CoherentPaaS

Coherent and Rich PaaS with a Common Programming Model

ICT FP7-611068

Use Cases Requirement Analysis

D9.1

March 2014

Document Information

Scheduled delivery	31.03.2014
Actual delivery	04.04.2014
Version	1.2
Responsible Partner	PTIN

Dissemination Level:

PU Public

Revision History

Date	Editor	Status	Version	Changes
03.01.2014	Vassilis Spitadakis	Draft	0.1	Draft Table of Contents
20.01.2014	Pedro Miguel Neves	Draft	0.2	Outline Document
14.02.2014	Raquel Pau Patrick Valduriez	Draft	0.3.1	Chapter 5 and Chapter 7
04.03.2014	Vassilis Spitadakis	Draft	0.3.2	Chapter 3 and Chapter 4
09.03.2014	Pedro Miguel Neves	Draft	0.3.3	Chapter 6
10.03.2014	Pedro Miguel Neves	Draft	0.3	Internal Draft
11.03.2014	Pedro Miguel Neves	Draft	1.0	Version for peer review
23.03.2014	Luis Cortesão	Draft	1.1.1	Internal Draft
01.04.2014	Vassilis Spitadakis Raquel Pau Patrick Valduriez	Draft	1.1.2	Internal Draft
02.04.2014	Luis Cortesão	Draft	1.1.3	Internal Draft
02.04.2014	Luis Cortesão	Draft	1.1	Restructured according to review
04.04.2014	Luis Cortesão Vassilis Spitadakis	Draft	1.2	Fulfilling reviewers' comments and preparing final release
11.04.2014	Luis Cortesão Vassilis Spitadakis	Final	2.0	Fulfilling reviewers' comments Final Delivery

Contributors

NEUROCOM, SPARSITY, PTIN, INRIA
Pedro Miguel Neves, Luis Cortesão, Vassilis Spitadakis, Raquel Pau, Patrick Valduriez

Internal Reviewers

FORTH, QUARTETFS
Angelos Bilas, Antoine Chambille

Acknowledgements

Research partially funded by EC 7th Framework Programme FP7/2007-2013 under grant agreement n° 611068.

More information

Additional information and public deliverables of CoherentPaaS can be found at: http://coherentpaas.eu

Glossary of Acronyms

Acronym	Definition
CDR	Call Detail Record
CEP	Complex Event Processing
CPaaS	CoherentPaaS
CQM	Client Quality Management
CSP	Communications Services Provider
DAP	Data Access Pattern
E2E	End-to-End
IoT	Internet of Things
KPI	Key Performance Indicator
KQI	Key Quality Indicator
M2M	Machine-to-Machine
MVC	Model View Controller
OLAP	Online Analytical Processing
OSS	Operation Support System
QoE	Quality of Experience
RDBMS	Relational Database Management System
VTP	Vehicle Telematics Provider

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Cloud data management, Big Data, and the Internet of Things (IoT) raise specific challenges with respect to scalable data management, both in real-time and offline.

In this landscape there is an increasing demand for efficiency and scalability that has resulted in the implementation and use of a wide diversity of cloud data stores – each one specialized and optimal for specific processing, thus leading to a "no one size fits all" solution. This trend has resulted in a large proliferation of APIs, a lack of common programming framework and a lack of coherence across different cloud data managers for the corresponding different technologies (traditional environments provided full coherence that has been totally lost in the cloud landscape). CoherentPaaS (CPaaS) addresses all these issues.

CPaaS outcomes will enable application developers to program with a unified framework that will attain simplicity thanks to the common query language and holistic coherence, scalability guaranteed by cloud data management technology, and efficiency by enabling the use of different cloud data managers specialized for the required tasks.

The main purpose of WP9 - Use Cases - is the analysis design implementation and finally validation of the use cases against the CPaaS programming model.

First step will be the analysis of the requirements of each use case, paying special emphasis to the application requirements for the data storage and management and their deployments on the cloud, as these requirements will be used as input for further development of the CPaaS programming model.

The next step will be the reengineering as well as the design of the use cases to support CPaaS. Special focus will reside on identifying the common components and modules that could be reused by more than one use case and the associated API of such components.

The use cases will be implemented taking into consideration the services and model offered by the CPaaS programming model.

Finally, the use cases will be deployed in the CPaaS cloud in order to validate the common programming model.

Task 9.1 - Use cases requirement analysis collects detailed requirements and expectations from the stakeholders involved in the design of the use cases. The scenarios describe the business and technological domains of each use case owner and the development processes relevant to each case study. The scenarios detail the work products and functionality required providing the basis for the use cases design.

This deliverable first provides a detailed description of the use cases. Each use case provides a description of the business scope and the overall business framework, a description of the main functionalities realized through the use case and concludes with the technical architecture framework details.

Second, this deliverable provides an analytical description of the requirements that are affecting the data store technology, with particular focus on identifying extreme and problematic conditions. Most suitable Data Stores (or group of Data stores), for the identified requirements, are mentioned here.

Each use case includes a brief description of forthcoming development process, mainly indicating the development approach - design and development from scratch or re-engineering of existing application.

The description concludes with the identification of the CPaaS platform validation process through the particular use case - what will be measured, which are the goal values and definition of key performance indicators (KPIs) in terms of correctness/completeness/performance.

