FP7 Project ASAP
Adaptable Scalable Analytics Platform

D1.3 Updated User Requirements and System Architecture

WP 1 – Architecture and Requirements Analysis

Nature: Report

Dissemination: Public

Version History

Version	Date	Author	Comments
0.1	9 Feb 2016	M. Chalkiadaki	Initial Version
0.2	13 Feb 2016	P. Pratikakis	First Revision
0.3	24 Feb 2016	P. Pratikakis, M. Chalkiadaki	Comments by K. Doka
1.0	24 Feb 2016	P. Pratikakis, M. Chalkiadaki	Comments by P.Rigaux

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.

Executive Summary

This document outlines the results of the requirements analysis for the ASAP project. It (i) describes issues relating to the ASAP system design, (ii) introduces the required functionality for the analytics applications, and (iii) presents the high-level architecture of the ASAP system as a whole.

The process for analyzing user requirements was based on interviews with engineers from all partners, as well as face-to-face all-consortium discussions. During the first year, the results were analyzed and discussed in several working groups, arriving at the specification of a set of use cases that overlap all areas of the proposed research and also a set of functional and non-functional requirements for all ASAP work packages. During the second year of the project, based on feedback from the development of each module, we refined the user requirements and enhanced the ASAP architecture specification.

This document is an extension of deliverables D1.1 "Early User Requirements" and D1.2 "User Requirements and System Architecture". It incorporates the use case requirements of all modules, including system modules, as well as better and more detailed definitions of the requirements of the analytics applications.

Contents

1 Introduction

One of the objectives of project ASAP is to perform research and development towards solving existing problems in data analytics. A deep understanding of the users' real requirements is a crucial factor in the development of any high-quality software product. Similarly to software engineering, research objectives and results have greater impact when they are applicable to — or even driven by— real problems in IT. To that end, the project includes two industrial partners (WIND, IMR) that are facing such problems and limitations in their real analytics applications, and would stand to gain by the proposed research results. During the first year of the project the research partners developed use cases and use-case requirements regarding the corresponding modules of the ASAP analytics system. During the second year, these use cases have been used to drive development, testing and integration of the ASAP modules. We have amended the existing use cases and added additional use cases as development progressed throughout the second year of the project.

As in most software development and research projects, user requirements were not completely clear at first. This was due to several reasons.

- Developers (researchers) did not have a clear idea of the user domain and specific needs of the application, even if they did understand the problem at an abstract level.
- Users did not have a clear view of what they may gain by using the technology developed, how it is applicable to their problem, or even what is and is not possible to achieve compared to the existing solutions.

For these reasons, we followed a process that elicits the requirements and specifications of the ASAP analytics applications in such a way that they form *well-defined* and *achievable* research objectives. However, as research projects usually include a higher level of uncertainty than traditional software development, it is not possible to follow a standard waterfall model for user requirements analysis and specification. Instead, during the first year of the project we developed an initial set of user requirements and specifications for the project applications. During the second year of the project, using feedback from development and testing of individual modules, as well as the testing of their integration and deployment in the ASAP applications, we updated the existing use-cases where necessary and added new use-cases when an aspect of a module or an issue with integration was not covered. Furthermore, their analysis, re-evaluation and re-definition is an on-going process that interacts with the remaining work packages, adapting to new constraints as they are discovered. This document extends the specifications of the analytics applications use cases and use-case requirements, includes use cases and use-case requirements regarding the research modules of the ASAP system, and presents the architecture of the ASAP system.

1.1 Purpose of this Document

This document presents the results of *Work Package 1: Architecture and Requirements Analysis* (WP1) regarding the user specifications and use cases developed within *Task 1.1: User Requirements*, as well as the architecture design of the ASAP system planned within *Task 1.2: Design of ASAP Architecture* of WP1.

A goal of project ASAP is to design and develop an analytics platform for the two analytics applications of consortium partners WIND and IMR, regarding telecommunication analytics and web analytics, respectively. To do that, the project ASAP develops a set of modules that enable and facilitate the satisfaction of the applications' requirements. As such, WP1 aims to elicit the specifications, requirements and constraints of these analytics applications and the ASAP system modules; and define open problems to drive the research and development performed in the project.

This report summarizes the activities carried out between March 2015 and February 2016 towards the development and update of user requirements definitions and specifications for ASAP and its two driving analytics applications, in WP1. We describe the methodology, to-date findings, and use cases developed during the phase of requirements analysis and all subsequent development. The results of this analysis influence work in other work packages and subsequent research. Finally, note that this document covers the analysis of use-case requirements for each ASAP module and analytics application; The detailed specification of each design, as well as the description of the current status of implementation per module is described in their corresponding WP deliverable documents.

1.2 General Description of ASAP

The ASAP project aims to develop a unified framework for data analytics. This section summarizes a high-level view of the project for convenience. We refer the reader of this document to the project Description of Work for a detailed discussion and analysis of the objectives and their motivation.

In short, ASAP aims to build a unified, open-source execution framework for scalable data analytics. It is based on the idea that (i) no single execution model is suitable for all types of tasks; (ii) no single indexing and data-store is suitable for all types of data; and (iii) an adaptive system that has correctly modeled analytics tasks, costs and is able to monitor its behavior during tasks is a more general, efficient way of tackling this problem.

ASAP aims to develop the technology to facilitate the development and execution of general-purpose analytics queries over irregular data. To achieve this goal, the project focuses on the following objectives:

- Develop a general-purpose task-parallel programming model, implemented by a task-parallel execution engine, making the development of complex, irregular datacenter queries and applications as easy as writing regular Map-Reduce computations. The task-parallel runtime will incorporate all the benefits of Map-Reduce systems and state-of-the-art task-parallel programming models, namely: (i) express irregular general-purpose computations, (ii) take advantage of resource elasticity to use resources only when required by the application, (iii) hide synchronization, data-transfer, locality and scheduling issues from the programmer, (iv) be able to handle large sets of irregular distributed data, and (v) be tolerant to node, system, or disk faults.
- Develop an intelligent management platform that models and manages multiple execution and storage engines to the submitted jobs. The modeling framework must take into consideration the type, location and size of data, the type of computation and available resources in order to decide on the most advantageous store, indexing and execution pattern available. To that direction, our system will complement our execution model with existing open-source solutions (Map-Reduce) as well as with state-of-the-art distributed storage engines (NoSQL, column-stores, distributed file-systems, etc.) in order to have a broad applicability and increased performance gains.
- A unique adaptation methodology that will enable the analytics expert to amend the task they have submitted at an initial or later stage. This is a process often required for analytics tasks that fail to capture the users' intention due to erroneous parameter or dataset choices. ASAP will be able to adapt the execution strategy according to the already created results and the changed parameters.
- A monitoring methodology that will enable the analytics expert to obtain accurate, intuitive and timely results of the analytics tasks they have initiated. Through a visualization engine, initial and intermediate results and meta-analytics will be shown in real-time, enabling the scientist to assess the usefulness of the method.

2 Methodology

In traditional software engineering, the requirements analysis process includes surveys, interviews, and work-group meetings, in which domain experts and engineers co-author a set of use cases that best describe the desired behavior of the system. The system specifications can then be extracted from the use cases in the next step of the process, by software architects [1].

ASAP requirements analysis applies this process to derive a set of use cases from the two analytics applications, using interviews, face-to-face meetings and working group discussions with expert engineers from WIND and IMR. The goal of the requirements analysis is to develop a set of use cases that must satisfy both constraints, that is (i) have interesting properties that overlap the proposed research areas in Work Packages 2 through 6, and (ii) be realistic scenarios of the two analytics applications, so that the technology developments will immediately benefit these applications.

Of course, the waterfall model is rarely directly applicable in software development, and never in research projects, where solutions, designs, and constraints may not be obvious at first. Thus, we perform an initial requirements analysis to elicit use cases that can be used to drive the research and development process, and benchmark results, while at the same time allowing the specification of requirements to evolve for the first part of the project, in what would correspond to a more agile software engineering methodology [2]. This document presents the to-date results of the requirements analysis, as a set of use cases that together can be used to specify the desired behavior of the ASAP platform that would benefit the analytics applications of the ASAP industrial partners.

2.1 Teleconferences

During the first and second year of the project, we conducted several teleconferences involving representatives from all partners. Among other agenda topics, the teleconferences involved brainstorming and plannin discussions on user requirements, and their implementation and integration. Based on the material presented in the kick-off meeting, two face-to-face meetings during the first year and one face-to-face meeting during the second (as posted on the project Wiki), these discussions resulted in a more detailed understanding of existing practices of the industrial partners on behalf of the research partners. Moreover, they resulted in a better understanding of the goals of the new technology proposed on behalf of the industrial partners.

2.2 One-on-one interviews

In addition to teleconferences involving the whole consortium, we conducted one-to-one teleconferences with representatives from industrial and research partners. During the first year of the project, we focused this technique on the WIND Telecommunication Analytics application, as it involves objectives and ideas that are currently not implemented. The objective for the Telecommunication Analytics application is to develop entirely new functionality using ASAP technology. In contrast, the IMR Web Analytics application is currently implemented to a large extent using an array of existing analytics technologies. It still stands to gain in performance, flexibility, and also novel functionality by using technology proposed in ASAP, although that refers to new components and functionality added to an existing application. During the second semester of the project, we conducted online teleconferences to extract use case requirements from all research partners regarding the corresponding ASAP modules.

