Recalibration of Analytics Workflows

Maxim Filatov
Yandex, Moscow, Russia
maxfil@yandex-team.ru

Verena Kantere
School of Electrical and Computer Engineering
National Technical University of Athens
verena@dblab.ece.ntua.gr

ABSTRACT
As business decisions and strategies become more and more automated, real-time, and data-driven, enterprises need to create, manage and execute end-to-end analytics workflows that process increasing data volumes, from new heterogeneous data sources, on specialized processing engines. Workflows become more complex and time-consuming to design and execute, since they span a variety of systems and the amount of data being processed grows. Therefore, it becomes increasingly difficult to debug workflows in order to handle errors, as well as adjust the workflow design and calibrate task parameters for applications that perform exploratory data analysis. Towards this end, the workflow management system should provide recalibration methods i.e. methods to monitor and to influence workflow processing at runtime. We demonstrate novel manual and automatic recalibration techniques for analytics workflows, on real use cases and data from the telecommunication domain and web analytics, but also on synthetic use cases and data.

1 INTRODUCTION
The analysis of Big Data is a core and critical task in multifarious domains of science and industry. Such analysis needs to be performed on a range of data stores, both traditional and modern, on data sources that are heterogeneous in their schemas and formats, and on a diversity of query engines. Workflow execution can be extremely resource- and time-consuming. Thus, a system that enables such long-term analytics processes on Big Data needs to be able to show the progress of the execution and the intermediate results. This means that the user should be able to monitor which workflow tasks have been executed, their produced results, which tasks are currently executing, as well as data accessing and resource utilization based on input from the runtime machines or the visualization tool. Further, the system should allow the user to influence workflow processing. This means that the system should provide methods that enable the analytics expert to change a workflow by altering task parameters or infusing new tasks manually at runtime, or even to predefine automatic changes while she creates the workflow by providing alternative workflow branches. Such recalibration methods constitute a powerful functionality of a workflow management system, since they enable the gradual design of exploratory analytics workflows based on feedback from intermediate results, as well as efficient error handling of complex and long-running workflows.

In this demonstration we focus on the novel functionality of PAW\(^1\) (Platform for Analytics Workflows) for workflow recalibration. PAW is a platform for the design, management, analysis, optimization and execution of analytics workflows. The first version of PAW is presented in [1, 2] and includes the functionalities of workflow design and analysis in order to clarify execution semantics, single workflow optimization and multi-workflow optimization. In this demonstration we present for the first time the new functionality of PAW on workflow recalibration. It includes novel techniques for (a) manually changing a workflow at runtime and re-executing it avoiding repeated computations, called recovery and monitoring points technique (3.1); (b) automatically changing a workflow at runtime based on conditional structures if-then-else (3.2) and goto statements (3.3).

Several existing workflow management systems support conditional structures to some extent. Each of these systems implements these structures in different ways and with some limitations: Kepler [7] allows the design of scientific workflows and executes them efficiently using emerging Grid-based approaches to distributed computation. It offers a structure called Comparator, which takes two inputs and performs a numerical comparison. Taverna [8] is a well-known workflow management system that does not include conditional structures in the workflow model, but tries to achieve the if and switch functionality at a higher layer of workflow management. In Taverna such conditional behavior is implemented using processors fail_if_false and fail_if_true placed as first vertices of parallel branches. Depending of their input one of those processors fails, another satisfies and only satisfied branch continue execution. UNICORE is a grid middleware, aims to provide seamless, secure and intuitive access to distributed resources [9]. UNICORE has a programming environment to design and execute workflows. It supports three specific types of if-then-else conditions, ReturnCode, FileTest and TimeTest. The first performs a numerical comparison, the second checks if a file exists or is executable and the third checks the current time. Recalibration in PAW offers an abstract if-then-else task that can be customized for a variety of input data and complex conditions that involve the execution of fully-fledged procedures. Only Taverna offers the same level of flexibility in the design of conditions as PAW does. The rest of the considered systems are very limited in possibilities to construct a condition. Also, they don’t provide goto statements.

