A Knowledge-Based Approach for Structuring Cyclic Workflows

Rafael Brandão[1], Vitor Lourenço[1], Marcelo Machado[1], Leonardo Azevedo[1], Marcelo Cardoso[1], Renan Souza[1], Guilherme Lima[1], Renato Cerqueira[1], Marcio Moreno[1]

[1] IBM Research, Rio de Janeiro, RJ, Brazil

Abstract. This paper showcases the Cycle Orchestrator, a microservices infrastructure designed to structure and manage workflows related to heterogeneous data, through a knowledge-based perspective. It aims at leveraging reasoning, explainability and collaboration among users over experiments that comprise workflow executions. We briefly discuss about design and implementation aspects to support the lifecycle of workflows (*i.e.*, modeling, configuration, execution, provenance tracking and querying), exploring a holistic representation called Hyperknowledge that is amenable to be consumed and reasoned upon.

Keywords: Knowledge-based Workflow Orchestration, Workflow Management Systems, Hyperknowledge Representation.

1 Introduction

Processing massive amounts of data through different techniques while enabling stakeholders to collaborate and consume experiments' results is a crosscutting challenge tackled by many industries and technical domains. In the natural resources domain, particularly in the oil and gas (O&G) industry, a motivating use case is seismic data interpretation, which is key in exploration processes that analyze geological structures in the subsurface. Experts, supported by specialized tools and domain knowledge, identify patterns and correlate geological factors by exploring different data sources. This practice aims at detecting geological structures, enhancing information, correcting potential inconsistencies in the data acquisition process, and so on. An increasing number of works in the literature have been proposed to apply Machine Learning (ML) workflows to support aspects of such processing. To systematically model complex data processing pipelines, such as the ML workflows of this motivating use case, while promoting collaboration and knowledge curation, a holistic perspective is required.

In this sense, we conceptualized and developed the Cycle Orchestrator, a knowledge-based workflow management system (WfMS) to support and operationalize the whole lifecycle of ML and general-purpose workflows. Including specification, setup, execution and provenance data management of such workflows. In the motivating use case, it was conceived primarily to support O&G exploration use cases that apply cyclic ML workflows. That is, streams of ML tasks that can yield improved results through a chain of execution iterations. These workflows are associated to particular types of data sources, *e.g.* pre-stack and post-stack seismic data. The initially considered use cases comprised unsupervised ML pipelines that train new models and reuse pre-trained models and weights against new datasets for improving the quality by cyclic evolution. In this context, orchestration involves the definition of what model and version should be applied to analyze specific data sources in exploration processes.

2 Knowledge-based Workflow Modeling

The Cycle Orchestrator draws on the Hyperknowledge [2] conceptual model for relating knowledge specifications aligned through a domain ontology to segments of multimodal content. In this model, the relation between nodes is done through links and connectors. Nodes' fragments can be referenced to with the anchor mechanism, making the structuring of data segments explicit. In addition, the use of nested contexts as a structuring mechanism promotes the overall organization of the knowledge base, allowing the clustering of information through different perspectives or dimensions.

The modeling process considers different moments of the workflow lifecycle, namely specification, setup, execution and analysis. In the workflow specification, the tasks are defined with their input data types, expected output data types and execution ordering. Then, in the setup step references to actual data are connected to the tasks' specifications. After these definitions, the execution stage may take place, followed by a fourth moment where users inspect workflows' output, querying and reasoning over experiments' results.

The specification and configuration of workflows rely on the Hyperknowledge Specification Language (HSL), a JSON-based definition scheme. Following the principles of a glue language, it does not constrain any modeling aspects, nor define supported data formats or specific content structuring. The language allows any modeling design that makes sense for interested parts (users and systems) who produce and consume information from the knowledge base. Assuming of course there is a previously agreed ontology with well-defined terms and relations. The Cycle Orchestrator's basic ontology defines handy entities and connectors for specifying and configuring tasks, execution strategies (sequential, cyclic, parallel, map-reduce like), ordering and other aspects.

To illustrate modeling decisions, consider the following "hello world" example to model a basic Python function that takes two integer numbers passed as parameters *factor_a* and *factor_b* and writes the resulting multiplication value in a text file. In the specification step, the basics of the task are specified, *i.e.* its input data types (two numbers) and output data types (a file). No execution ordering is needed, since we have a single task. For each input parameter identifier of the executable Python function, it should be an anchor in the media node with the same identifier.

Listing 1 depicts the HSL modeling with the specification for this executable node's input and output to entities representing their compatible data types. Afterwards, Listing 2 shows the setup of the task. Following the same rationale, it connects the entities representing actual data sources, numbers (10 and 25 as input parameters) and file (*mult_result.txt* as output data) to the task specification.

```
["context", "BasicOntology", {"subconceptOf":"Ontology"}, [
        ["concept", "Number", {"subconceptOf": "Attribute"}],
        ["concept", "SystemPath", {"subconceptOf": "Attribute"}]]
    ],
],
["node", "MultiplicationTask", {"instanceOf": "DataTransformation", "mimeType": "applica-
tion/wf_task", "uri": "multiply.py"}, [
            ["property", {"package": "my_package", "function": "multiply"}],
            ["anchor", "factor_a", {"type": "data", "description": "Parameter A to be multiplied"}],
            ["anchor", "factor_b", {"type": "data", "description": "Parameter B to be multiplied"}],
],
["link", null, {"connector": "consumes"}, [
    ["bind", {"task": "MultiplicationTask#factor_a"}],
    ["bind", {"data": "Number"}]]
],
["link", null, {"connector": "consumes"}, [
    ["bind", {"task": "MultiplicationTask#factor_b"}],
    ["bind", {"data": "Number"}]]
],
["link", null, {"connector": "produces"}, [
    ["bind", {"task": "MultiplicationTask"}],
    ["bind", {"data": "SystemPath"}]]
]
```