The objective of the one-to-one teleconferences was to arrive at representative scenarios for the applications that clarify:

- The objective of the application.

- How the application creates value and why its development is desired by the industrial partner.
- The nature, size, complexity, and availability of data involved.
- The kinds of computations required to implement parts of the application.
- The actors involved in operating the application.

Resulting from these interviews, we were able to arrive at several indicative application scenarios. These were then discussed in a focus group among the whole consortium.

During the second year of the project, research partners formed working groups with industrial partners, related to the modules involved in each use case. We used working group teleconferences to coordinate development of the new functionality in ASAP modules and their use in the development of the ASAP applications, and their use-cases in particular. Furthermore, during the second year we used the ASAP code repository and teleconferences among iterested partners (with frequency ranging from three per week to one per month) to organize the integration of implemented modules into the use-cases of the industrial partners, the deployment of all software on both IMR and WIND clusters, and regression and integration testing.

2.3 Focus group

During the first face-to-face meeting of the ASAP project, we performed summary presentations of the two applications, involving scenarios and any specifications elicited during the teleconferences. Also, research partners presented specifications of the research work packages. The objective of the presentations was to communicate the constraints and problems attacked by the planned research to the industrial partners, to help define the properties of desirable use cases that, if found in an application, would fit well with the planned research.

During the second face-to-face meeting of the ASAP project, the industrial partners extended the specification of both applications in more detail, focusing on points of interest to the research partners, where each application overlaps the corresponding research modules. Moreover, the research partners presented and specified an early set of requirements and designs for the corresponding modules, depending on the progress of each work package. Detailed results on the design and implementation progress regarding the research modules can be found in the corresponding work package deliverable documents. This document focuses on the use case requirements extracted through that process, as well as the overall architecture of the ASAP system.

During the third face-to-face meeting of the ASAP project, the industrial partners presented implemented or partially-implemented workflows corresponding to the application use cases, and discussed their integration with ASAP modules. The research partners presented the status of the ASAP module implementation and the plan for maximum integration with the implemented applications by the end of the second year. Detailed results on the progress of the implementation of the ASAP modules and their integration and use by the application use-cases, as well as testing results, can be found in the corresponding work package deliverable documents.

2.4 Use cases

Finally, during all face-to-face meetings, we used focus groups to elicit a set of use cases that best fit the properties and objectives desirable to the research partners, while still being part of the analytics applications of the industrial partners. In the third face-to-face meeting which took place during the second year of the project, the focus groups discussed technical details and time plan required to achieve maximum integration of the application use-case implementations with the existing ASAP modules by the end of the second year.

3 System Architecture

This section presents use cases for the whole of ASAP system. It is an extension of the corresponding section in Deliverable D1.2, where use cases 3.4 and 3.5 have been refined or extended.

We refer to the following actors as users of the system:

- Developer: The programmer that designs and implements an analytics operator or custom computation. Usually an expert programmer who does not have to be familiar with the details of the workflows that will use the operator.
- Workflow Designer: The analytics expert who designs analytics workflows by combining available operators. The workflow designer need not know the details of the operator's implementation, but must understand its effect on data.
- User: The non-expert user of a workflow, e.g., a marketing expert trying to discover trends or associations, or a web-user that issues a specific query.

3.1 Use Case: Define Workflow

Description The Workflow Designer uses a workflow editor to define a workflow, as a sequence of operators on data sources.

Preconditions

- The operators are declared in an operator library.
- The schemata of the data sources are known.

User Requirements

- The Workflow Designer should be able to create workflows by connecting operators to data sources.
- The workflow created should be executable by the ASAP system.

3.2 Use Case: Monitor a Long-running Workflow

Description The Workflow Designer monitors intermediate data between phases of computation (operators) during the execution of a long-running workflow. For example, the Workflow Designer queries the average value in a dataset between two operators of the workflow, visualizes a histogram of the data, etc.

Preconditions

- The workflow is long-running.
- The workflow includes multiple phases, which communicate via storing intermediate results.
- The intermediate results can be monitored while parts of the workflow are running.

Requirements

- The User can query the intermediate data between two phases of the workflow.
- The Workflow Designer can create visualizations of intermediate data.

3.3 Use Case: Calibrate a Long-running Workflow

Description The Workflow Designer monitors intermediate data in a long-running workflow and calibrates a parameter of one of the computations performed in the workflow. For example, by monitoring the average of all values in a table of intermediate data, the expert adjusts a threshold value for a subsequent phase in the workflow.

Preconditions

- The workflow includes multiple phases, which communicate via storing intermediate results.
- The workflow execution allows the monitoring of the intermediate data during its execution.

Requirements

- The User should be able to monitor the intermediate data between operators in a workflow.
- The User should be able to adjust the parameters of an operator used in the workflow.
- The online adjustment of an operator of the workflow should not require the restart of the whole workflow, but may require the restart of a phase.

3.4 Use Case: Profile and Model

Description The Developer declares multiple ways of implementing an operator. The ASAP system profiles alternatives with respect to variables such as input dataset size, required memory, etc.

Preconditions

- There is an implementation of an operator and its metadata description.

Requirements

- The system should know which input parameters to sample and which output metrics to measure during profiling in an automatic or developer-defined way.
- The system can learn a cost model per operator implementation based on its declaration by the Developer, by automated or guided profiling of the operator.

3.5 Use Case: Workflow Execution Plan

Description The user executes a workflow. The system uses the cost models of operator implementations to generate an efficient execution plan of the workflow instance.

Preconditions

- There are multiple available implementations of an operator (phase) of the workflow.
- The profiler has learned a cost model for the alternative implementations or the developer has provided one.

Requirements

- The system should execute the workflow correctly.
- The system should optimize the workflow execution plan by selecting the best implementation for each operator based on dynamic properties of the workflow instance (input size, threshold values, etc.)
- The system should decide on missing input parameters (e.g., number of nodes, etc.)
- The system should handle multi-objective optimization.

3.6 Use Case: Optimize Performance for Multiple Workflows

Description A system services multiple arriving requests, each serviced by the execution of a workflow. The system achieves high throughput of workflows by dynamically selecting the best execution plan depending on data set sizes or other parameters of a request. For example, a workflow is executed per web-user request to analyze a subset of the large, complete data set; depending on the subset size, maximum throughput of workflows is achieved by selecting between single-node shared-memory implementation or hadoop distributed memory implementation of the workflow operators.

Preconditions

- There are more than one alternative implementations/plans of a specific workflow.
- There are multiple instances of the workflow running on the system.
- Selecting the optimal implementation of the workflow depends on dynamic parameters (dataset size, threshold value, etc.)

Requirements

- The ASAP system should dynamically execute the optimal implementation of a workflow instance.
- The total throughput of workflow executions should exceed the throughput achieved by not dynamically optimizing.

3.7 Use Case: Develop Optimized Operator

Description A Developer monitors the performance of a frequent workflow and detects a slow-running operator. The Developer implements an alternative version of the operator that performs better for that specific query and introduces it in the library of available operators. The ASAP system adjusts by optimizing that workflow to use the faster implementation when its preconditions are met.

Preconditions

- The Workflow Designer is able to detect bottlenecks in a workflow.
- The Developer is able to customize the implementation of an operator to outperform its generic version.

Requirements

- The Developer can declare the constraints of an operator implementation (e.g., can only be used on tables that fit in memory).
- The system can check which implementations of an operator apply.
- The system can estimate and compare the costs of alternative plans of the same workflow.

3.8 Overall System Architecture

Figure 3.8 shows the use-case diagram for the use cases described above. Based on this use-case analysis, we have arrived at a high-level view of the system architecture, as shown in Figure 3.8. The ASAP system comprises of a set of modules aiming to address the above requirements, namely:

- Workflow description language: The language in which Workflow Designers define workflows.
- Operator definition language: The language in which Developers can implement operators or other analytics computations.
- IReS profiler: The module that profiles operator implementations.
- IReS modeling engine: The module that uses profiler data to build operator cost models.
- IReS Decision making module: The module that, given cost models, selects an optimal plan for a workflow with multiple alternative implementations.

