Robust and Adaptive Multi-Engine Analytics using IReS

Nikolaos Papailiou[1], Katerina Doka[1], Victor Giannakouris[1], Vassilis Papaioannou[1], Dimitrios Tsoumakos[2], and Nectarios Koziris[1]

[1] Computing Systems Laboratory, National Technical University of Athens, Greece
[2] Ionian University, Greece

Abstract. The complexity of Big Data analytics has long outreached the capabilities of current platforms, which fail to efficiently cope with the data and task heterogeneity of modern workflows due to their adhesion to a single data and/or compute model. As a remedy, we demonstrate *IReS*, the *Intelligent Resource Scheduler* for complex analytics workflows executed over multi-engine environments. *IReS* is able to optimize a workflow with respect to a user-defined policy by (a) allocating distinct parts of it to the most advantageous execution and/or storage engine among the available ones and (b) deciding on the exact amount of resources provisioned. Moreover, IReS can efficiently adapt to the current cluster/engine conditions and recover from failures by effectively monitoring the workflow execution in real-time. During the demo, the attendees will be able to create, optimize and execute workflows that match real use cases over multiple compute and data engines, imposing their preferred optimization objectives. Moreover, the audience will have the chance to confirm the resilience and adaptability of the platform in cases of failing nodes, unavailable engines and surges in load.

Key words: Multi-Engine Optimization, Cost Modeling, Big Data, Analytics Workflows, Fault Tolerance

1 Introduction

Big data analytics have become a key basis of business competition, underpinning new trends on market analysis, productivity growth, innovation and consumer surplus [6]. The need for near-real-time, data-driven analytics has given rise to diverse execution engines and data stores that target specific data and computation types (e.g., [1, 3, 5, 2, 4]). However, in this highly complex and constantly changing landscape, analytic workflows may include (a) multiple data types (e.g., relational, key-value, graph, etc.) generated from different sources and stored on different engines [9] and (b) diverse operators, ranging from simple Select-Project-Join (SPJ) and data movement to complex NLP-, graph- or custom business-related operations. Finally, analysts need to execute such workflows under varying constraints and policies (e.g., optimize performance or cost, require different fault-tolerance degrees, etc.). There currently exists no single engine that can optimize for this complexity, thus multi-engine analytics have been proposed as a promising solution [12].

To address multi-engine analytics workflow optimization, we have designed and implemented the *Intelligent Multi-Engine Resource Scheduler (IReS)*[8], an integrated, open source platform for optimizing, planning and executing complex analytics workflows[1]. Its goal is to provide adaptive, cost-based and customizable resource management of the diverse execution and storage engines available. By incorporating a modeling framework that constantly evaluates the cost and performance of data and computational resources, IReS is able to decide on the most advantageous store, indexing and execution pattern. The core elements upon which the platform bases its operation are:

- A profiling and modeling engine that benchmarks operator performance and cost for different engine configurations. Operator performance metrics are collected from actual executions of the operator, triggered explicitly or via automated profiling experiments. The trained models are stored and utilized for the optimization of workflows.
- An extensible tree-based metadata framework that describes operators in abstract and materialized forms, enabling the automatic matching of operators that perform similar tasks. Workflows can be described in an abstract way, leaving the task of finding all alternative execution paths to IReS.
- A decision-making process that selects the most prominent workflow execution plan, consulting the cost and performance models of the various operators.
- An execution layer that enforces and monitors the selected multi-engine execution plan. Our executor is implemented on top of YARN [13], allowing for fine grained resource allocation control and fault tolerance.

In this work, IReS has been enhanced with new features that advance the functionality of the platform, adding adaptability, elasticity and fault tolerance. The most prominent newly added features are:

Online model refinement: Upon execution of a workflow, the currently monitored execution metrics provide feedback to the existing models in order to refine them and capture possible changes in the underlying infrastructure (e.g., hardware upgrades) or temporal degradations (e.g., due to unbalanced use of engines, collocation of competing tasks, surges in load etc.). Evaluations have shown that this module provides adaptability in an effective manner, ameliorating the accuracy of the models after only a few runs.

Optimal resource provisioning: Apart from deciding on the specific implementation/engine of each workflow operator, the optimizer of IReS provisions the correct amount of resources to execute the workflow conforming as much as possible to the user-defined criteria. This operation relies on genetic algorithms that supply resource-related parameters (e.g., #cores, memory) from the local minima of the trained models.

Fault tolerance: This feature relies on the periodic execution of customizable and parameterized health scripts that report on the status of cluster nodes and check if all services (i.e., engines and datastores) needed for the execution of an operator are up and running. This information is used to plan according to cur-

[1] `https://github.com/project-asap/IReS-Platform`

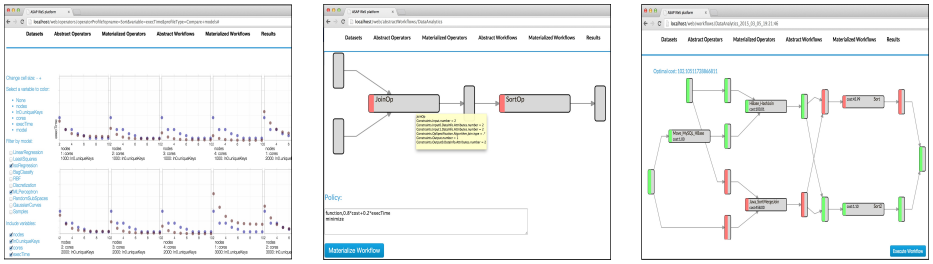

Fig. 1. IReS web application GUI. a)Operator models, b)Abstract Workflows, c)Materialized Workflows

rently available engines and resources, detect failures during workflow execution and re-plan the remaining part of it.

