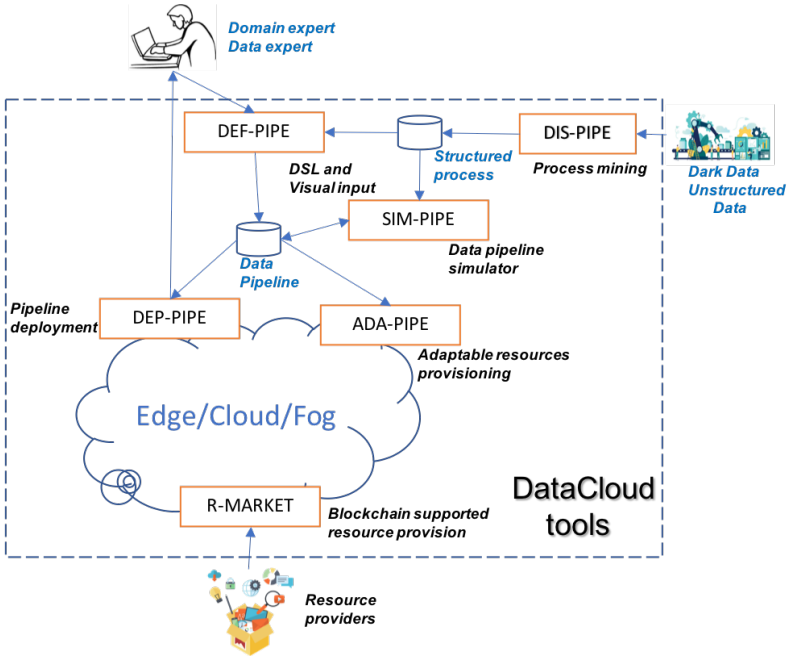

Fig. 2. DataCloud Tools.

We evaluate the DataCloud solutions on five business cases provided by the DataCloud business partners, which cover a broad spectrum of Big Data pipeline applications:

- Smart mobile marketing campaigns
- Automatic live sports content annotation
- Digital health system
- Predicting deformations in ceramics
- Analytics of manufacturing assets

2.2 Identifying Requirements

The diversity and complexity of modeling data, processing Big Data pipelines, and the heterogeneity of Computing Continuum platforms require a multidisciplinary effort using expert knowledge of the domain, data, and technical knowledge of the computational environment. However, the collaboration among domain, data, and technical experts requires repeated communication cycles introducing significant overhead and barriers to success. Therefore, it is crucial and challenging to provide tools to bridge the technological and knowledge gaps between all relevant experts and enable them to collaborate while keeping the separation of concerns. Nevertheless, there are many design solutions for Big Data pipelines, so reusing them can boost the design and the development of new pipelines. The recent development of containers (as a technology allowing efficient and reliable installation of software components on different platforms) provides a good foundation for implementing reusable solutions. This is why containerization is a promising approach to support reusable solutions.

In designing Big Data pipelines, we need to consider the computational resources and decide about the deployment of the pipelines. However, it is not easy to make such a decision in the design phase in many cases, and running pipelines in the production computational environment might be expensive. Therefore, to make such a process more efficient, we need a simulation and debugging environment for pipelines that operates without deploying the pipeline. This is why combining pipeline descriptions with simulation tools is an essential requirement for modern systems.

By summarizing the above aspects and taking into account the properties of the DataCloud project described in Section 2.1, we outline the following requirements for Big Data pipeline orchestration tools:

- Req1: Provide separation of concerns between design and run-time aspects.
- Req2: Provide convenient means for describing pipelines including visual and textual (DSL-based) interfaces.
- Req3: Support reusability of the previously developed steps and pipelines in designing new pipelines.
- Req4: Provide flexible data transfer between steps in pipelines.
- Req5: Support containerization for nodes and pipeline descriptions.
- Req6: Provide smooth integration of description and simulation of components.