Section 2 will provide an overview of each use case, including how CPaaS will be used and what are the main issues to be solved. Section 3 and Section 4 will encapsulate the two constituent parts – Call Detail Records (CDR) analytics and Machine-To-Machine (M2M) - of Neurocom telecom framework and cloud M2M use case. Section 5 will address Sparsity media planning use case, Section 6 will cover PTIN real-time network performance analysis in telco environment use case and, finally, Section 7 will focus on INRIA bibliographic search use case.

Section 8 will include a summary of the data requirements of the use cases, an identification of common requirements and components (particular focus on those that imply the need for a common component or functional feature in the unified platform) and it will wrap-up with the overall validation guidelines.

For all use cases, there is more detailed documentation in Appendix A (Section 9) regarding business orientation, functional description and technical framework.

2 USE CASES OVERVIEW

The Cloud Telecom/M2M use case is divided in two sections: The M2M and the specific case of CDR analytics. The M2M section focuses on a vehicle telematics provider (VTP) which brings together multiple types of operations and data structures, enabling the platform to accommodate the needs of multiple M2M. The challenges to be addressed by the CPaaS are the ability to process high amounts of traffic, to perform real time analysis of data while presenting the results in interactive dashboards, and to provide a common base to develop rapidly custom made and vertical applications and to facilitate the migration of a subset of the BD to another PaaS if needed. The CDR analytics focuses on pricing simulation and margin analysis, aiming to rapidly execute several pricing scenarios on several traffic patterns. The challenge for the CPaaS is the need for decision making over large amounts of data which requires a scalable solution.

The use of the CPaaS technology will allow the development of a flexible framework, which could be easily adapted to customer needs and specific environment and requirements and, at the same time, is designed from the scratch to be deployed in a cloud environment. The different types of data will be stored in the most appropriate data stores, but they will still have to be joined with one another and the operations that span several data stores have to be done atomically, something that CPaaS will provide and facilitate with its common query language and holistic transactions.

The Media Planning use case objective is to analyze the data generated by social platforms like Facebook, Whatsapp, Twitter, etc._These social platforms produce huge amount of data that is interesting for marketing purposes but, due to the high volume of data that need to be analyzed, traditional technologies are not sufficient. CPaaS will enable the natural integration of different sources, both with respect to coherence for the underlying different data stores and to programmer's point of view regarding the common query language. The sources will be stored in different repositories - key-value (lists of comments), relational (basic data from users) and graph (the links between those data types). The main advantage will emerge from the transactional capabilities of CPaaS, which will allow the updates on the multiple repositories to be performed atomically, preserving the coherence of the data across data platforms at all times. Additionally, CPaaS will save time from copying or translating big amounts of data from one repository to another and thus increase the efficiency of our application. This will be important to manage the real time changes introduced by the users, in such a way that all the answers will be feasible in real time.

While there are multiple initiatives to analyze social media, news or blogging data, there aren't any solutions that exploit an integrated view of multiple data sources in different formats. Integration is also very expensive in a highly dynamic scenario as micro-blogging or on-line news. With CPaaS it will be possible to build a real platform for media analysis, which queries the original data sources without the requirement of data integration. Also, it will reduce the effort of media planners to build queries across multiple data sources.

In the use case of "Real-Time Network Performance Analysis in a Telco Environment", the objective is to detect network problems before any degradation or unavailability of services occur, by actively supervising it. However, monitoring the whole network implies analyzing big amounts of data in real-time and the current deployed solution does not provide the required degree of scalability that can be found in cloud environments. The plan is to use CPaaS platform to provide a rich Platform-as-a-Service that supports several data stores accessible via a uniform programming model and language. The platform will monitor the underlying resources being used by each virtual machine, scaling up and down intelligently. This will enable our developers to focus more in development and improvement of our business applications instead of reconfigurations and management of the platform back-end tiers. At the same time, CPaaS will provide traceability of performance bottlenecks and debugging in applications.

The performance analysis application will take advantage of online processing provided by CPaaS in order to provide real-time analytics. Second, the application will be enormously simplified since applications will be written based on queries that are declarative as opposed to the current programmatic map-reduce jobs that take long to be written and require skilful programmers, which will enable to write queries on the fly. Third, thanks to the unified framework and holistic transactions, CPaaS can play the role of both production system and data warehouse/big data analytics platform, since it will be able to handle extremely large OLTP loads (in the range of 100s of thousands of update transactions per second), but will also support multiple data stores with a rich variety of functionality enabling to perform all kinds of query processing.

The Bibliographic Search use case will aim at collecting a feed of tweets about science, filtered and enriched with the help of the used ontologies. These tweets will be automatically linked with European projects (through Cordis), bibliographies (through DBLP, PubMed) and patents (through Wipo) related to the search keywords or the extracted keywords from the feeds. It will also find relationships and social metrics of scientists through the bibliographies or through the social information obtained from twitter feeds. In order to provide those functionalities, the project will use different data store technologies and queries that blend the technologies created in CPaaS. On the transactional side, the feed from Twitter will be used to synchronize the data from the repositories and provide accurate and timely answers to the users. From the language point of view, a single point of entry will be used, easing the application developers' task to create new functionalities from different data stores put together. CPaaS will be used to remove the necessity to integrate data from multiple data sources into the same data store by using a common query language. This allows the relationships between authors and reputation to be obtained in runtime from the graph data store, which will be used to join information from other data stores.