The resulting optimization is orthogonal to (and in fact enhanced by) any optimization effort within a single engine. Unlike [11], IReS is a fully open-source platform that targets both low (e.g., join, sort, etc.) as well as high level (e.g., machine learning, graph processing) operators. Recent works [10, 7] in the field of multi-engine workflow execution focus more on the translation of scripts from one engine to another, being thus tied to specific programming languages and engines. Contrarily, our system is engine agnostic, treating operators as black boxes. This allows for extensibility to new engines and easy addition of new operators regardless of their implementation language.

2 Demonstration Description

Our system is controlled by a comprehensive web-based GUI that attendees can utilize. The basic interaction dimensions include input parametrization, operator model visualization, execution plan inspection and execution output evaluation. The GUI controls a cloud-based deployment of several runtime engines and data stores over 16 virtual machines of an Openstack cluster hosted in our lab. The users will have the opportunity to test IReS either using one of the predefined workflows, driven by real business use cases, or assembling their own, using operators from the ASAP operator library. A diverse set of operations of varying complexity and execution parameters is covered, including basic SQL queries (selections, projections, joins), ML algorithms (classification and clustering) as well as custom business-related tasks. Our demonstration of IReS will showcase the following functionalities.

Modeling of operators: To achieve extensibility, each operator is profiled and modeled as a black box with an input space consisting of parameters that affect its performance, e.g., amount of resources, dataset size, configuration parameters etc., and an output space containing all performance/cost metrics that need to be approximated, e.g., execution time, memory consumption, etc. The attendees will be able to visualize the various trained models and experience how they are refined in real-time, by actual workflow executions (Figure 1).

Workflow materialization and optimization: The input of IReS is an abstractly described workflow, visualized as a graph consisting of operator and data nodes (Figure 1b). IReS is charged with the intelligent exploration of all the

available execution plans and the discovery of the optimal one according to the user defined optimization objectives. The supported choices include minimizing cost, execution time or a function of them. After selecting the abstract workflow and its input parameters, the user is able preview the materialized plan and inspect the platform's choices for each of the operators and intermediate results, along with an estimation of the execution cost and performance (Figure 1c). At this stage, the user will be able to validate the strategy to be followed.

Resource allocation: Apart from the selection of the optimal plan, IReS specifies the amount of resources needed for the various operators. The users will be able to inspect the provisioned resources for different operators and optimization policies.

Multi-engine workflow execution: IReS enforces the execution of the optimal plan through YARN, utilizing container based resources and orchestrating the execution of the workflow, i.e., the DAG of operators, each of which may run on a different engine. Users will have the chance to attend the whole execution process, from the allocation of the right amount of resources to the materialization of the final results.

Fault tolerance: Our system constantly monitors the health of the connected engines as well as the execution status of the workflow. This allows us to provide workflow-level fault tolerant execution. The attendees will be able to confirm that when an engine fails during a workflow execution, IReS automatically re-plans the rest of it and executes the best available alternative.

References

1. Apache Hadoop. `http://hadoop.apache.org/`.
2. Apache HBase. `http://hbase.apache.org/`.
3. Apache Spark. `https://spark.apache.org/`.
4. monetdb. `https://www.monetdb.org/`.
5. Stratosphere Project. `http://stratosphere.eu/`.
6. 84% Of Enterprises See Big Data Analytics Changing Their Industries' Competitive Landscapes In The Next Year . Forbes Magazine, 2014.
7. D. Agrawal et al. Rheem: Enabling multi-platform task execution. 2016.
8. K. Doka, N. Papailiou, D. Tsoumakos, C. Mantas, and N. Koziris. Ires: Intelligent, multi-engine resource scheduler for big data analytics workflows. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1451–1456. ACM, 2015.
9. M. Ferguson. Architecting a big data platform for analytics. *A Whitepaper Prepared for IBM*, 2012.
10. I. Gog, M. Schwarzkopf, N. Crooks, M. P. Grosvenor, A. Clement, and S. Hand. Musketeer: all for one, one for all in data processing systems. In *Proceedings of the Tenth European Conference on Computer Systems*, page 2. ACM, 2015.
11. A. Simitsis, K. Wilkinson, U. Dayal, and M. Hsu. HFMS: Managing the Lifecycle and Complexity of Hybrid Analytic Data Flows. In *ICDE*. IEEE, 2013.
12. D. Tsoumakos and C. Mantas. The Case for Multi-Engine Data Analytics. In *Euro-Par 2013: Parallel Processing Workshops*. Springer, 2014.
13. V. K. Vavilapalli et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.