3 Overview of Existing Solutions

In this section, we describe a set of criteria that we consider important for Big Data pipeline orchestration tools and compare current solutions with respect to those criteria. Most of the criteria refer to the requirements identified in Section 2.2.

3.1 Criteria for Comparison

The traditional way of representing pipelines is DSL that is incorporated into an orchestration environment. We performed an analysis of existing tools based on the following classifiers reflecting the requirements identified in the previous section.

Type of Workflow/Pipeline. We consider three types of workflows/pipelines: *business workflows*, *scientific workflows*, and *Big Data workflows* (defined in Section 1). Each tool falls into one of these three types according to its main applications.

Workflow/Pipeline Model. These categories are not mutually exclusive. Different possible workflow models can be categorized as follows:

- *Script-based*: In these type of workflows the composition of nodes is described using scripting languages. These workflows are useful for expert users to design complex applications more flexibly and concisely [33].
- *Event-based*: These workflows are characterized by a discrete set of states. A transition from one state to another happens on the occurrence of events emitted asynchronously or by an external trigger. Hence, in event-based workflows, users define event rules to declare under what circumstances state transitions should occur. This provides a responsive orchestration process [6, 50].

- *Adaptive*: This model allows designing adaptive or context-aware workflows to consider the run-time situations and exceptions, such as a failure in pre-processing an input file. Such workflows can respond to dynamic environmental needs effectively [1, 6, 49, 58].
- *Declarative*: In a declarative approach, a minimal set of requirements, which are often expressed by a set of constraints, is defined. Therefore, the execution of the workflow is allowed until this set of constraints is satisfied. This is advantageous for increasing flexibility and is especially useful in highly unpredictable contexts, in which there are a large number of allowed and possible alternatives [7, 13, 14].
- *Procedural*: This workflow model explicitly specifies the sequence of steps and tasks known as control-flow. Thus, during the execution of the model, it is possible to execute a process only as explicitly specified in the control-flow [14, 44].

Separation of Concerns. This criterion is related to the Req1 requirement. If a tool has a mechanism for the separation of high-level workflow definition concerns from step-specific implementation and deployment details, it supports separation of concerns for stakeholders (the value of the classifier is *Yes* in the tables in the rest of the paper). Otherwise, it is concluded that separation of concerns is not a focus [15] (the value of the classifier is *No*).

Type of Language. This criterion is related to the Req2 requirement. We consider the following types of languages:

- *General-Purpose Language (GPL)*: GPL is a highly applicable language (e.g., Python, R, Java) across a variety of application domains.
- *Domain-Specific Language (DSL)*: DSL is a language that is specially designed for a specific problem domain. The main advantage of using a DSL is that domain experts, who have little knowledge outside of their domain, can efficiently design relevant parts of the pipeline logic. On the other hand, its main drawbacks are the limited portability across different environments and low applicability outside the discrete domain.

Input Supported. This criterion is related to the Req2 requirement. Here, we consider two possible input types for designing and implementing Big Data pipelines: *text-based* input and *graphical* or *visual* input. We assume that tools with visual input types may also support text-based input types.

Ease of Use. This criterion is related to the Req2 requirement. The possible values for this classifier are *Hard*, *Medium*, and *Easy*, which refer to the level of expertise a user needs to have to be able to use the tool. For example, if a tool allows designing a workflow through a clear graphical interface, the tool is considered to be Easy to use. However, if the tool relies on using a lightweight, human-readable text format, such as YAML, XML, and JSON, the level of ease-of-use is considered Medium. Otherwise, if programming knowledge and skills are needed for using the tool, it is concluded that the tool is Hard to use. Sometimes it is difficult to place a tool into one only category and we allow a combination of values, for example, Medium/Hard or Easy/Medium.