After analyzing the uses cases, we can conclude that each type of data store to be used in CPaaS solves a particular set of data requirements. In short:

- Relational data stores are used to store data that has several related properties which are required to be indexed/partitioned;
- Columnar data stores are used to store mainly data that needs to be aggregated, i.e. to calculate sums, averages, etc., since these types of operations are usually limited to a column or a small subset of them;
- Key-value data stores are mainly used to store raw or rated data, i.e. data that is made of single values that must be accessed individually through a key;
- In-memory data stores are being used for their benefits regarding reading and writing of data. Therefore, this type of data store technology is appropriate for data analysis, such as what-if analysis.

3 Cloud Telecom/M2M Use Case – CDR Analytics

3.1 Use Case Outline

The Cloud Telecom/M2M use case - CDR analytics focuses on pricing simulation and margin analysis, aiming to rapidly execute several pricing scenarios (plans) on several usage patterns. The analysis of this data could then be used in the following cases:

Per customer:

- Find customers with the biggest delta (distance) from their best plan;
- Find the best plan for each customer;
- Find the best offers that bring the same revenue.

Accumulative:

- Find average distance of the invoices from the corresponding best plan;
- Find average distance of the invoices from the corresponding best plan per plan;
- Perform segmentation of customers based on usage, what they pay, etc.

The challenge for the CPaaS is the need for decision making over large amounts of data which requires a scalable solution. Neurocom has already commenced a project, exercising price simulation over MapReduce (see Figure 1).

Price simulation is dependent on the rating engine performance upon multiple rate plans. Rating is the application that performs rating of CDRs based on a single contract scenario which includes the base plan, the addon(s), the discount plan(s) and the promo(s). Price simulation uses the rating engine to apply what-if scenarios. It consists of two phases: rating and post-processing. During the rating phase it takes as input raw files of CDRs and rates them using different combinatory plans. Plans are stored in a SQL data store and loaded at the beginning of execution into an in-memory data structure. The output of the rating phase is raw files of CDRs rated based on the different plans. It finally exports many rated CDRs for each input CDR. After the rating phase, it stores in memory aggregated results. Using this data, it executes a post-processing phase and computes final results to report. These include more aggregations, selections based on thresholds, possible discounts etc.