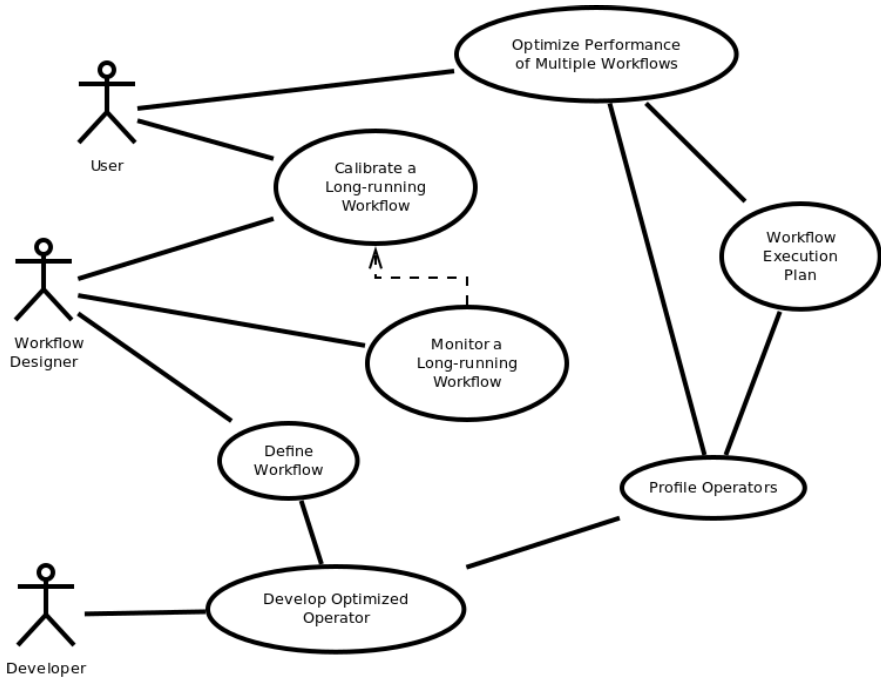
Optimize Performance of Multiple Workflows
User
Calibrate a Long-running Workflow
Workflow Execution Plan
Workflow Designer
Monitor a Long-running Workflow
Define Workflow
Profile Operators
Develop Optimized Operator
Developer

Figure 1: Use-case Diagram: Use cases of the ASAP system as a whole.

- IReS enforcer module: The module that monitors and implements the execution of a workflow execution plan.
- Recursive analytics runtime system: An execution engine that allows Developers to write recursive analytics computations.
- Online monitoring: The module that allows the User to query intermediate results during the execution of a long-running workflow.
- Online adaptation: The module that allows the User to change the parameters of a long-running workflow without restarting the whole workflow computation.
- Visualization: The module that presents the results of an analytics computation for the User.

In addition to the modules developed within the Project, ASAP will use existing components to implement the requirements, namely:

- Existing Analytics Runtimes: Existing execution engines for analytics computations, such as Hadoop, Spark, Flink, etc.

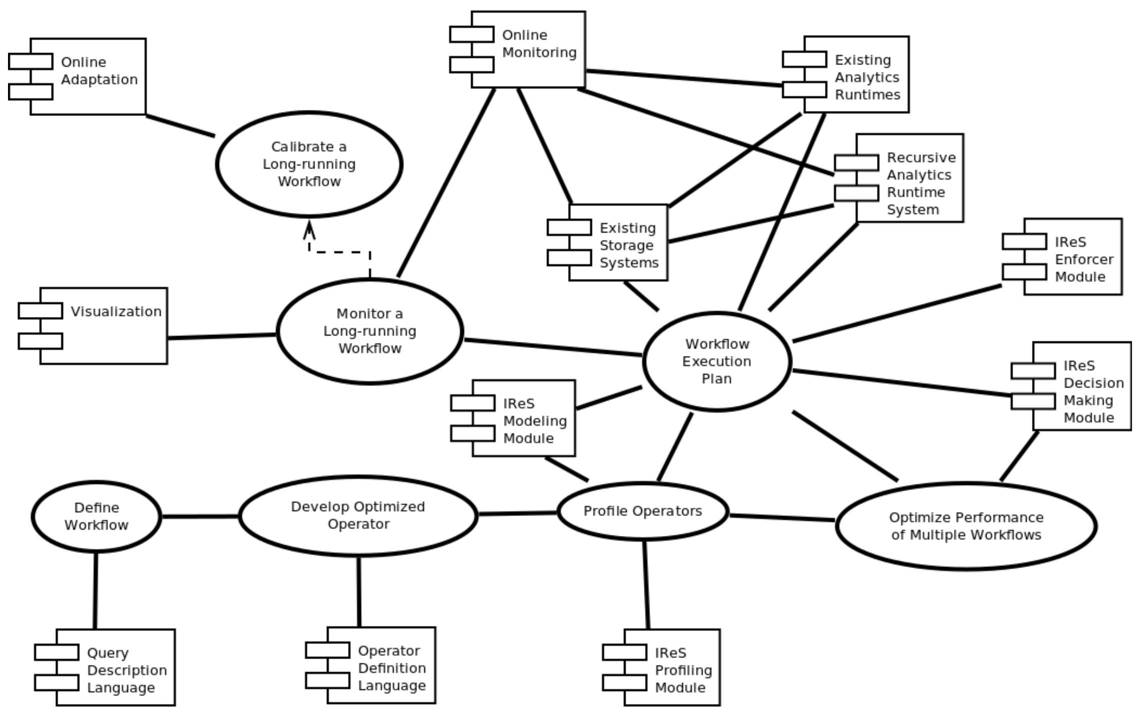
Online Adaptation
Online Monitoring
Existing Analytics Runtimes
Calibrate a Long-running Workflow
Recursive Analytics Runtime System
Existing Storage Systems
IReS Enforcer Module
Visualization
Monitor a Long-running Workflow
Workflow Execution Plan
IReS Decision Making Module
IReS Modeling Module
Define Workflow
Develop Optimized Operator
Profile Operators
Optimize Performance of Multiple Workflows
Query Description Language
Operator Definition Language
IReS Profiling Module

Figure 2: Architecture: The modules of the ASAP system

- Existing Storage Systems: Data storage systems used by the execution engines (HDFS, key-value stores, RDBMS, etc.)

The remainder of this deliverable presents use cases specific to each component of the ASAP system, and derives user requirements.

4 Language

This section describes two use cases relating to WP2, the development of the ASAP query language and operator language. We describe the use cases below and elicit user requirements,

4.1 Use Case: Recursive Analytics Computation

Description The Developer implements an analytics computation that can be recursively parallel and irregular. For example, an operator that recursively visits a tree or graph, creating parallel computations per node discovered, as opposed to “flat-parallelism” map-reductions.

In more details, the graph analytics algorithm is the following:

- *Goal*: cluster data sources (servers, publishers) and identify the most influential ones.
- *Input*: a set of documents and their URLs crawled from the internet.
- *Output*: a list of data sources and a numeric value indicating how influential they are. Higher values imply more influential.
- *Algorithm*:
  1. Build a list of data sources based on the crawled URLs. Data sources may be identified by server name.
  2. Retrieve URLs used in HTML href elements.
  3. Build graph where nodes correspond to data sources and edges occur from data source S to D when there exists a document on data source S that has a link to a document on data source D. The edge has a numeric element associated to it that corresponds to the number of such documents.
  4. Evaluate the PageRank algorithm on the graph.
  5. Sort data sources by relevance.

Requirements

- The Operator definition language must allow the Developer to invoke computations inside other computations.

4.2 Use Case: Alternative Implementations

Description The Developer implements an optimized version of an operator that outperforms the default implementation under certain constraints. For example, the Developer implements a specialized map-reduce or k-means computation tailored for data sets that fit in memory, or data sets that fit in a single node.

Requirements

- The developer should be able to write multiple implementations of an operator, optimized for different platforms or data sets, in the Operator definition language.
- The developer should be able to specify constraints for the use of each alternative implementation of an operator, such as data sizes, the existence of an index for the data, etc.
- The workflow designer must be able to use an operator without knowing its internal implementation details.

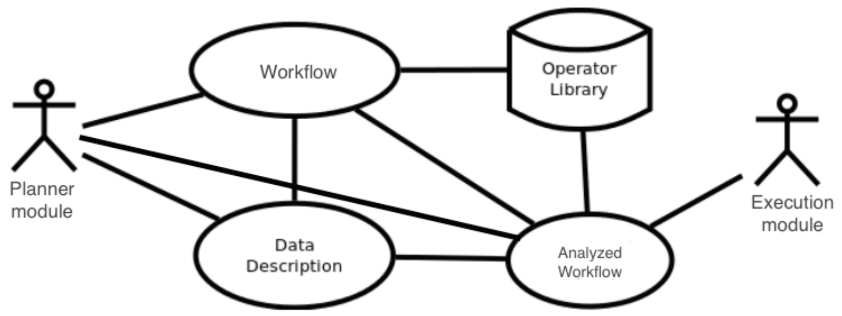

Figure 3: A summary of the workflow management use case

- The ASAP system should be able to automatically select among different implementations of the same operator.

4.3 Use Case: Workflow Specification

Description The Workflow Designer uses a set of existing operators, (described appropriately in a specification language), to design an analytics workflow. For example, the Workflow Designer connects a filtering and a k-means operator to a data source to create a clustering of the data.

Requirements

- The workflow language should be able to describe data sources and operators.
- The workflow language should include a library of standard operators and data sources.
- The Workflow Designer should be able to graphically express an analytics workflow using the available operators.
- The ASAP system should parse and execute the workflow.

5 Online Monitoring and Adaptation

5.1 Use Case: Workflow Management

Figure 3 displays the actions involved in workflow management. The objective of this use case is to use a set of methods for manipulating the workflow in order to change its structure and execution.