Focus on Reusability. This criterion is related to the Req3 requirement. Reusability refers to the characteristic of a designed workflow, whereby it can be used to create another similar workflow. The reusability is a tool's focus if (i) it provides a visual drag-and-drop feature for reusing the previously-designed workflows, or (ii) it supports a search for previously-designed similar solutions to be imported as text-based workflows for reusing in designing the current one. Otherwise, it is concluded that reusability is not a focus [15]. For the former group of tools, the value of the classifier is *Yes*, and the value of the classifier for the latter is *No*.

Reusable elements. This criterion is related to the Req3 requirement. In this research, the reusability of each tool is evaluated in three aspects:

- Whether or not it is possible to reuse the definition of the entire workflow.
- Whether or not it is possible to reuse the definition of each step.
- Whether or not it is possible to reuse the step implementation.

The possible values for each of these aspects are *Yes*, *No*, and *Partial*, which means that although no specific way is designed to share assets, sharing can be done by manually copying scripts.

Nested Step Definition This criterion is related to the Req3 requirement. This is a binary classifier, the value of which would be *Yes*, if the tool has a specific way for defining and using a step inside and as a part of another more granular step. Otherwise, its value would be *No*.

Configurable Data Transmission Medium Definition. This criterion is related to the Req4 requirement. This classifier has two possible values; *Yes* and *No*, depending on whether or not the tool allows users to choose between multiple data transmission mediums (such as shared file system, web services, RPC, file transfer, HTTP, and FTP) for transferring data between steps during the pipeline execution.

Configurable Communication Medium Definition. This criterion is related to the Req4 requirement. This classifier has two possible values; *Yes* and *No*, depending on whether or not the tool supports choosing the medium of control flow communication between steps during the pipeline definition, invocation, and execution. For example, this could be done using RESTful API, message queues, or RPC.

Containerization. This criterion is related to the Req5 requirement. In different workflow types, containers can be used to automate the deployment process, enabling better scalability. We consider three possible approaches for containerization [15]:

- *Workflow-level*: In this approach, the entire workflow is wrapped inside a container allowing scalability of the full workflow.
- *Step-level*: In this approach, each step is encapsulated and wrapped inside a container allowing scalability of individual steps.
- *Encapsulation*: Here, a Big Data pipeline framework/tool is available as a container image for installation and local or distributed usage.

Integrates a Simulation Tool. This criterion is related to the Req6 requirement. Here, we consider if a tool integrates a simulation tool. The value of this classifier is either *Yes* or *No*.

Monitoring. If the tool provides means for monitoring the execution of the pipeline. Depending on how the execution can be monitored, it can be further classified into the following three subcategories:

- *Runtime*: If the monitoring is available real-time.
- *Logging*: If the run logs are available at the end of the execution.
- *No*: If the tool offers no monitoring support.

3.2 Tools Comparison

In Tables 1-4, we compare a representative number of the most popular Big Data pipeline orchestration tools with respect to the classifiers described in the previous subsection. In these tables ✓ denotes *Yes* and ✕ denotes *No*.