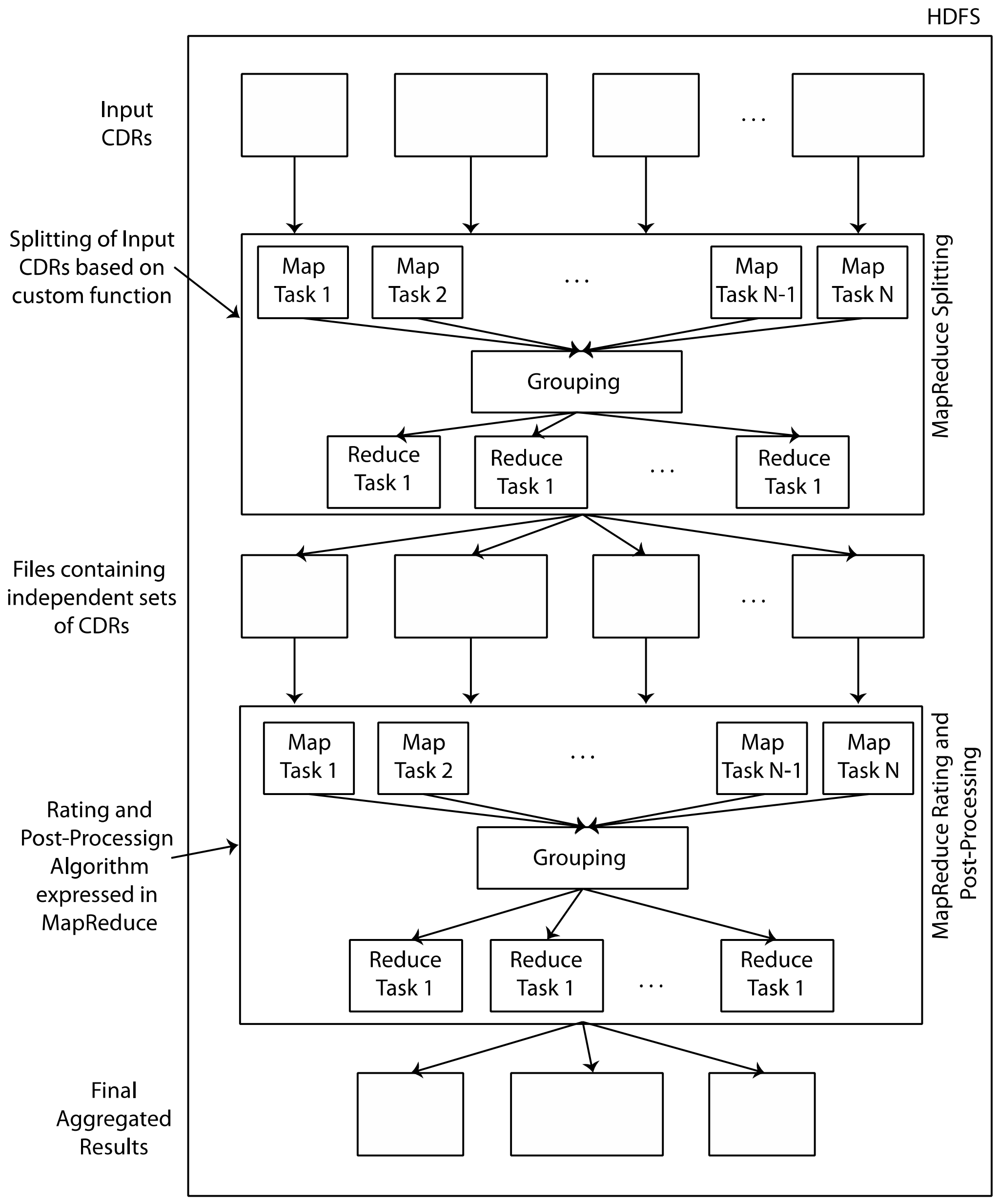


Figure 1: Cloud Price Simulation over MapReduce.

The current implementation of price simulation faces two main challenges:

- **Scaling:** It is a big-data application in that it needs to process large amounts of records and plans and, depending on the use-case, provide results at real-time or near real-time;
- **Extensibility:** It needs to adapt to different use-cases as business requirements change. Therefore the functionality provided by the post-processing phase and to some extend the rating phase itself need to change over time and for different applications. Currently, both phases are written in Java, making even small extensions both error-prone and time-consuming. Instead, implementing each phase in a high-level framework, such as a query language or MapReduce can significantly reduce the time and effort required for adapting the application to new use cases.

For further information about the business orientation and the functional description refer to Section 9.1 in Appendix A.

3.2 CDR Analytics: Data Requirements & Limitations

The currently developed price simulation and margin analysis applications by Neurocom use relational data stores (Oracle and PostgreSQL) as a main data storages.

- Under specific, but realistic conditions, the margin analysis functions are not scalable. For instance, the following set of figures are the elements of a use case requested:Number of contract numbers: 500,000;
- Period of time: 3 months;
- Avg. number of invoice lines per invoice: 10;
- Number of combinations: 5000;
- **→ 25 billion invoice records need to be processed.**

For this scenario, it is necessary to make decisions on a **25-billion line table** that exceeds size of 1.2 Terabytes. Current solution is not scalable and cannot be scalable, with relational data stores, especially if results should be available almost real-time through a web interface.

Numbers can become even higher when:

- Number of combinations grows very quickly due to the fact that a customer may have multiple addons and promos/discounts
- Number of contract numbers (MSISDNs) can also be quite higher for big telcos
- Usage data need to be loaded for more billing cycles, in order to capture seasonal needs in a more representative way
- Telcos add more services to attract more customers or increase the ARPU.

The schema below (Figure 2) is an abstract data-flow diagram aiming at depicting the basic steps of margin analysis. Some rough estimations of time required for the example mentioned above when using relational data stores have been included too:

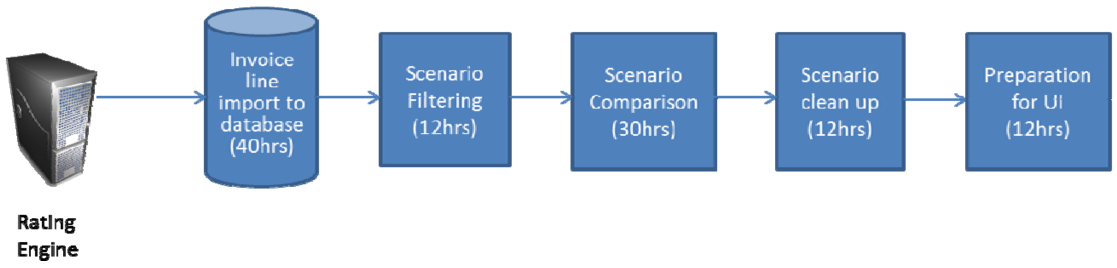

Rating Engine
Invoice line import to database (40hrs)
Scenario Filtering (12hrs)
Scenario Comparison (30hrs)
Scenario clean up (12hrs)
Preparation for UI (12hrs)

Figure 2: Margin Analysis Flow.

More specifically:

- Rating Engine processes the CDRs and generates the invoice lines for each contract number and each alternative scenario;
- Invoice lines are imported into the relational data store for analysis;
- For each contract number, irrelevant scenarios are identified and filtered. Whether a scenario is useful or not is determined by the business logic. E.g. it does not make sense to offer a scenario that includes a data addon if the user does not generate data usage;
- For each contract number and customer, all remaining scenarios are compared to each other and to what the customer currently pays to identify the ones that produce a bill amount close to the targets (e.g. best, less expensive by 5%, 10%, etc., more expensive by 5% etc.);

- For each contract number, scenarios that do not meet any of the targets are cleaned up;
- The final step prepares the data for use by the user interface (UI) so that the amount of calculations done, when the user requests a report, is minimal and the system responds as fast as possible.

The requirements that affect data stores are divided into the following clusters:

Table 1: List of Requirement Clusters – CDR analytics case.

Requirement Cluster (RC)	Name	Brief Description
RC-A	Technological	Requirements regarding the support of several technologies.
RC-B	General	Requirements regarding the system's general requirements.
RC-C	Delays	Requirements regarding maximum delay
RC-D	Throughputs	Requirements regarding the minimum throughput across the system.