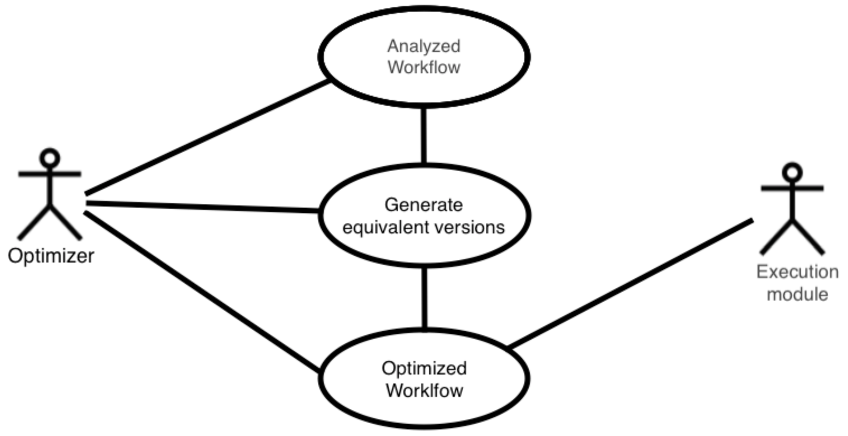

Figure 4: A summary of the processing deployment use case

Actors **Planner module** The module that takes as input the initial workflow posed by analytics expert and creates methods for the production of an analyzed version of the workflow that shows the detailed and partial tasks on multiple data sources.

Execution module The module that executes analyzed workflow produced by planner module.

Requirements **UR-1** The planner module should analyze the workload and determines on which data source or sources each task should be executed.

UR-2 In new analyzed workflow each new task should be augmented with information on the data to be accessed and produced.

UR-3 The planner module should determine dependencies between tasks and take them into account when creating analyzed workflow.

5.2 Use Case: Processing Deployment

Figure 4 displays the actions involved in processing deployment. The objective of this use case is to change the structure of a workflow so that it can be executed more efficiently than originally designed.

Actors **Optimizer** The module that takes as input the analyzed workflow and produces an optimizated version of a workflow.

Execution module The module that executes the optimized workflow.

Requirements **UR-1** The optimizer should generate all possible equivalent workflow versions taking into account the characteristics of tasks.

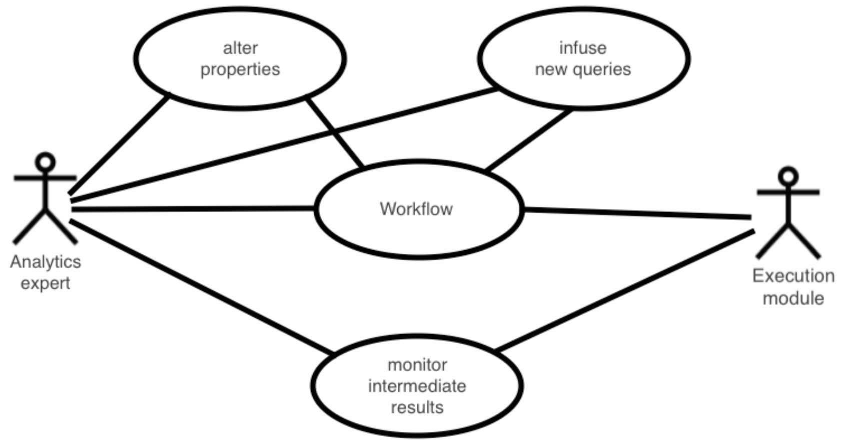

Figure 5: A summary of the dynamic re-calibration use case

UR-2 The optimizer should select an optimized workflow version based on the estimation of processing costs.

UR-3 The optimizer should be able to perform optimization of multiple workflows.

5.3 Use Case: Dynamic Re-calibration

Figure 5 displays the actions involved in dynamic re-calibration. The objective of this use case is to change the original workflow by altering parameters or infusing new queries, while analytics expert monitors the progress of processing from the runtime machines or the visualization tool.

Actors **Analytics expert** The actor who monitors intermediate results of processing workflow and changes parameters or reshedules tasks on-the-fly.

Execution module The module that executes the specified workflow and provides the intermediate results to the analytics expert. It includes methods for scheduling temporary data replication in order to perform updates and integration.

Requirements **UR-1** The analytics expert should obtain accurate, intuitive and timely results of the analytics tasks he has initiated (Monitoring progress of processing).

UR-2 The analytics expert should be able to change parameter values as he inspects intermediate results, either in original data formats or visually.

UR-3 The change of parameter values should directly reflected in the analyzed workflow and the respective individual or combined task execution schedules.

UR-4 The re-calibration methods should take into consideration requirements on meeting deadlines and milestones of processing.

6 Recursive Query Execution Runtime

6.1 Use Case: Table Join

Description In this use case, the Developer has implemented a naïve Table Join operator recursively, by performing a map-reduction on the first table, where the mapper function performs a new map-reduction on the second table, for each element. The objective for the ASAP system is to correctly execute the "nested" analytics queries and take advantage of the additional parallelism.

Requirements

- The Developer should be able to write operators that spawn analytics computations (e.g., map-reductions) within other analytics computations, recursively.
- The execution engine should be able to take advantage of all available nodes to execute the analytics queries.
- The execution engine must transparently handle "inner" queries while load-balancing the total computation over all available resources.

6.2 Use Case: Hierarchical K-Means

In this use case, the Developer implements Hierarchical K-Means, a well-known analytics algorithm of hierarchical decomposition of a data set by iterative applications of classification. Using language extensions implemented in the ASAP system, the developer should be able to express computation as a single, recursive application of classification and decomposition that includes many parallel steps, instead of iterative steps that execute strictly one after the other.

Requirements

- The Developer should be able to write a hierarchical classification and decomposition computation without implementing iteration of separate queries.
- The execution engine should be able to take advantage of all available nodes to execute the analytics queries.
- The execution engine must transparently handle "inner" queries while load-balancing the total computation over all available resources.

6.3 Use Case: Distributed Scheduling

In this use case, the Developer has implemented a hierarchical query or a series of standard queries with fine granularity. In traditional settings, a single scheduler may not be able to sustain all the overhead and “keep busy” all the available worker nodes. Using a distributed scheduler the user should be able to run these fine-grain queries seamlessly by assigning more than one scheduler nodes without worker nodes being idle, even for fine-grain computation loads.

Requirements

- The User should be able to execute queries with large parallelism and little computation on large numbers of worker nodes.
- The execution engine should be able to take advantage of more than one scheduler node to keep all worker nodes busy without idling.

7 Intelligent Resource Scheduling (IReS) platform

7.1 Use Case: New Operator/Dataset Submission

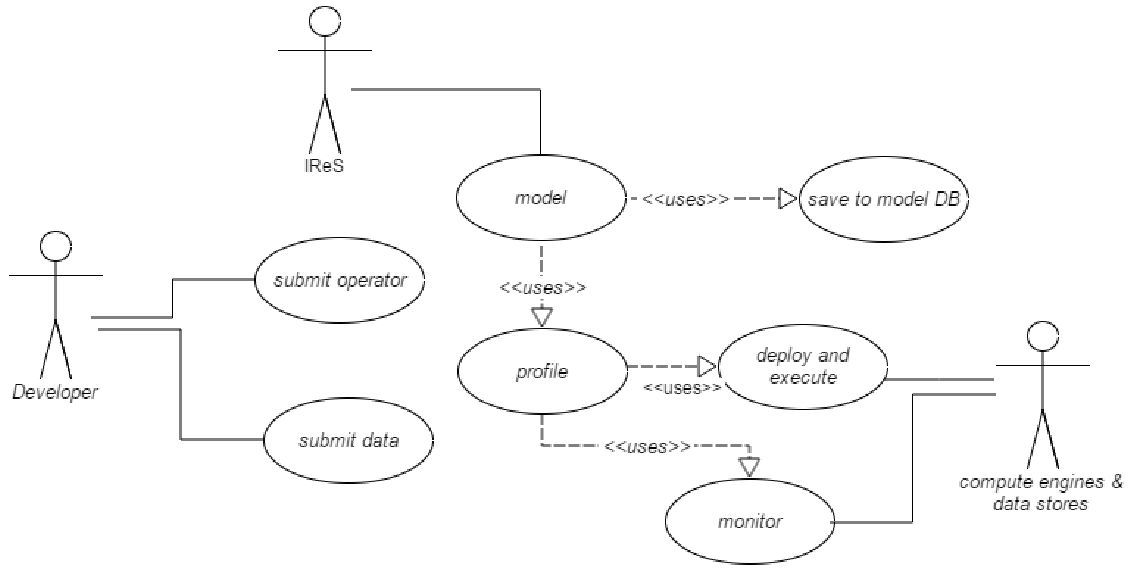

Figure 6: Summary of New Operator/Dataset Submission use case

Submit a new operator and/or dataset to the Intelligent Resource Scheduling (IReS platform). The submission of an operator triggers the offline modeling process, which profiles, i.e., automatically executes and monitors the operator using different resource and dataset configurations. The offline modeling process can help our decision module have a steeper learning curve and avoid planning errors for operators with unknown performance.

