FP7 Project ASAP
Adaptable Scalable Analytics Platform

ASAP D7.1
Integration Prototype ASAP System Prototype v.1

WP 7 – Integration of the ASAP System

Nature: Report

Dissemination: Public

Version History

Version	Date	Author	Comments
0.1	15 Feb 2015	Maria Chalkiadaki, Polyvios Pratikakis	Initial Version
0.2	15 Feb 2015	Polyvios Pratikakis	First Revision
1.0	15 Feb 2015	Polyvios Pratikakis	Final Version
1.1	1 May 2015	Polyvios Pratikakis	Revised Version

Acknowledgement This project has received funding from the European Union's 7th Framework Programme for research, technological development and demonstration under grant agreement number 619706.

Contents

1 Introduction

The main objectives of this work package are the following:

- Ensure the integration of the contributions in WP2, WP3, WP4, WP5 and WP6.
- Coordinate development and delivery of the integrated modules based on the research and development results in the different Research Areas.
- Ensure that these prototypes are used to integrate and coordinate the coherent delivery of the ASAP system for its application in WP8 and WP9

1.1 Task Description

The task T7.1 investigates the practices adopted and rules to be followed when developing components of the ASAP system. It includes the establishment of a suitable distributed development management platform to manage and coordinate the coherent production and integration of all components by the various development teams over the duration of the project. The infrastructure enables automated testing, bug reports, and management of feature requests

2 Integration Narrative

We plan to use several scenarios, as developed in the form of use cases in WP1, as drivers for integration of the ASAP modules. In the early stage of integration we focus on the following scenario as described in the following steps for the ASAP users:

1. The Developer implements a new analytics primitive computation as an operator, using the operator language designed and implemented in WP2.
2. The Developer builds the operator and creates the metadata describing the operator's use, semantics, alternative implementations, etc., in the workflow description language developed in WP5.
3. The Developer adds the operator to a running installation of the ASAP platform, in the library of operators recognized by the IReS module.
4. The IReS profiler builds a cost model of that operator implementation.
5. The Workflow Designer adds a data source to the ASAP system and declares its metadata using the data description component of the workflow language developed in WP5.
6. The Workflow Designer creates a workflow in the workflow language developed in WP5 using the workflow editor tool.

7. The User initiates the computation of that workflow.
8. The IReS platform schedules the workflow using the best possible implementation of the operator, based on the metadata and cost profile.
9. The User can monitor the workflow execution and view the intermediate data resulting from one operator before they are consumed by the next operator of the workflow.
10. The User can decide to override the IReS schedule and force a specific implementation of the operator.
11. The system adapts without restarting the whole workflow.

3 Integration map

3.1 ASAP Components

The ASAP platform is decomposed in the Application Management tools, the Intelligent Multi-Engine Resource Scheduler (IReS) and the Runtime Tools.

The Application Management Tools is a UI suite further decomposed in the following components:

- Query Description Tool: a workflow editor which enables users to express complex analytics queries over multiple and heterogeneous data. The output is encoded in the workflow language defined in the scope of the WP5 and consists of operators deployed to the operator library maintained by the IReS and define calculations over various data sources.
- Visualisation Cockpit: a tool for enabling live monitoring of the query progress and statistics. The methods employed for optimal information representation are subject of the WP6.
- Query Cockpit: a tool for visualizing intermediate results of a long-running workflow. The tool, upon a user's request, queries the monitoring components residing in the Runtime level of the ASAP Platform and are developed in the scope of WP5.

The IReS is a platform consisting of the following components (thoroughly described and implemented in the scope of WP3):

- Job parser: a tool which receives as input from the Application Management UI user-defined operators or workflows to be executed, validates them and identifies required metrics over the data and execution artifacts. Its output is a number of stored subtasks in a dependency graph.

- Modeling and Learning Engine: a tool which receives a parsed operator and evaluates the projected cost and performance under several execution scenarios such as several storage engines with variable resources per execution.
- Decision Making Module: a module which decides which part(s) of the job are executed over which technology using analyses and comparing against already stored models extracted by the Modeling/Learning Engine.
- ASAP Scheduler: a bunch of tools for scheduling the query execution to the most beneficial runtime and datastore available and handling any failures.
- Runtime Deployment Inspector: a tool for managing dynamically resources among the query execution engines and data stores depending on the executed job.

Finally, the lower level ASAP, the Runtime Tools encompass:

- Execution Engines: a number of parallel engines to run queries over heterogeneous data stores. Besides the already existing engines for analytics computations there is also an execution engine permitting recursive analytics computations. Its design and implementation is subject of WP4.
- Compute and Storage Monitor: scripts which monitor the executed jobs and store intermediate and final job results.

Figure 1 depicts the ASAP components and their interdependencies.