A new system that offers interactive debugging framework for big data processing is BigDebug [10], which provides a set of debugging primitives: (a) simulated breakpoints and on-demand watchpoints that allow users to selectively examine intermediate data of computation; (b) data provenance capability, crash culprit determination, tracing and (c) capability to fix code at runtime by the user, avoiding a program re-run from scratch. Unlike PAW, BigDebug is a single-engine system and works only on top of Apache Spark. Moreover, debugging is only one of the several uses of the manual recalibration of PAW.

Overview of PAW. PAW implements a novel workflow model [4, 5]. A workflow \(W\) is a directed, acyclic graph (DAG) \(G = (V, E)\). The vertices \(V\) represent data processing tasks \(T\) and the edges \(E\) represent the flow of data. Each task is a set of inputs, outputs and an operator. Data and operators need to be accompanied by a set

\(^1\)Source code and live demo can be found on https://github.com/project-asap/workflow
We propose three techniques that perform recalibration in an online manner, i.e., during workflow processing.

**Recovery and monitoring points.** This technique offers to the user manual recalibration. It enables the user to monitor intermediate results, make workflow changes and if the changes are in the already executed workflow part, then re-execute only the changed part, avoiding to repeat computations; if changes affect only the non-executed part then workflow changes are applied and execution continues.

**Conditional points.** This automatic technique conditionally changes an executed workflow part to a predefined alternative and re-executes it.
Recalibration using this technique is performed in four steps: (1) the user augments a workflow with recovery and monitoring points and starts the workflow execution; (2) when the execution reaches a recovery point the system stores the intermediate results of the preceding task, required for a possible re-execution of the workflow part following this recovery point; (3) when the execution reaches a monitoring point the user observes intermediate results of the preceding task; the workflow keeps executing after the monitoring point, while the user observes the intermediate results; (4) the user changes the workflow part following a recovery point and performs a re-execution of the workflow from this recovery point and on.

When the user changes the workflow and re-executes it, PAW determines which intermediate results are required to re-execute the changed workflow part that follows a specific recovery point (or points). It prepares this workflow part as a new materialized workflow with these intermediate results as input datasets and runs it. The execution of the previous (original) workflow is aborted.

Figure 3 displays the workflow from the motivating example augmented with recovery and monitoring points. The user observes the result of filter_regions at monitoring points, decides to change the parameters of filter_regions and re-executes the workflow from the recovery points. So the most time-consuming part of DataFilter is not re-executed.

Figure 4 displays a ‘Peak Detection’ workflow in the interface of PAW. Green and yellow tasks are executed and currently executing, respectively. Using the geographical map on the bottom the user monitors intermediate results at the monitoring point marked with a blue stroke.

3.1 Recovery and monitoring points

This recalibration technique allows the user to change a workflow during its execution and avoids to unnecessarily repeat computations in the already executed workflow part. It involves the employment of two novel types of tasks: recovery and monitoring points. A recovery point rPT is a task that stores the result of task T. A monitoring point mPT is a task that invokes the visualization of the result of task T or part of it. We use the phrase intermediate result to refer to the result of a task T that has been executed, while the whole workflow execution is not yet finished. The visualization of intermediate results assists the user in making a recalibration decision.

Recalibration using this technique is performed in four steps: (1) the user augments a workflow with recovery and monitoring points and starts the workflow execution; (2) when the execution reaches a recovery point the system stores the intermediate results of the preceding task, required for a possible re-execution of the workflow part following this recovery point; (3) when the execution reaches a monitoring point the user observes intermediate results of the preceding task; the workflow keeps executing after the monitoring point, while the user observes the intermediate results; (4) the user changes the workflow part following a recovery point and performs a re-execution of the workflow from this recovery point and on.

When the user changes the workflow and re-executes it, PAW determines which intermediate results are required to re-execute the changed workflow part that follows a specific recovery point (or points). It prepares this workflow part as a new materialized workflow with these intermediate results as input datasets and runs it. The execution of the previous (original) workflow is aborted.