Table 1. Classification Summary

Tools	Workflow	Model	Type of Language	Input	Focus on Reusability	Ease of Use	Reusable			Integrates a SIM Tool	Configurable Data Transmission Medium	Configurable Communication Medium	Separation of Concerns	Mapping to Containers			Monitoring	Nested Step Definition
							Entire Workflow	Step Definition	Step Implementation					Step Level	Entire Workflow	Encapsulation		
Apache Airflow [30, 3]	Big Data	Script-based Event-based Adaptive Procedural	GPL	Text[4]	×	Hard	✓	✓	✓	×	✓	×	✓	✓	×	✓	✓	✓
Argo Workflow [4]	Big Data	Script-based Declarative Procedural	DSL	Text	×	Hard	✓	✓	✓	×	✓	×	✓	✓	×	✓	✓	✓
Pachyderm [40, 42]	Big Data	Script-based Adaptive	DSL	Text	×	Hard	✓	×	Partial	×	×	×	✓	×	✓	✓	Runtime	×
Kepler [59]	Scientific	Procedural	DSL	Visual	×	Easy	✓	✓	✓	×	×	×	✓	×	✓	✓	Runtime	✓
Trifacta [56]	Scientific	Procedural	DSL	Visual	×	Easy	✓	✓	✓	×	×	×	✓	✓	✓	×	Runtime	✓
StreamPipes [54]	Big Data	Script-based Event-based Adaptive Procedural	DSL[5]	Text[6]	×	Easy Medium	✓	✓	✓	×	×	×	✓	✓	✓	×	Runtime	×
MakeFlow [32]	Scientific	Script-based Declarative	DSL	Text	×	Hard	✓	×	✓	×	✓	✓	×	✓	×	×	Runtime	✓
CGAT-Core [16]	Scientific	Script-based Adaptive	GPL	Text	×	Hard	✓	×	Partial	×	×	×	×	×	×	×	Runtime	✓
Snakemake [28, 52]	Big Data	Script-based Adaptive Declarative	DSL	Text	×	Hard	✓	×	✓	×	✓	×	✓	✓	✓	✓	Runtime	✓
Pegasus [12, 43]	Scientific	Script-based	DSL	Text	×	Hard	✓	✓	✓	×	✓	×	×	✓	✓	×	Runtime	✓

[4]There exist visual and graphical tools for working with workflows, but not for creating and defining workflows.

[5]Despite the fact that Java programming language is used for defining data sources, data sinks, and data processors, since they are all converted to RDF notation, and also since the graphical tool is used for defining the whole control flow, the language is DSL, not GPL.

[6]Data sources, data sinks, and data processors are implemented in Java programming language. However, for creating the whole workflow, there exists a GUI interface.

Table 2. Classification Summary

Tools	Workflow	Model	Type of Language	Input	Focus on Reusability	Ease of Use	Reusable			Integrates a SIM Tool	Configurable Data Transmission Medium	Configurable Communication Medium	Separation of Concerns	Mapping to Containers			Monitoring	Nested Step Definition
							Entire Workflow	Step Definition	Step Implementation					Step Level	Entire Workflow	Encapsulation		
NextFlow [35]	Scientific	Script-based Adaptive Declarative	DSL	Text	✗	Medium	✓	✓	✓	✗	✓	✗	✗	✓	✓	✗	Runtime	✓
KubeFlow [29]	Big Data	Script-based Adaptive Procedural	GPL	Text	✗	Hard	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗
Toil [55]	Big Data	Script-based Adaptive Procedural	GPL	Text	✗	Hard	✓	✗	✓	✗	✓	✗	✓	✗	✗	✓	Runtime	✓
BioDepot Workflow Builder [8]	Scientific	Script-based Procedural	DSL	Visual	✗	Easy	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	Runtime	✗
Hyperloom [23]	Big Data	Script-based Adaptive Procedural	GPL	Text	✗	Hard	Partial	Partial	Partial	✗	✓	✗	✓	✗	✗	✓	Logging	✓
MachineFlow [31]	Scientific	Procedural	GPL	Text	✗	Medium	✓	✓	✓	✗	✗	✗	✗	Partial	✗	✗	Logging	✗
SoS Workflows [57]	Scientific	Script-based Adaptive Procedural	DSL[7]	Text	✗	Medium	✓	✗	Partial	✗	✗	✗	✗	Partial	✓	✓	Runtime	✓
Dray [17]	Scientific	Script-based Procedural	GPL	Text	✗	Medium	Partial	✗	✓	✗	✓	✗	✓	✓	✗	✓	Runtime	✗
Flyte [18]	Scientific	Script-based Adaptive Procedural	GPL	Text	✗	Medium	✓	✗	✓	✗	✗	✗	✗	✓	✓	✓	Runtime	✓
Galaxy [20]	Scientific	Procedural	DSL	Visual	✓	Easy	✓	✗	✓	✗	✗	✗	✓	✓	✗	✓	Runtime	✓

[7]The DSL provides domain-specific syntax and is built on top of Python 3.
[8]Provides explicit support for external automation tools to deploy experiments.