However, within the CDR analytics use case, our basic requirements fall into the RC-C category, being related to system response metrics within various queries and tasks. Based on the functional requirements listed above and taking into account technical and non-technical parameters, the following table lists the performance indicators we would like to review. For each of them, the acceptable and target metrics have been included:

Table 2: List of Delay Requirements – CDR analytics case.

Delay Requirements	Name	Brief Description	Acceptable Metric	Target Metric
ReqC.01	BILLING CYCLE PROCESS	Import, analyze and prepare the data for one billing cycle	60 mins	30 mins
ReqC.02	ONE CONTRACT LOAD DATA	Retrieve real usage, cost and billing data for one contract number	15 secs	5 secs
ReqC.03	ONE CUSTOMER LOAD DATA	Retrieve real usage, cost and billing data for a customer account of 1000 active contracts	120 secs	60 secs
ReqC.04	LOAD ONE NUMBER BEST SCENARIO DATA	Retrieve data for best alternative scenario for one contract number, while user is dynamically changing scenario parameters, assuming decision making on ~5000 scenarios and 3 billing cycles	30 secs	10 secs
ReqC.05	LOAD ONE CUSTOMER BEST SCENARIO DATA	Retrieve data for best alternative scenario for one customer account of 1000 active contracts, while user is dynamically changing scenario parameters, assuming	4 mins	2 mins

		decision making on ~5000 scenarios and 3 billing cycles		
ReqC.06	LOAD ONE NUMBER TARGET SCENARIO DATA	Retrieve data for a target alternative scenario (e.g. 10% less expensive than real billed amounts) for one contract number, while user is dynamically changing scenario parameters, assuming decision making on ~5000 scenarios and 3 billing cycles	40 secs	15 secs
ReqC.07	LOAD ONE CUSTOMER TARGET SCENARIO DATA	Retrieve data for a target alternative scenario (e.g. 10% less expensive than real billed amounts) for one customer account of 1000 active contracts, while user is dynamically changing scenario parameters, assuming decision making on ~5000 scenarios and 3 billing cycles	5 mins	2.5 mins

3.2.1 Addressing The Appropriate Data Store

Fitting the available data store technology selection to our CDR analytics use case needs a more careful insight into the use case design, which is an ongoing task of CPaaS project. However, there are certain facts related to structural and size-related patterns, which provide with hints and 1st priority options to pick up:

- **Column-oriented organizations** (such as MonetDB) are more efficient when an aggregate needs to be computed over many rows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can be faster than reading all data. Henceforth, margin analysis upon huge amount of invoice lines requires rapid identification of relevant data lines and efficient analytics performed on the usage data associated;
- **Key-value stores** for the storage of the usage records before (e.g. raw data) and after processing (e.g. rated data). It is expected that key-value stores will be used for the storage of individual usage records while aggregates created on them and other extracted information will be stored using the column-oriented data stores;
- The **In-Memory data store** solution is promising for analytics on CDRs and especially for what-if analysis on variant (and interactively built by the user after applying dynamically changes) scenarios. Tariff simulation at a given number or group of numbers (corresponding to one customer) generates hundreds (or even thousands) of scenarios, where each scenario is a combination of given baseplans, addons and discounts. Then each number is "priced" against each scenario, and analytics measuring revenues for each case (risk analysis) are derived from that. A multi-core in-memory engine could help turning calculations that used to take several hours into something interactive. But in the case of scenario based simulations, we could use a vector-based approach combined with in-memory solution to make aggregation even faster, and quite compact in memory.

Deeper investigation into the necessary dataset which are processed, their flow and the intermediate data produced during tariff simulation and margin analysis, will show the best fitting combination of data stores to datasets, which will be employed through CPaaS.

3.3 Development Process

This use case needs **reengineering** an existing application, which provides tariff simulation and margin analysis. We assume the following work components:

- **Design** - Aims at designing in higher details the mixture of data stores that are judged as necessary in order to satisfy the listed requirements, as well as the transactions that need to be performed in order to realize functionality;
- **Loading/Rating/Off-loading** - Refers to the group of tasks that result in calculating the invoices that will feed the margin analysis procedure. It takes into account the customer segment under consideration, loads related usage and scenario data, rates usage according to those scenarios, does billing and finally delivers the invoices. Final data is offloaded and passed to margin analysis procedure for further analysis;
- **Query Implementation** - The next step is to create the workflow, queries and operations which are necessary to analyze the invoice data efficiently. All queries will be tested; their performance will be measured for validation purposes before they are linked to visual interface;
- **Visualization** - It visualizes the results of the analytical queries mentioned above. May require integrating them into or adapting existing screen interfaces.

Some of the above work-components (especially, "Design" and "Query implementation") will be iterated based on requirements to be met, measurements and findings during testing. We assume that significant changes and reengineering will be required for:

- The data schema and store technologies;
- The data processing functions.

While the following components will remain relatively unchanged and will only be adjusted to become compatible with the components that will change considerably:

- The rating engine;
- The visualization components.

3.4 Guidelines For The Validation Phase

Validation will be done in a systematic way so that to check that the new implementation of the use case:

- Functions well, is correct and complete, comparing always with the previous implementation;
- Performs within acceptable levels, or even reaching the goals set and;
- Has utilized the best mixture of technologies from what is available through CPaaS.