Listing 1. HSL example for the specification of a multiplication task.

```
["node", "number_10", {"instanceOf": "Number", "value": 10}, []],
["node", "number_25", {"instanceOf": "Number", "value": 25}, []],
["node", "result_file", {"instanceOf": "SystemPath", "src": "./mult_result.txt"}, []],

["link", null, {"connector": "consumes"}, [
    ["bind", {"task": "MultiplicationTask#factor_a"}],
    ["bind", {"data": " number_10"}]]
],
["link", null, {"connector": "consumes"}, [
    ["bind", {"task": "MultiplicationTask#factor_b"}],
    ["bind", {"data": " number_25"}]]
],
["link", null, {"connector": "produces"}, [
    ["bind", {"task": "MultiplicationTask"}],
    ["bind", {"data": " result_file"}]]
]
```

Listing. 2. HSL example for the setup of a multiplication task.

3 System implementation

Information in the Cycle Orchestrator is represented in the Hyperknowledge Base, a hybrid storage solution that uses a direct hyperlinked knowledge graph to maintain all information about workflow execution plans and provenance data stored in the knowledge base. The proposed modeling adheres to the MLWfM ontology [1] to structure basic aspects of ML and the PROV-ML [4] as provenance data model.

Users interact with the system through a REST API and a web UI for curating and querying information, named Knowledge Explorer System (KES) [3]. The REST API has three endpoint sets for workflow specification, execution, and lineage retrieval. The specification endpoints provide basic operations for workflow plans. HSL files with workflow definitions are parsed, producing both a Hyperknowledge representation and a directed acyclic graph (DAG) data structure. The Execution endpoint interfaces with the execution engine's API (Apache Airflow[1]) to maintain and command workflows. The execution handler captures provenance data, structuring according to the provenance data model that can be queried through the Lineage endpoint. Figure 1 shows the architectural overview of the system.

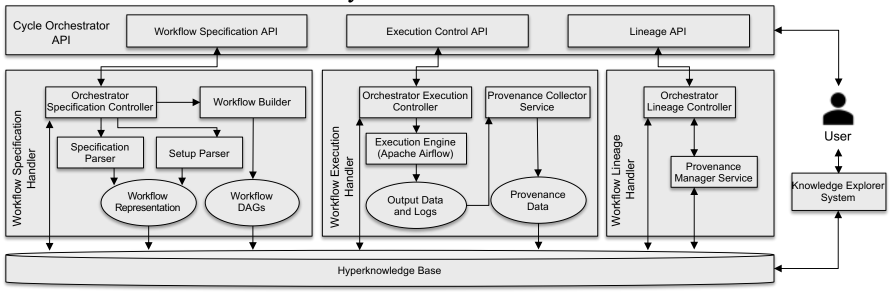
Cycle Orchestrator API
Workflow Specification API
Execution Control API
Lineage API
Workflow Specification Handler
Orchestrator Specification Controller
Workflow Builder
Specification Parser
Setup Parser
Workflow Representation
Workflow DAGs
Workflow Execution Handler
Orchestrator Execution Controller
Provenance Collector Service
Execution Engine (Apache Airflow)
Output Data and Logs
Provenance Data
Workflow Lineage Handler
Orchestrator Lineage Controller
Provenance Manager Service
User
Knowledge Explorer System
Hyperknowledge Base

Fig. 1. Cycle Orchestrator's infrastructure overview.

4 Demo

The main goal of this demo is to showcase the Cycle Orchestrator's approach in action, presenting how workflows modeled with HSL definitions can be posted to the provided API with a command-line tool, and inspected through the KES dashboard UI. For that end, we explore a simple example for a parallel matrix multiplication workflow.
Short video: https://ibm.box.com/s/byg4wb6beo632f0d68tqg5f5jh0o8k4f
Long video: https://ibm.box.com/s/g3kupho1pn3gv3zupl2xcf1samqdfs98

References

1. Moreno, M. et al.: Managing Machine Learning Workflow Components. In: 14th IEEE Conference on Semantic Computing, ICSC. pp. 25–30 (2020).
2. Moreno, M.F. et al.: Extending Hypermedia Conceptual Models to Support Hyperknowledge Specifications. Int. J. Semantic Computing. 11, 01, 43–64 (2017).
3. Moreno, M.F. et al.: KES: The Knowledge Explorer System. In: 2018 International Semantic Web Conference (P&D/Industry/BlueSky), ISWC. (2018).
4. Souza, R. et al.: Provenance Data in the Machine Learning Lifecycle in Computational Science and Engineering. In: 2019 IEEE/ACM Workflows in Support of Large-Scale Science, WORKS. pp. 1–10 (2019).

[1] https://airflow.apache.org/