Table 3. Classification Summary

Tools	Workflow	Model	Type of Language	Input	Focus on Reusability	Ease of Use	Reusable			Integrates a SIM Tool	Configurable Data Transmission Medium	Configurable Communication Medium	Separation of Concerns	Mapping to Containers			Monitoring	Nested Step Definition
							Entire Workflow	Step Definition	Step Implementation					Step Level	Entire Workflow	Encapsulation		
Popper [24]	Scientific	Script-based Procedural Declarative[8]	DSL	Text	×	Easy-Medium	✓	×	✓	×	×	×	✓	✓	×	×	×	×
StreamFlow [53]	Scientific	Script-based Procedural	DSL	Text	×	Medium-Hard	Partial	×	Partial	×	×	×	✓	✓	×	✓	×	×
BPipe [10]	Scientific	Script-based	DSL	Text	×	Medium Easy	✓	✓	✓	×	×	×	✓	✓	✓	×	Logging	✓
YAWL [19]	Business	Adaptive Declarative	DSL	Visual	×	Easy	×	×	×	×	×	×	×	×	×	×	×	×
Apache Oozie [41]	Big Data	Script-based Event-based Procedural	DSL	Text	×	Hard	Partial	Partial	Partial	×	×	×	×	×	×	✓	Runtime	×
KNIME [27]	Big Data	Script-based Adaptive Declarative Procedural	DSL	Visual	×	Easy	×	✓	✓	×	×	×	✓	×	×	✓	Runtime	×
Google Workflow [21]	Business	Script-based Adaptive Declarative Procedural	DSL	Text	×	Medium	×	✓	✓	×	✓	×	×	×	×	✓	Logging	✓
Keboola [26]	Big Data	Script-based Event-based Procedural	DSL	Visual	×	Easy	×	✓	✓	×	✓	×	✓	×	✓	✓	Runtime	✓
Accio [2]	Scientific	Script-based	DSL	Text[4]	×	Medium	✓	×	×	×	×	×	×	×	×	×	Runtime	×
Node-RED [38, 39]	Big Data	Script-based Event-based Adaptive Declarative Procedural	DSL	Visual	×	Easy	×	×	✓	×	×	×	✓	×	×	✓	Logging	✓

Table 4. Classification Summary

Tools	Workflow	Model	Type of Language	Input	Focus on Reusability	Ease of Use	Reusable			Integrates a SIM Tool	Configurable Data Transmission Medium	Configurable Communication Medium	Separation of Concerns	Mapping to Containers			Monitoring	Nested Step Definition
							Entire Workflow	Step Definition	Step Implementation					Step Level	Entire Workflow	Encapsulation		
Skitter [48, 51]	Big Data	Script-based Adaptive	DSL	Text	✗	Hard	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
Dagster [11]	Big Data	Script-based Event-based Adaptive Procedural	GPL	Text[4]	✗	Hard	Partial	✓	✓	✗	✓	✗	✓	✗	✓	✓	Logging	✓
Prefect [45, 46]	Big Data	Script-based Event-based Adaptive Declarative Procedural	GPL	Text[4]	✗	Hard	Partial	✓	✓	✗	✓	✗	✓	✗	✓	✓	Runtime	✓
Apache NiFi [36]	Big Data	Declarative Procedural	DSL	Visual	✗	Easy	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	Runtime	✓
Conductor [34]	Big Data	Script-based Event-based Adaptive Declarative	DSL	Text[4]	✗	Medium	Partial	Partial	Partial	✗	✓	✗	✗	✗	✓	✓	Logging	✗
Reflow [47]	Scientific	Procedural	DSL	Text	✗	Hard	Partial	✓	✓	✗	✗	✗	✗	✓	✓	✗	Logging	✗
BMC Control-M [9]	Business	Script-based Event-based Adaptive Declarative Procedural	DSL	Visual	✗	Easy	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗	Realtime	✓