Description The developer submits a newly developed operator to the IReS operator library and/or makes a new dataset available for future use in the construction/execution of a workflow. Upon submission of an operator, the operator is profiled under various input parameters and its models are saved in the IReS model DB for further use when planning and executing workflows.

Post-conditions The new operator or the new dataset along with its metadata description is saved in the IReS operator or dataset library respectively and is ready for use. The operator is modeled and its models are saved in the IReS model database.

Pre-conditions The operator has been developed by the developer. The dataset is available.

Triggers The developer invokes a submission request on the UI.

7.2 Use Case: Basic flow

1. The developer submits an operator executable and/or makes a dataset available to the platform.
2. In case of an operator submission, the developer optionally accompanies it with a representative dataset, i.e., a dataset that can be used in the profiling process.
3. The developer provides with the metadata description of the operator and/or dataset.
4. The system saves the operator and/or dataset and its description in the operator library.
5. In case of an operator, the system triggers the modeling process.
6. The system profiles the operator with a number of different operator configurations.
7. The system deploys the operator over the infrastructure and executes it with parameters defined by the profiler.
8. The system monitors various performance metrics in order to identify the relationship between a specific configuration and the operator's performance.
9. The system created a number of machine learning models using the monitored metrics.

10. The system saves the models in the model DB for future reference.

Requirements

- Generate the most accurate profile within a specified time budget

7.3 Use Case: Workflow Planning/Execution

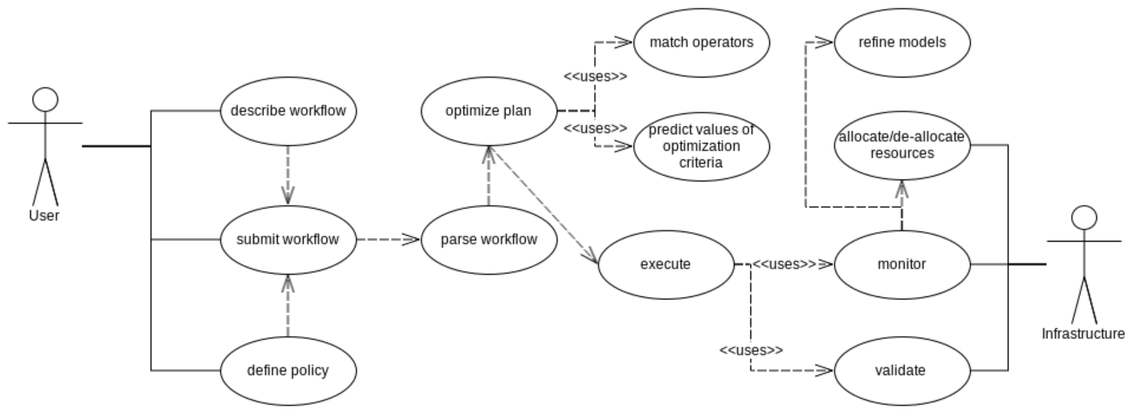

Figure 7: Summary of Workflow Planning and Execution use case

Submit and execute a workflow, optimizing the user-provided criteria.

Description The user submits a workflow along with some optimization policy. The IReS platform discovers all alternative execution plans that combine existing operators and datasets and decides on the one that optimizes the user-defined policy. The resulting plan is validated and enforced.

Post-conditions The workflow is executed in a way that optimizes the user-defined optimization criteria.

Pre-conditions The user provides an abstract description of the workflow she wants to execute. Additionally, she defines some optimization policy, be it performance, cost, fault-tolerance etc.

Triggers The user submits a workflow for execution.

Basic flow

1. The user provides an abstract description of the workflow she wants to execute.
2. The user provides the optimization policy that she wants to enforce on its execution. This policy can consist of one or a function of multiple operator performance metrics like cost, execution time, etc.
3. The system parses the workflow.
4. The system matches the abstract operators present in the user provided abstract workflow with the materialized operators imported in the platform's operator library and creates the materialized graph of the workflow that contains all the possible alternative execution plans that match with the abstract workflow plan.
5. The system locates the optimal plan, i.e., the one that best matches to the user defined policy.
6. The system validates the optimal plan against the actual infrastructure.
7. The system enforces the execution of the optimal plan.
8. The system monitors the utilization of the engine resources to (a) refine the operator models and (b) take decisions for allocation and de-allocation of computing resources in order to improve the workflow execution.

Requirements

- *Abstraction*: The user should be able to describe her workflow in any level of abstraction.
- *Optimization*: Optimize the execution according to the (multiple) user-specified criteria.
- *Fault tolerance*: locates the top-k best plans in order to be able to fall back to the execution of another plan.
- *Real-time reaction*: The system should be able to detect if the actual plan execution deviates largely from its expected execution.
- *Elasticity*: IReS should be able to allocate and de-allocate computing resources in order to improve the execution of the workflow.

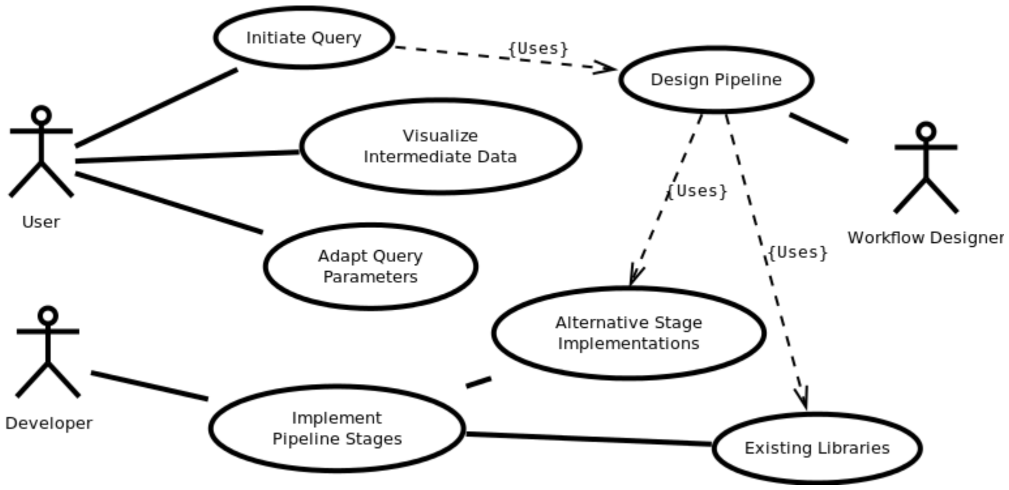

Figure 8: A summary of the web analytics use case.

8 Web Analytics

The Web Analytics application includes several data flow processes and query workflows. Figure 8 presents a summary of the application actions. In short, developers implement stages of computation pipelines, which are then synthesized by workflow designers, and executed to answer user queries. This chapter presents three indicative use cases selected for relevance to the ASAP research Work Packages. The use cases are based both on (i) existing implementations redesigned to take advantage of the optimizations proposed in ASAP, and on (ii) development of new computations that were not previously possible with existing tools. Initially, the use cases are described and specified to target a small data set of almost 18k documents, averaging 200kb per document; in total 139GB of data. This is so that use cases are easy to deploy outside of the actual IMR data center, to facilitate research and development by the ASAP research partners. We plan to eventually deploy and test the ASAP platform on the larger, actual data set in the IMR data center.

8.1 Use Case: NLP-classification

Figure 9 presents an overview of the data flow in this use case. The first query in the pipeline uses Elasticsearch to select documents from the document store. The results are processed to annotate each document with its metadata, in JSON format.

The second stage extracts the textual content from the documents using one of three alternative algorithms, two of which are open-source and one is proprietary, developed by IMR, then appends the extracted plain texts to the annotated documents as additional metadata. The stage outputs the annotated documents with the extracted plain texts in JSON format, and sends them directly to the next stage.

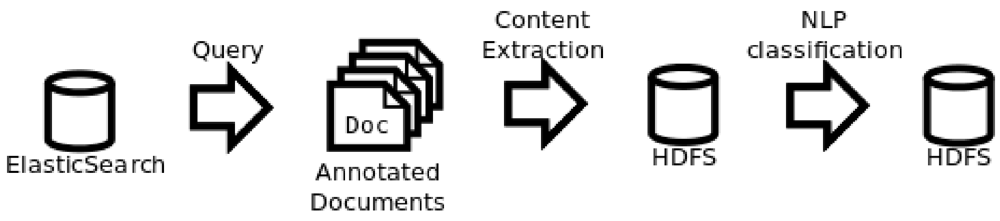

Figure 9: The data flow of the first use case: storage and computation phases.

In the final stage of this use case performs NLP classification on this data, using one of two alternative implementations. The stage also appends classification results to the annotated documents as additional metadata. The three stages are pipelined and deployed on multiple worker machines in a cluster. Intermediated data of each stage are directory propagated to the next stage deployed on the same machine. Finally, the resulting data of the final stage are stored in HDFS.