It must seek ways to ensure that the use case is implemented in a way to remain:

- **Correct:** Previous architecture (already and extensively tested) delivers specific output for a given input of usage. Same data will be given as input to new implementation;

- **Efficient:** Measuring performance will show at which level the desirable (either at the level of acceptance or at the level of ultimate target) metrics are reached. Metrics are given in Section 3.2;
- **Scalable:** Moreover, stress-testing should be applied. Input sizes should be modified (either number of scenarios, billing cycles, customers or usage levels or all possible combinations) and system behavior will be assessed into different input levels. Finally, it should be clear how system must scale in order for the use case to remain acceptable as load increases;
- **Optimal Use of Technologies:** Combining validation results with X-ray platform findings will drive decisions towards improving and/or correcting the CPaaS platform or selecting another design choice for data storing and analytics.

4 Cloud Telecom/M2M Use Case – Machine-To-Machine

4.1 Use Case Outline

The Cloud Telecom/M2M use - M2M focuses on a VTP which brings together multiple types of operations and data structures, enabling the platform to accommodate the needs of multiple M2M. It is expected that the PaaS will store and distribute securely information produced by a variety of devices. The use case we are defining should be based on an architecture where messages are decoded and routed to the back-end logic process (as in the architecture drawing in later subsection 4.2) where they are processed and inserted in the PaaS. A web application will then access the PaaS and will provide services like management of users and devices, alerts, representation of entity behaviour, etc. As the amount of devices increases every year, we need to improve this infrastructure in order to support scalability, high availability, stability and performances.

The challenges to be addressed by CPaaS are the ability to process high amounts of traffic, to perform real time analysis of data while presenting the results in interactive dashboards, as well as to provide a common base to develop rapidly custom made and vertical applications and to facilitate the migration of a subset of the BD to another PaaS if needed.

Our current infrastructure is based on these building blocks:

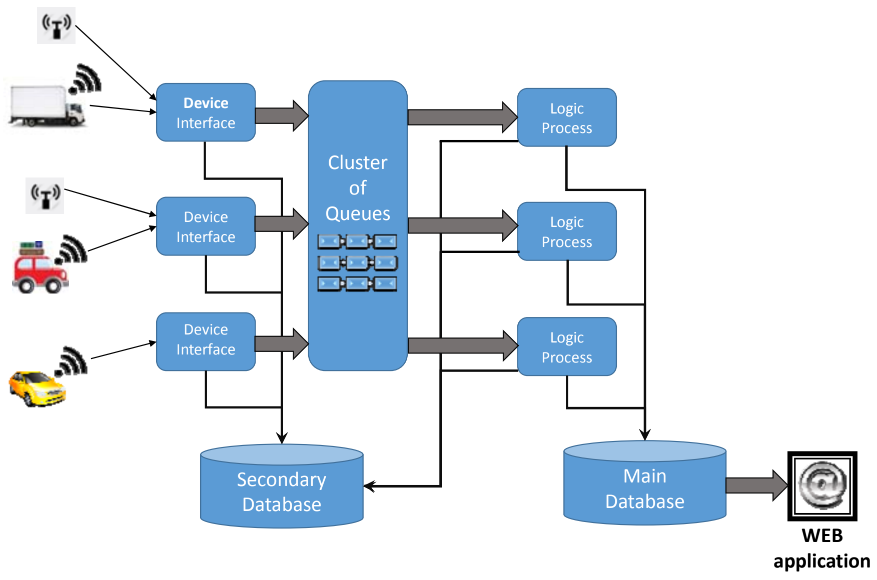


Figure 3: Building Blocks for the Vehicle Telematics M2M Case.

Figure 3 shows the interaction between each block, namely that:

- Vehicles equipped with devices (GPRS and GPS) or sensors communicate to a TCP application server (Device Interface);
- Device interface application handles messages:
  - Acknowledges the messages based on specific protocol definition;
  - Stores messages in a secondary DB (SQL Server) for specific consolidation;
  - Puts the result in queues managed by a cluster of brokers.

- A second application server (Logic Process) takes messages from the queues:
    - Decodes the messages based on specific protocol;
    - Handle some logic based on the secondary data store;
    - Stores positions in the main DB (SQL Server), creates notifications, alarms, etc.

The main DB is also used by the web application in order to deliver information to the end user. For further information about the business orientation and the functional description refer to Section 9.1.1 in Appendix A.

4.2 M2M: Data Requirements & Limitations

During the use case, we would like to validate the following points:

- High traffic: currently the platform handles millions of messages with a peak of 120.000 per hour. The goal is to be able to grow to reach a doubling of the traffic in one year, or even 3 times more to satisfy upcoming requirements within forthcoming years;
- Performances: different points of view must be taken into account:
    - The handling, treatment of a message must be done between 1ms and 4ms;
    - The overall performances must not be impacted by the increase of the connected devices. The plan is to have a linear increase of 10000 connected devices by year.
- Real Time analysis and increased interactivity with the user: fast availability of consolidated data's in order to create dashboards is very important. The platform must be able to analyse and consolidate, in real time, data's like fuel consumption, cumulated distance, cumulated hours, driver scoring, … : data aggregation by hours, days, weeks, months, years, … trends computation
  The platform must also be able to generate notifications, alerts based on different patterns:
    - Threshold reach under conditions (working hours, based on geographical location/zone, …);
    - Entering or leaving a geographical zone
    - Notification based on an input (on/off) under conditions.
- Flexibility to be able to develop rapidly tailor made and vertical application reusing a common base (the PaaS);
- Flexibility of the reports and dashboard which should be adaptable on-line and refreshed in real-time.