4 Conclusions

In this paper we investigated existing Big Data pipeline orchestration tools. By analysing them (in Section 3.2) we show that important requirements defined in the DataCloud project are not (or are only partially) satisfied by the currently available tools. In particular, only a few tools support a graphical input language for the description of pipelines. While several tools allow some levels of reusability, most reusability aspects (such as support for searching available suitable solutions) are not implemented. Moreover, integration with other tools (including simulation tools) is not supported by many of them. However, we can mention that Apache Airflow, Argo Workflow, and Snakemake are the closest tools to our requirements among the reviewed ones. Nevertheless, although some aspects of some requirements are considered in some tools, no tool supports all required aspects.

To address these problems in the DataCloud project, we are developing a DEF-PIPE component (see Section 2.1) for designing Big Data pipelines. This component will provide means for the description and manipulation of pipelines and the environment. It will also support a complete graphical and textual interface, accumulation and reuse of solutions across different applications, flexible integration with a simulation tool, and separation of concerns between the description of design-time and run-time aspects. Preliminary results in this direction are reported in [37].

References

1. van der Aalst, W.M., Basten, T., Verbeek, H., Verkoulen, P.A., Voorhoeve, M.: Adaptive workflow: On the interplay between flexibility and support. In: Enterprise Inf. Systems, pp. 63–70. Springer (2000)
2. Accio: Accio - Workflow Authoring. `https://privamov.github.io/accio/docs/creating-workflows.html`, [Online; accessed 01-April-2021]
3. Airflow, A.: Tutorial. `https://airflow.apache.org/docs/apache-airflow/stable/tutorial`, [Online; accessed 25-March-2021]
4. Argo: Argo Workflow - Docs. `https://argoproj.github.io/argo-workflows/`, [Online; accessed 21-April-2021]
5. Barga, G., Taylor, D.: Workflows for e-Science, chap. Scientific versus Business Workflows, pp. 9–16. Springer (2007)
6. Barika, M., Garg, S., Zomaya, A.Y., Wang, L., Moorsel, A.V., Ranjan, R.: Orchestrating big data analysis workflows in the cloud: research challenges, survey, and future directions. ACM Computing Surveys (CSUR) **52**(5), 1–41 (2019)
7. Bernardi, M.L., Cimitile, M., Di Lucca, G., Maggi, F.M.: Using declarative workflow languages to develop process-centric web applications. In: 2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops. pp. 56–65. IEEE (2012)
8. BioDepot: BioDepot-Workflow-Builder - General Information. `https://bwb.readthedocs.io/en/latest/#general-information`, [Online; accessed 01-April-2021]
9. BMC: BMC Control-M - Docker Image with Embedded Agent. `https://docs.bmc.com/docs/display/public/ctmapitutorials/Manage+workload+in+Docker+Containers`, [Online; accessed 26-March-2021]
10. BPipe: BPipe - Pipeline Even. `http://docs.bpipe.org/Guides/PipelineEvents/`, [Online; accessed 02-April-2021]
11. Dagster: Dagster - Concepts. `https://docs.dagster.io/concepts`, [Online; accessed 26-March-2021]
12. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Da Silva, R.F., Livny, M., et al.: Pegasus, a workflow management system for science automation. Future Generation Computer Systems **46**, 17–35 (2015)
13. Demeyer, R., Van Assche, M., Langevine, L., Vanhoof, W.: Declarative workflows to efficiently manage flexible and advanced business processes. In: Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming. pp. 209–218 (2010)