This use case captures a typical form of the "current" IMR Web analytics pipeline such that (i) a pipeline is a sequence of operators applied per document, and (ii) each operator is implemented as an exchangeable component which allows us to freely choose and connect to define a pipeline specialised for customer's requirements.

The use case fits well with several of the desired properties for the research work packages in ASAP:

- It contains several stages that compute intermediate data (WP3, WP5, WP6).
- It is an on-line computation, as the initial query is customized by the user (WP5).
- It contains more than one alternative implementations per state of computation (WP3).

Moreover, as an existing part of the Web Analytics application it clearly benefits from the results of that research.

Requirements

- Given an Elasticsearch query, the cost of the pipeline can be estimated for all possible alternative implementations of each stage.
- The user can visualize or query intermediate data between stages while the next stage is being computed.
- The user can choose to adapt a parameter of the query that is used in a later stage, after the initial query starts its computation, but before that stage has finished its computation.
- The developer can implement logic of each stage as an exchangeable component with enough metadata including schema of input data and output data.

- The workflow designer can define a pipeline by using a well-abstracted language without coding.

8.2 Use Case: K-Means

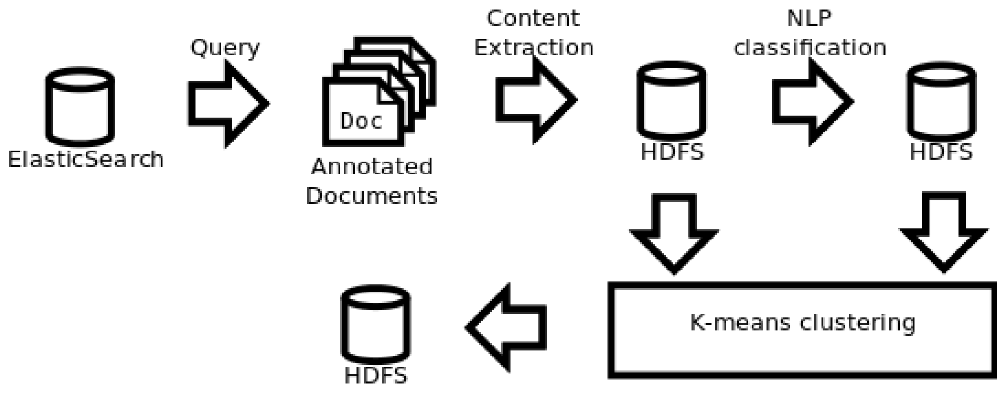

Figure 10: The data flow of the second use case: storage and computation phases.

In contrast to the first use case, the second use case is not currently implemented in the Web Analytics application and will be implemented in ASAP. Figure 10 shows the data flow of the second use case. As described above, the user submits a query to Elasticsearch, selecting a set of annotated documents. The second phase similarly extracts the relevant content from the annotated documents. In addition to these computations the second use cases includes a last stage of computation that reads the annotated document content as produced by the content extraction phase, the results (feature vectors, classification output, etc) of the NLP classification computation, and performs clustering using the k-means algorithm. Alternative implementations of k-means clustering are included in systems such as Mahout, WEKA, Spark MLlib.

This use case is an example of applications which IMR currently does not have but intends to implement in the near future for more advanced analysis. The framework which IMR is currently using does not have operators which abstract iterations in a pipeline, and decision making engines to determine the best engine and strategy for executing a pipeline.

Since it is very similar, this use case is also very relevant to many of the research objectives of ASAP. In addition, the k-means computation is an iterative algorithm, which performs sub-optimally when expressed in Map-Reduce. A better iterative computation engine (such as Spark) already outperforms Mahout because of this phenomenon. We expect that for the same reason, the k-means iterative computation is an interesting motivating example for ASAP research:

- The algorithm is iterative and hierarchical, includes data decomposition and redistribution in every step and so, would benefit from a programming model with such abstractions (WP2).

- It fits the characteristics of algorithms that will benefit from a dynamic dependence analysis in the presence of imbalance (WP4).
- The results of k-means clustering and NLP classification are visualizable in an intuitive way (WP6).

Requirements

- The k-means clustering algorithm should have multiple implementations.
- Some of the clustering alternative implementations should allow the user to adjust the parameters of computation.
- The k-means algorithm must be expressable in the ASAP programming model.
- The developer can implement (or can reuse an existing implementation of) k-means logic using iterations as an exchangeable component.
- The workflow designer can define a pipeline without coding.

8.3 Use Case: Named Entity Disambiguation

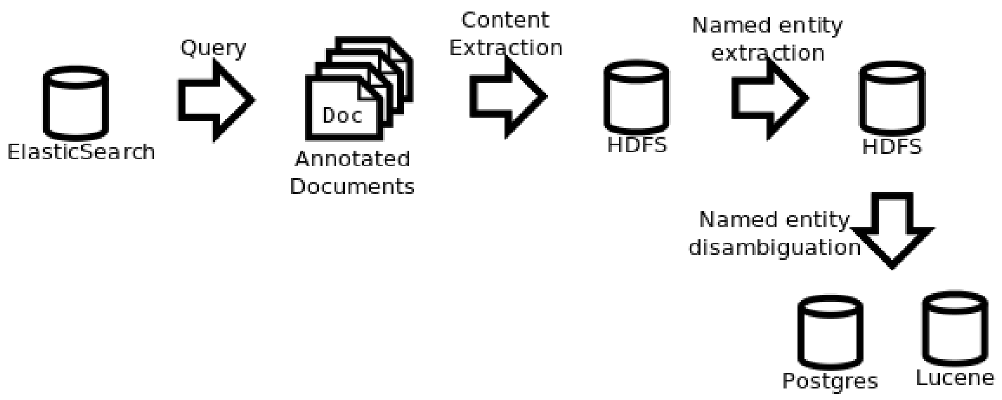

Figure 11: The data flow of the third use case: storage and computation phases.

Figure 11 presents the data flow among all phases of the third use case. As above, document selection and preprocessing remains the same. The post processing of selected documents has two phases. The first phase is named entity recognition (NER) which seeks occurrence of named entities in plain texts. It is performed by using one of existing libraries including GETA, Stanford NER, etc. The last phase is named-entity-disambiguation (NED). It determines identities of named

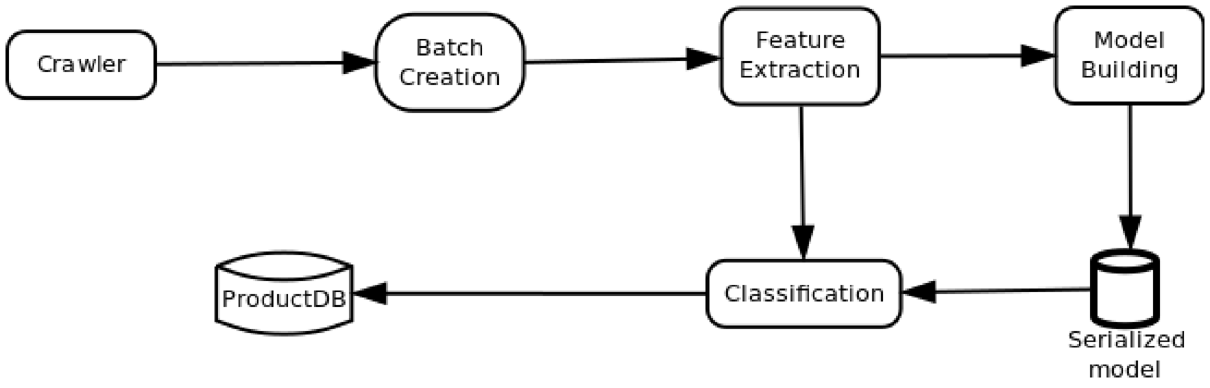

Figure 12: Web Analytics Application with continuous classification.

entities in texts and link them to a knowledge base like DBpedia. There are several open source implementations of NED including DBpedia spotlight and AIDA. It is long-running computation using large (50-100GB) indices of target knowledge bases, installed in different software (PostgreSQL, Lucene, etc), sometimes by using specialised hardware (SSD, large main memory, etc.). The result of NED is stored in HDFS.

This use case is one of applications that IMR is experimentally doing, but has difficulty to put into production because it requires manual intervention in scheduling and deployment of a pipeline. We expect this use case to demonstrate improvement using ASAP research, by distributing the last phase of the computation as well as overcoming resource-bound performance constraints. This use case is different to the previous two in terms of interaction, as it is not triggered by user queries and is a long-running, off-line, computation.

Requirements

- The computation of NED should be distributed to multiple machines in a cluster by taking into account machine resources.

8.4 Use Case: OutWatch Product Matching

This use-case consists of OutWatch, a Mignify service that uses a continuous classification process which forms a basic part of the Mignify platform. OutWatch builds and continuously maintains a catalog of product references, and discovers product offers. To do that, it computes and maintains a model of all products, continuously updating the model with new data, while using the model to classify products into categories for accuracy.