3.2 ASAP Integration

The Web Analytics and Telecommunications Applications described in WP8 and WP9 respectively have provided workflows useful for standardising the workflow description language proposed in WP5. Moreover, they have served also as a pilot for conducting a set of operators required for the workflow execution. These operators are implemented in Cilk and Swan (as proposed in WP2) or in spark etc and are uploaded (via REST) to the IReS which is able to profile and execute them without having knowledge of their implementation details. Therefore a JSON-based metadata language is employed as a general abstraction.

Finally, the industrial partners have provided the cluster infrastructure and data for performing testing and experimentations. Access to the clusters is granted via SSH using keys provided to the partners upon request.

Whenever a new materialized operator gets inserted in the IReS platform the system estimates its cost in terms of performance and money using several configurations and saves each plan in a database. The later is queried by the Decision Making Module in order to choose the optimal execution plan for a workflow uploaded by the user for execution.

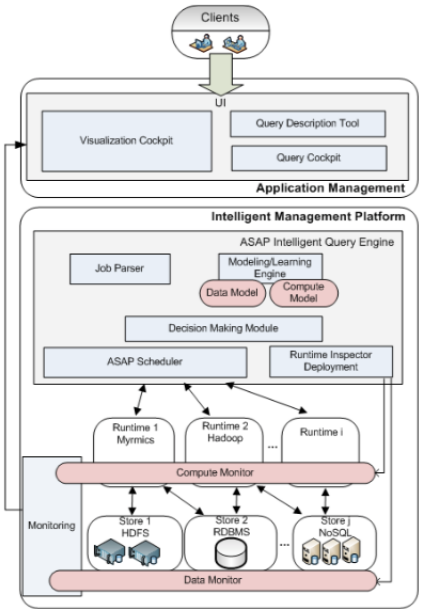

Figure 1: ASAP Technology Overview

Figure 1: ASAP components

The subtasks resulted by the workflow analysis are sent (via REST) to the execution engines which are also installed in the above infrastructure according to the selected plan. The execution is monitored by wrappers developed in the scope of WP5 which flush the output to an Elasticsearch engine.

Finally, the Visualization and Query cockpits query the Elasticsearch in order to visualize the statistics and the intermediate results respectively.

4 Integration prototype

This chapter describes briefly the tools employed in order to ensure smooth integration of the software components and enhanced stability of the overall system.

4.1 GitHub

We decided to use GitHub [2] for sharing code and documents between project partners. GitHub is a web-based graphical Git repository with Web-based graphical interface and a number of extra features that make GitHub powerful.

4.2 Jenkins

We installed Jenkins [3] for continuous integration of the code of our project. With Jenkins, we can easily build the uploaded code in automated way. It monitors the building and the execution, stores and e-mails the outputs to the appropriate partners.

5 Source Code

The ASAP outcomes will be open-source. Therefore, each partner has its own GitHub account and repository for uploading its code. At this moment, there are two repositories: one for hosting the code of the Profiler and another hosting the execution engine developed in the scope of WP4.

6 Execution engine

We have forked Apache Spark 1.1.0 and have introduced some modifications in order to perform nested reduction calculations. These modifications are discussed in more detail in D4.1. The Apache Spark core component is rewritten in Scala [8] and uses SBT [7] for compilation and building, so we do. Moreover, the Apache Spark core component contains a test suite which we use and we will extend in order to reinforce the system's stability.

7 Profiler

Both the Profiler and the IReS platform are implemented as Web servers offering REST APIs that are used for their integration with the other ASAP modules. The REST servers are implemented in Java using the Jetty [5] servlet engine and the Jersey [4] RESTfull Web service framework. Apache Maven [1] is used to compile, build and package the code of both the Profiler and the IReS platform. We also use JUnit [6] to automatically run unit tests for the basic functionalities of our system. The IReS platform and the Profiler will be packaged as .deb or .rmp packages in order to facilitate their installation using automated installation tools like apt and yum.

In order for the Profiler to offer a wide variety of surrogate estimation models for the modeling of different operators, we integrate with WEKA [9], an open-source machine learning and data mining software which implements a variety of machine learning algorithms. WEKA is also Java based and its integration is performed using maven dependencies.

References

[1] Apache maven. http://maven.apache.org/.

[2] Github. https://github.com/.

[3] Jenkins. http://jenkins-ci.org/.

[4] Jersey. https://jersey.java.net/.

[5] Jetty. http://eclipse.org/jetty/.

[6] junit. http://junit.org/.

[7] Sbt. http://www.scala-sbt.org.

[8] Scala. http://www.scala-lang.org.

[9] Weka. http://weka.wikispaces.com.

FP7 Project ASAP
Adaptable Scalable Analytics Platform

End of ASAP D7.1
Integration Prototype ASAP System Prototype v.1

WP 7 – Integation of the ASAP System

Nature: Report

Dissemination: Public