14. van Der Aalst, W.M., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science-Research and Development **23**(2), 99–113 (2009)
15. Dessalk, Y.D., Nikolov, N., Matskin, M., Soylu, A., Roman, D.: Scalable execution of big data workflows using software containers. In: Proceedings of the 12th International Conference on Management of Digital EcoSystems. pp. 76–83 (2020)
16. Developers, C.: CGAT-core documentation. `https://cgat-core.readthedocs.io/en/latest/`, [Online; accessed 21-April-2021]
17. Dray: Dray Overview. `https://github.com/CenturyLinkLabs/dray` (2015), [Online; accessed 17-March-2021]
18. Flyte: Flyte Documentation. `https://docs.flyte.org/en/latest/index.html` (2020), [Online; accessed 17-March-2021]
19. Foundation, Y.: YAWL - Yet Another Workflow Language. `http://www.yawlfoundation.org/documents/YAWL_leaflet-final.pdf` (2007), [Online; accessed 19-March-2021]
20. Galaxy: Galaxy Tutorials. `https://galaxyproject.org/learn/` (2005), [Online; accessed 17-March-2021]
21. Google: GoogleWorkflow - Error Handling Syntax. `https://cloud.google.com/workflows/docs`, [Online; accessed 31-March-2021]
22. Henning Baars, J.E.: From data warehouses to analytical atoms - the internet of things as a centrifugal force in business intelligence and analytics. In: wenty-Fourth European Conference on Information Systems. ECIS (2016)
23. Hyperloom: Hyperloom - Basic Terms. `https://loom-it4i.readthedocs.io/en/latest/intro.html#basicterms`, [Online; accessed 01-April-2021]
24. Jimenez, I., Sevilla, M., Watkins, N., Maltzahn, C., Lofstead, J., Mohror, K., Arpaci-Dusseau, A., Arpaci-Dusseau, R.: The popper convention: Making reproducible systems evaluation practical. In: 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 1561–1570 (2017). https://doi.org/10.1109/IPDPSW.2017.157
25. Kashlev, A., Lu, S., Mohan, A.: Big data workflows: A reference architecture and the dataview system. Services Transactions on Big Data (STBD) **4**(1), 1–19 (2017)
26. Keboola: Keboola - Overview. `https://help.keboola.com/overview/`, [Online; accessed 02-April-2021]
27. Knime: KNIME - Extensions. `.https://docs.knime.com/2019-06/analytics_platform_quickstart_guide/index.html#extend-knime-analytics-platform.`, [Online; accessed 31-March-2021]
28. Köster, J., Rahmann, S.: Snakemake—a scalable bioinformatics workflow engine. Bioinformatics **28**(19), 2520–2522 (2012)
29. Kubeflow: Kubeflow - Pipelin. `https://www.kubeflow.org/docs/components/pipelines/overview/pipelines-overview/#what-is-a-pipeline`, [Online; accessed 01-April-2021]
30. Lal Chattaraj, J., Villamariona, J.: Apache Airflow Tutorial – DAGs, Tasks, Operators, Sensors, Hooks & XCom. `https://www.qubole.com/tech-blog/apache-airflow-tutorial-dags-tasks-operators-sensors-hooks-xcom/`, [Online; accessed 25-March-2021]
31. MachineFlow: MachineFlow Overview. `https://github.com/sean-mcclure/machine_flow` (2018), [Online; accessed 17-March-2021]
32. MakeFlow: MakeFlow - Overview. `https://cctools.readthedocs.io/en/latest/makeflow/#overview`, [Online; accessed 02-April-2021]
33. Marozzo, F., Talia, D., Trunfio, P.: Js4cloud: script-based workflow programming for scalable data analysis on cloud platforms. Concurrency and Computation: Practice and Experience **27**(17), 5214–5237 (2015)
34. Netflix: Conductor - Configuration. `https://netflix.github.io/conductor/configuration`, [Online; accessed 25-March-2021]
35. Nextflow: NextFlow - Completion Handler. `https://www.nextflow.io/docs/latest/metadata.html#completion-handler`, [Online; accessed 25-March-2021]