Migration: it could be possible to migrate a subset of the DB to another PaaS

We divide the requirements into the following clusters:

Table 3: The List of Requirement Clusters.

Requirement Cluster (RC)	Name	Brief Description
RC-A	Technological	Requirements regarding the support of several technologies.
RC-B	General	Requirements regarding the system's general requirements.
RC-C	Delays	Requirements regarding the maximum delay across the

		system.
RC-D	Throughputs	Requirements regarding the minimum throughput across the system.

The tables below describe for each requirement cluster the requirements which are strongly related to data stores:

Table 4: List of Technological Requirements – M2M Case.

Technological Requirements	Name	Brief Description
ReqA.01	SYSTEM_FLEXIBILITY	The system must be providing the flexibility and ability to develop rapidly tailor made and vertical application reusing a common base (the PaaS).

Table 5: List of General Requirements.

General Requirements	Name	Brief Description
ReqB.01	LINEAR SCALABLE	The overall performances must not be impacted by the increase of the connected devices. It is expected to have a linear increase of 10000 connected devices by year
ReqB.02	MULTIPLE ALARM CREATION	The platform must also be able to generate notifications, alerts based on different patterns: • Threshold reach under conditions (working hours, based on geographical location/zone, …) • Entering or leaving a geographical zone • Notification based on an input (on/off) under conditions

Table 6: List of Delay Requirements – M2M Case.

Delay Requirements	Name	Brief Description	Acceptable Metric	Target Metric
ReqC.01	MESSAGE HANDLING TIME	The handling, treatment of a message must be done between 1ms and 4ms	4ms	1ms
ReqC.02	DASHBOARD CREATION UPDATE	Availability of consolidated data's in order to create dashboards is very important. The platform must be able to analyse and consolidate, in real	~3 min	< 1 min

		time, data's like fuel consumption, cumulated distance, cumulated hours, driver scoring, … : data aggregation by hours, days, weeks, months, years, … trends computation		

Table 7: List of Throughput Requirements – M2M Case.

Throughput Requirements	Name	Brief Description	Acceptable Metric	Target Metric
ReqD.01	MESSAGE HANDLING THROUGHPUT	The platform should be able to handle more than 80.000.000 messages per month. Day peaks should exceed 4.000.000 per day and 240.000 per hour.	240.000 msgs/hour	360.000 msgs/hour

4.2.1 Addressing The Appropriate Data Store

In the VTP M2M use case, it is challenging to receive and process in real-time a large – and increasing - amount of vehicle data and at the same time access and analyse a large dataset of archive data containing all sensed data. Apart from the real-time processing which is necessary for alarm production and the presentation of real-time views, a big, immutable dataset is created which needs to be accessed when historic views are required to be extracted.

It is clear that a big dataset is produced, which is built from the append-only transactions. No updating is necessary. All data are historic and are accessed only for statistic reasons, for aggregate creation and when past views need to be created. Therefore, a **columnar store** is strongly considered for the analytical part, while a **Hadoop distributed data store** will most favorable to support the archive demands.

The continuous reception, queuing and processing of vehicle data indicated that a **complex event processing engine (CEP)** should be employed. CEP nodes should interact with memory data so that processing can be accomplished in real-time. All necessary processing for real-time decision should avoid accessing archive and analytical data stores. An **in-memory data store** should be able to accommodate the active workload which is required in order to have the real-time views and the realization of real-time processing.

Deeper investigation into the necessary dataset which are processed, their flow and the intermediate data produced during data reception and processing will show the best fitting combination of data stores to datasets, which will be employed through CPaaS. At a glance, we can easily infer that this use case deals with two different environments: the high-speed one and the batch one; however, there are certain functionalities that need to combine functionality from both contexts.

4.3 Development Process

The case of the VTP needs **reengineering** an existing application which provides today VTP's functions and services:

- **Design -** Aims at designing in higher details the mixture of data stores and the necessary transactions in order to achieve the desirable metrics in terms of performance. Moreover, the design phase might redistribute the functions between device interface and logic component responsibilities, in a different way than the current allocation;
- **Query Implementation -** The next step is to create the workflow, queries and operations which are necessary to gather and analyse vehicle data. Query implementation job is split into device interface component and logic component adjustments related to secondary and main data store accordingly;
- **Visualization -** This is the integration part of the work, so that existing application user interfaces will present the results produced.

Design and Implementation of queries will be revisited based upon findings of unit testing as well as larger scale validation tests.

4.4 Guidelines For The Validation Phase

Validation will be done under realistic conditions so that the efficiency of the new architecture can be reliably assessed.

For validation purposes, a vehicle data generator should be implemented so that the system will be fed with measurements and real-time responsiveness will be evaluated. All unit testing measurements should be prepared and recorded so that we can finally have a well-justified picture on the level of achievement.

Validation will be organized into different phases (or scenarios). Each scenario should feature different workloads in terms of a) volume, b) data production speed and c) level of complex queries required.