3.2 Conditional points

PAW includes a new type of task that realizes the conditional structure of the form if-then-else. The latter allows the design of a workflow with several alternative workflow parts. Depending on the intermediate results of the task preceding the if-then-else task,
workflows is described as a motivating example (Section 2). The workflow is augmented with two tasks: a workflow branch is chosen for execution, over another one. These workflow branches are not yet executed. Figure 5 displays a workflow that has been augmented with one if-then-else point and two following workflow branches. The if-then-else task has two outputs; the boolean condition evaluates to true or false, depending on which PAW executes one of two branches.

The operator of the if-then-else point is implemented for any particular data. For example, for tf-idf PAW has an if-then-else task that evaluates if the weight of some word is above a certain value. Additional conditional points can be added through the interface of PAW.

### 3.3 Goto points

The workflow is augmented with two tasks: goto label and goto condition points, and an alternative workflow part related to the goto label point (Fig. 6). When the workflow execution reaches the goto condition point and if this task triggers 'goto' to goto label, then it re-executes the workflow from that point choosing for execution the alternative workflow part. Therefore, this technique is a combination of the recovery and monitoring points and conditional techniques.

The goto condition task has two outputs and a boolean condition evaluating to true or false, depending on which PAW continues execution or jumps to the goto label alternative workflow branch. The implementation of goto condition is similar to the if-then-else point.

### 4 DEMONSTRATION

In the following, we describe the proposed demonstration.

**System setup.** PAW is demonstrated on a cluster, with the following configuration: The cluster consists of 4 server-grade physical nodes. Each one of those is equipped with a 3rd generation i5 CPU (@ 2.90 GHz) and 16GB of physical memory and an array of two HDDs on RAID-0. The operating system is Debian 6 (squeeze) Linux. For the time being, four software platforms are running: Hadoop (CDH 4.6.0), Spark (1.4.1), Elasticsearch (5.1) and Weka (3.6.13).

**Workloads.** The demonstration uses synthetic and real workflows on real data. The real workflows and data come from the two use cases of ASAP [3] and belong to the domains of telecommunications and web analytics. One of the telecommunication workflows is described as a motivating example (Section 2). The web analytics use case involves anonymization of web content (WARC files) stored in ElasticSearch. The workflows are implemented in Spark and run over varying data set sizes ranging from 1 million to 4 billion rows. There are two types of workflows: one models entity recognition/disambiguation and k-means, and another models continuous processing of incoming data, e.g., subscription/notification at scale.

**Demonstration scenarios.** The demonstration focuses on the recalibration functionality of PAW. It includes four types of scenarios that aim to show each a distinct view of the recalibration benefits and create discussion on the potential of recalibration of analytics workflows. The demonstration is interactive with the audience. The participants are invited to experience all functionalities of PAW, create workflows from scratch or change existing ones, watch the processing of the workflow, as well as review the internals of the platform, e.g. internal workflow representation. Even more, the participants are guided to play with the recalibration of workflows, by adding recovery and monitoring points, goto and if-then-else points, and change the workflow while monitoring intermediate results in GUI of PAW.

**Scenarios A.** These demonstrate the recovery and monitoring points technique. Specifically, they show real necessities to change workflows during execution. We show real workflows which need infusion of new tasks or alteration of task parameters during the execution.

**Scenarios B.** These also demonstrate the recovery and monitoring points technique. Specifically, they show how the user can design a workflow in a gradual and modular manner, while he is testing and debugging already created parts by monitoring intermediate results. We show how this workflow design process benefits exploratory data analysis.

**Scenarios C.** These demonstrate the conditional technique for workflows with a natural conditional branching, for which data analysis based on some conditions follows different paths, and the selection between these alternative paths should be made at runtime.

**Scenarios D.** These demonstrate the goto technique using workflows that benefit from the goto point in order to find anomalies in data, narrow or refocus the search or analysis, as well as meet deadlines and milestones of analysis.

**REFERENCES**