Figure 12 presents the data flow among all phases of the OutWatch use case. Continuous data supplied by a crawler is batched into computation batches. Each batch is analyzed to result into a model and a set of features, subsequently used to perform machine learning and continuous adaptation of an existing model. As the model is updated with the data in the new batch, a more precise classification of the data is possible.

Requirements

- The system should create batches from a continuous input and schedule them through all computation phases.
- The workflow is tree-like; one branch for updating a classification model, one other branch for classifying item with the updated model.
- The following dependency should be satisfied by the scheduler: the first branch has to be executed before the second one.

9 Telecommunication Analytics

The Telecommunication Analytics application includes many possible query workflows, on-line queries, off-line long-running computations and ad-hoc analytics. This chapter presents indicative use cases selected for relevance to the ASAP research Work Packages. As with the use cases presented in the previous chapter, the use cases are selected to allow for the optimizations and novel features that are being researched in ASAP.

Initially, the use cases are described and specified to target a small data set of Call Detail Record (CDR) data from five regions of Italy for a duration of a few days, describing many millions of calls and averaging about 1GB per region. All user data are anonymized in this "development" small data set. This is so that use cases are easy to deploy outside of the WIND data centers, to facilitate research and development by the ASAP research partners. ASAP plans to perform experiment and test the developed technology on actual data and WIND infrastructure.

The current status of the implementation includes modules for three basic analyses: peak detection, user profiling, and sociometer. Moreover, for data visualization purpose a module to compute spatio-temporal aggregation over time is implemented. Those modules are used to build workflows in the ASAP platform as shown in 13.

9.1 Use Case: Peak detection

Figure 14 displays the actions involved in this use case. We assume an anonymization process occurs at regular times, e.g., weekly or monthly, refreshing the data available in the analytics database from real user data. The second phase involves the processing of the anonymized CDR data for clustering along the geographical area and time. The geographical area is partitioned to a set of regions and the time is partitioned into time slots. Both time slots and regions are provided by the user. These two parameters define a spatio-temporal grid and density which is the number

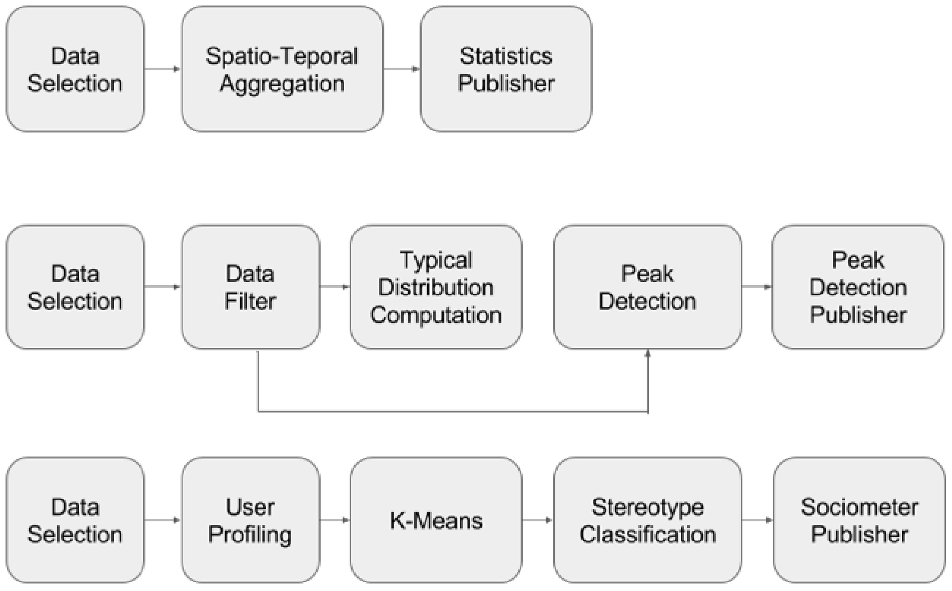

Figure 13: Data Analytics Applications in common basic analyses.

of observation per cell. Moreover, the input data is partitioned into two sets: a training dataset and a test dataset. For both datasets the spatio-temporal grid of densities is computed. The first is used to compute the densities of a typical period for each region. The second dataset is then compared against such typical period in order to detect significant deviations. The objective of this processing is to detect peaks in load, according to a set of criteria. Criteria may include:

- A granularity of deviations, expressed as a percentage relative to the expected density;
- A minimum relative deviation, also expressed as a percentage, used to select significant deviations;
- An absolute minimum deviation, expressed as an integer number, used to discard extreme cases with very low densities.

or other parameters of the clustering algorithm.
These parameters should be adjustable by the analytics engineer, marketing expert, etc., who uses the peak analysis results. The results of this phase are added to a database (relational or graph DBMS) that contains peaks detected in previous data. The database of peaks can then be queried by a user, to discover clusters of calls that occur with or without any regularity or similar ad-hoc queries based on the pre-computed peak data.

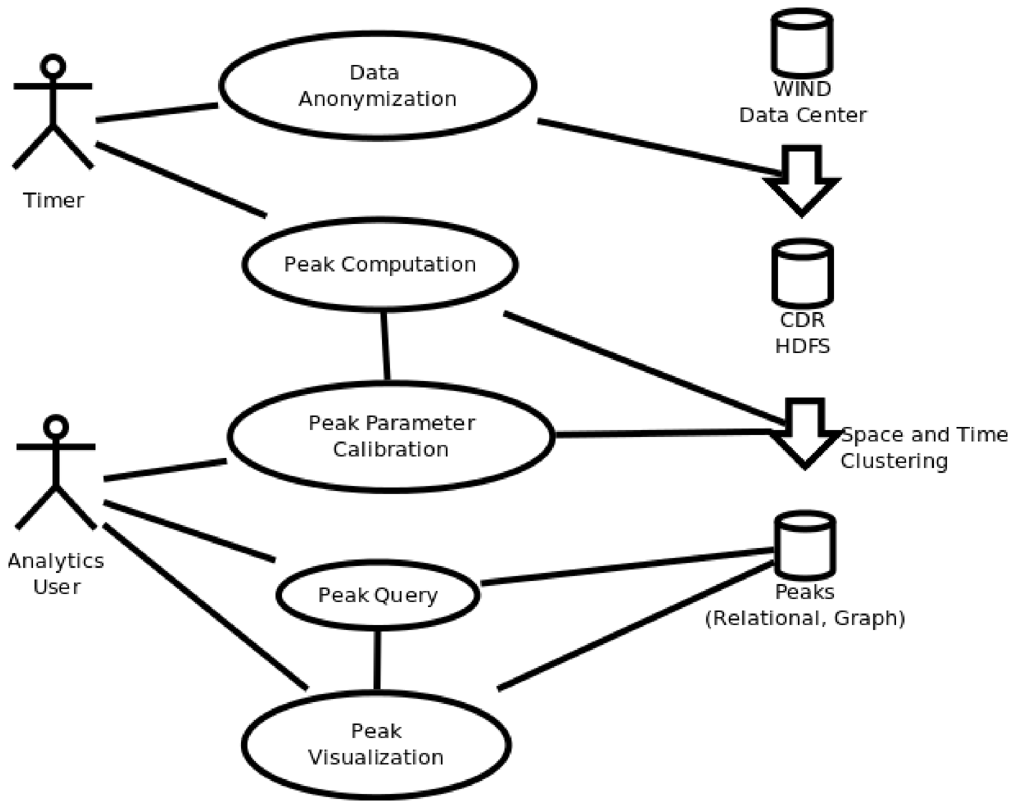

Figure 14: Use case diagram of the peak detection scenario.

9.2 Actors

Timer Some of the computations in this use case are expected to be triggered automatically at a regular basis.

Analytics User An actor that can query or visualize the results of the peak detection analysis. Based on this feedback, the Analytics User is able to readjust the parameters for peak detection.

9.3 Requirements

UR-9 The data coming from the data center must be anonymized.

UR-10 Every anonymization maintains user identity across different calls by the same user but removes all other private user information.

UR-11 Clustering of calls into peaks should be highly parallel and distributable, using either existing parallel implementations in existing execution engines (e.g., Hadoop) or written in the ASAP programming model and runtime.

UR-12 The user should have a high-level way to visualize and query the database containing the detected peaks.

UR-13 For call aggregation over the spatio-temporal partitions, it is necessary an operator equivalent to the *Distinct* of the SQL and the possibility of performing *Join*.

9.4 Use Case: User's call profiling

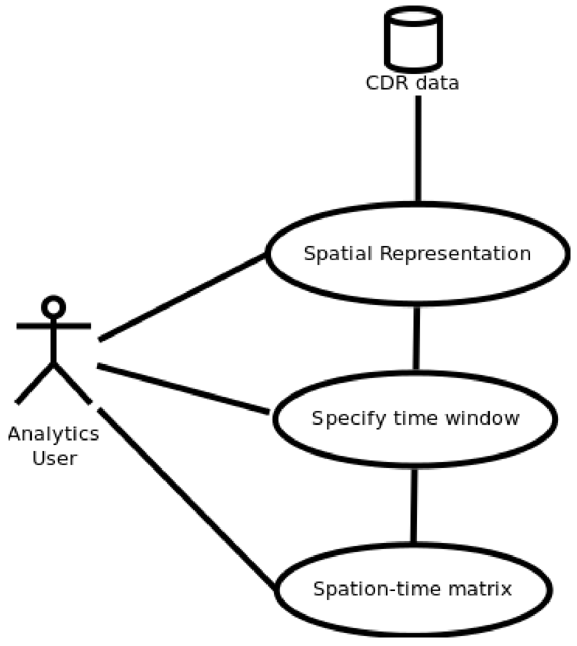

Figure 15: Use case diagram of the user's call profiling scenario.