36. NiFi, A.: Apache NiFi - Expression Language Guide. https://nifi.apache.org/docs/nifi-docs/html/expression-language-guide.html, [Online; accessed 26-March-2021]
37. Nikolov, N., Dessalk, Y.D., Khan, A.Q., Soylu, A., Matskin, M., Payberah, A.H., Roman, D.: Conceptualization and scalable execution of big data workflows using domain-specific languages and software containers. Internet of Things p. 100440 (2021). https://doi.org/https://doi.org/10.1016/j.iot.2021.100440, https://www.sciencedirect.com/science/article/pii/S2542660521000834
38. Node-RED: Node-RED - Flow Control. https://cookbook.nodered.org/, [Online; accessed 01-April-2021]
39. Node-RED: Node-RED - Subflow. https://nodered.org/docs/creating-nodes/, [Online; accessed 01-April-2021]
40. Novella, J.A., Emami Khoonsari, P., Herman, S., Whitenack, D., Capuccini, M., Burman, J., Kultima, K., Spjuth, O.: Container-based bioinformatics with pachyderm. Bioinformatics **35**(5), 839–846 (2019)
41. Oozie, A.: Oozie, A. Apache Oozie - Coordinator Job. .https://oozie.apache.org/docs/5.2.1/CoordinatorFunctionalSpec.html#a1._Coordinator_Overview, [Online; accessed 26-March-2021]
42. Pachyderm: Pachyderm - Introduction. https://docs.pachyderm.com/latest/how-tos/create-pipeline/, [Online; accessed 25-March-2021]
43. Pegasus: Pegasus - Documentation. https://pegasus.isi.edu/documentation, [Online; accessed 02-April-2021]
44. Pesic, M., Schonenberg, H., van der Aalst, W.: Declarative workflow. In: Modern Business Process Automation, pp. 175–201. Springer (2010)
45. Prefect: Prefect - Core Concepts. https://docs.prefect.io/core/concepts/, [Online; accessed 25-March-2021]
46. Prefect: Prefect - Latest API. https://docs.prefect.io/api/latest/, [Online; accessed 25-March-2021]
47. Reflow: Reflow - Overview. https://github.com/grailbio/reflow, [Online; accessed 25-March-2021]
48. Saey, M.: A dsl for distributed, reactive workflows (2018)
49. Seiger, R., Huber, S., Schlegel, T.: Toward an execution system for self-healing workflows in cyber-physical systems. Software & Systems Modeling **17**(2), 551–572 (2018)
50. Semeniuta, O., Falkman, P.: Epypes: a framework for building event-driven data processing pipelines. PeerJ Computer Science **5**, e176 (2019)
51. Skitter: Skitter - Documents. https://soft.vub.ac.be/~mathsaey/skitter/docs/latest/readme.html, [Online; accessed 26-March-2021]
52. Snakemake: Snakemake - Tutorial. https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html#snakemake-tutorial, [Online; accessed 26-March-2021]
53. StreamFlow: StreamFlow GitHub. https://github.com/alpha-unito/streamflow (2020), [Online; accessed 17-March-2021]
54. Streampipe, A.: StreamPipes - Tutoria. https://streampipes.apache.org/docs/docs/dev-guide-tutorialsources/, [Online; accessed 01-April-2021]
55. Toil: Toil - Workflows with Multiple Job. https://toil.readthedocs.io/en/latest/developingWorkflows/developing.html#workflows-with-multiple-jobs, [Online; accessed 01-April-2021]
56. Trifacta: Trifacta - Function. https://docs.trifacta.com/display/SS/Wrangle+Language#WrangleLanguage-Functions.1 (2013), [Online; accessed 26-March-2021]
57. Wang, G., Peng, B.: Script of scripts: A pragmatic workflow system for daily computational research. PLoS Computational Biology **15** (2019)
58. Wieland, M., Schwarz, H., Breitenbücher, U., Leymann, F.: Towards situation-aware adaptive workflows: Sitopt—a general purpose situation-aware workflow management system. In: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). pp. 32–37. IEEE (2015)
59. Wikipedia: Kepler - HierarchicalWorkflow. https://en.wikipedia.org/wiki/Kepler_scientific_workflow_system#Hierarchical_workflows (2015), [Online; accessed 01-April-2021]