Figure 15 displays the actions involved in this use case. The objective of this use case is to detect the presence of a specific user in a particular time window, using the CDR data and according to an area of interest. The presence of a user is based on a matrix structure and representation of the spatio-temporal user profile.

9.5 Actors

Analytics User An actor that queries for a specific user in an area of interest and in a particular time identified by the information of the CDR data and creates a matrix representation of the spatio-temporal user profile with the results.

9.6 Requirements

UR-15 In this case the primitives needed are the aggregation of the calls over a the spatio-temporal partitions and an operator equivalent to the *Distinct* of the SQL.

9.7 Use Case: Origin/Destination Matrix for systematic flows

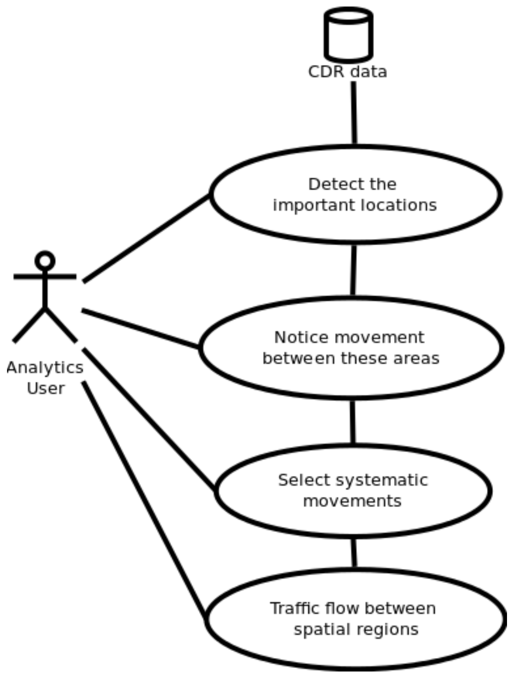
CDR data
Detect the important locations
Analytics User
Notice movement between these areas
Select systematic movements
Traffic flow between spatial regions

Figure 16: Use case diagram of the O/D matrix for systematic flows scenario.

Figure 16 displays the actions involved in this use case. This use case is separated into two stages. At the first stage, at a specific area and using the frequency of calls made by an individual user, it is extracted the two most important locations for that user. The objective for that step is to notice the movements between these two locations. The second step is to synthesis the origin and destination matrices that includes the traffic flows between spatial regions.

9.8 Actors

Analytics User An actor that queries for a specific user the important locations and the systematic movement between them and calculates the sequence between these two areas over the total places that visited in a day.

9.9 Requirements

UR-16 In this case the requirement for ASAP platform is the possibility of searching a specific sequence (i.e., from home to work) over the sequence of places visited in a specific day.

9.10 Use Case: Sociometer

Exploiting the methodology called Sociometer it is possible to classify the users using the presence of cellphone users. Once the profiles have been created, the Sociometer classifies them implementing a set of domain rules that describes our mobility behavior categories. In particular we are interested to identify residents, commuters and visitors.

- A person is Resident in an area A when his/her home is inside A. Therefore the mobility tends to be from and towards his/her home.
- A person is a Commuter between an area B and an area A if his/her home is in B while the work/school place is in A. Therefore the daily mobility of this person is mainly between B and A.
- A person is a Dynamic Resident between an area A and an area B if his/her home is in A while the work/school place is in B. A Dynamic resident represent a sort of opposite of the Commuter.
- A person is a Visitor in an area A if his/her home and work/school places are outside A, and the presence inside the area is limited to a certain period of time that can allow him/her to perform some activities in A.

10 Use Cases and ASAP Objectives

Based on the overall objectives of the project, the objective of WP1 is to perform user requirements analysis and arrive at specifications and use cases that (i) are part of the ASAP analytics applications and (ii) overlap all research areas in ASAP and provide interesting benchmarks for the technology being developed. The above use cases indeed overlap with one or more components of new technology in the platform:

- a new analytics programming model that will incorporate a user's cost and performance requirements;
- an intelligent management platform that models and manages multiple execution and storage engines to the submitted jobs;
- an analytics execution engine that enables the user to amend queries at a later stage;
- a unique runtime monitoring methodology for retrieving the progress of analytics jobs in real time; and
- state of the art visualization tools and UIs that enable intuitive, real time access to the processed data and computations.

10.1 Irregular Parallelism and Graph Computations

Research proposed in *Work Package 2: A unified analytics programming model* (WP2) aims to design and develop a programming model in which it will be possible to express irregular parallel computations. Such computations cannot usually be expressed in Map-Reduce, or can be expressed only as expensive iterative Map-Reduce computations, because they include data-dependent flow of control or compute properties that are not data-parallel or results of simple reductions. Usual examples of irregular queries are graph computations. Some graph computations are easy to express as Bulk-Synchronous Parallel (BSP) computations, which require execution engines such as Giraph or Pregel. Other graph computations are not easy to express in BSP, either because they use multiple kinds of data stores, multiple kinds of graphs, etc.

ASAP aims to develop a programming model that is able to express arbitrary graph, tree, or hierarchical computations. To drive the development of this idea, we investigated whether the two analytics applications could take advantage of such functionality. The above use cases, especially use cases 3, 4 and 5, can take advantage of such technology.

In addition, *Work Package 4: A dependency-aware query execution engine* (WP4) proposes to design and implement a query execution system that will be able to efficiently execute such computations, without reducing them into iterative super-steps, as these executions often suffer from load imbalance. The execution of such queries will include detection of dependencies between units of computation, so that they are synchronized and scheduled for maximum efficiency (on the same node, if possible), without delaying the rest of the system. Graph computations with multiple steps would immediately benefit from such technology. Therefore, the above use cases in the applications including such irregular and data-dependent graph computations, can be used to motivate and drive the research in WP2 and WP4, as well as evaluate their results.

10.2 Operands with Alternative Implementations

Work Package 3: Intelligent, Multi-engine Resource Scheduling Platform (WP3) aims to develop a modeling, cost-estimation and high-level scheduling platform for optimizing queries and workflows with many alternative implementations of different cost and performance. The elicited use cases indeed showcase these characteristics: workflows of queries, where one or more components have multiple alternative implementations. Such examples can be difficult to schedule as exploring the space of all possible parameters is prohibitive. They would, therefore, benefit the most from the cost estimation and scheduling system developed in WP3.

10.3 Workflows, Long-running Queries, Adaptive Computations

The objectives of *Work Package 5: Adaptive Data Analytics* (WP5) include the development of methods for monitoring long-running queries and long workflows, allowing analytics experts to understand partial results or intermediate data and calibrate the parameters of the query without stopping it. Such functionality will best benefit computations that involve long workflows or dataflow graphs of queries, where monitoring the intermediate results may give insights much quicker than waiting for the whole computation. As both applications include scenarios that may result in such computations, the elicited use cases also demonstrate these desirable properties, and thus stand to gain from the developed adaptive query technology.

10.4 Resource-bound Performance Constraints

In many cases, even when there are alternative implementations for parts of a workflow, storage formats, execution engines, etc, it is not clear that the most time-efficient should always be selected, when optimizing for total cost. Research in WP2 aims to allow for constrained cost optimization in the scheduler and to benefit such use cases that include queries with constrained and resource-bound scheduling performance constraints.

10.5 Visualization of Intermediate Results

Finally, *Work Package 6: Information Visualization* (WP6) aims to develop visualization tools for monitoring query data and also visualizing intermediate information during query execution, in combination with work in WP5. Indeed, the selected use cases include computations that produce visualizable information, including as intermediate data in long-running workflows.

Conclusion

This document describes the to-date results of the requirements analysis process in ASAP. The process has resulted in several use cases covering interesting aspects and computations from both analytics applications and research modules of the ASAP system. Each selected use case motivates several research and optimization topics in the research work packages. In addition, the use cases assist in designing mock-ups, early prototype implementations and evaluation benchmarks. Finally, the requirement analysis of the ASAP applications has evolved during the second year of the project, based on the results and design of the planned research as well as feedback from implementation and testing. This document was updated to include such extensions and refinements.

References

[1] H. D. Benington. Production of large computer programs. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 299–310, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[2] Michael Hicks and Jeffrey S. Foster. Score: Agile research group management. *Commun. ACM*, 53(10):30–31, October 2010.

FP7 Project ASAP
Adaptable Scalable Analytics Platform

End of D1.3 Updated User Requirements and System Architecture

WP 1 – Architecture and Requirements Analysis

Nature: Report

Dissemination: Public