Validation will aim to show that the use case implemented through CPaaS, will remain **correct** , while at the same time it will perform **efficiently,** either reaching the target metrics or at least within the acceptable metric level range. Measurements of applications' responsiveness to continuously increased data volumes (due to vehicle increase) will be necessary to show the level of **scalability** reached. The use case provider needs to be aware of sizing plans and needs, in response to higher emerging growth. While, the use of CPaaS is expected to exhibit improved performance and capabilities for the use case, it is necessary to assess how and whether each technology responds adequately to specific data access pattern (DAP). Therefore, insight views of CPaaS performing within the use case, obtained by the X-ray application platform, will prove whether **optimal use of technologies** is made. Refinement proposals might be proposed either to the way CPaaS is used or to the way data store access is implemented through CPaaS, addressing the need to changes in the core transactional layer of CPaaS.

5 MEDIA PLANNING USE CASE

5.1 Use Case Outline

The Media Planning use case objective is to analyze the data generated by social platforms like Facebook, Whatsapp, Twitter, etc. These social platforms produce a huge amount of data that is valuable for marketing purposes. However, due to the high volume of data that need to be analyzed, traditional technologies are not sufficient. In the most popular social networks, the analysis of registered messages allows us to infer how the public perceives a brand and then, to perform corrective actions through the social networks to change it, such as designing effective marketing campaigns, analyzing target audiences, classifying it by roles, or predicting the impact of a given communication with the most influencers in a given topic/community.

CPaaS will enable the natural integration of different sources, both with respect to coherence for the underlying different data stores and to programmer's point of view regarding the common query language. The main advantage will emerge from the transactional capabilities of CPaaS, which will allow the updates on the multiple repositories to be performed atomically, preserving the coherence of the data across data platforms at all times. Additionally, CPaaS will save time from copying or translating big amounts of data from one repository to another and thus increase the efficiency of our application. This will be important to manage the real time changes introduced by the users, in such a way that all the answers will be feasible in real time.

The following diagram summarizes the physical and logical view of the system's modules.

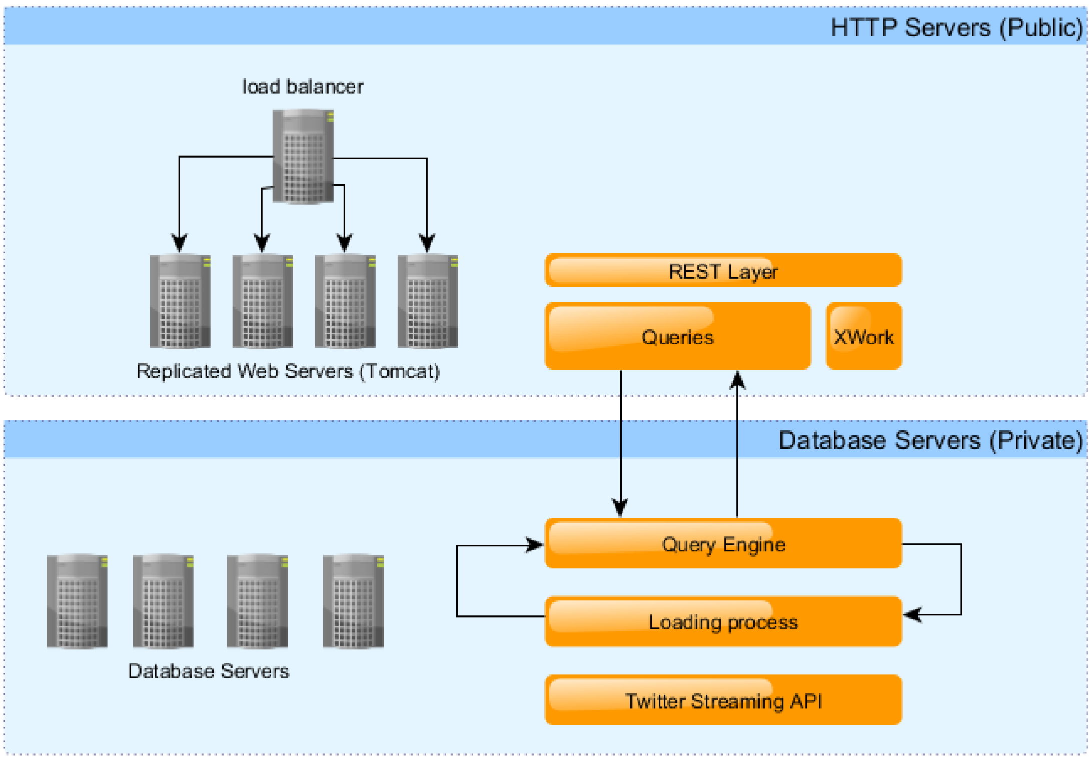


Figure 4: Media Planning Use Case: Physical and Logical View.

Namely, these modules are responsible for:

- **Loader**: Performs a real time data collection from twitter. This component might use storm to provide a distributed and incremental real-time computation system.

The loader will store the basic information related with a document and will index it in the most appropriate way to ensure a high performance in the analytical part;

- **Analytical layer**: It performs analytical queries using the common query engine. These queries are recommendation engines to resolve communities and influencers. We will use the XWork / Struts2 because it is a rich Model View Controller (MVC) framework with an IoC container;
- **REST layer**: It exposes the analytical layer to the web through a REST API. Multiple servers with the same REST api will be running and a single load balancer will distribute the requests.

For further information about the business orientation and the functional description refer to Section 9.2.1 in Appendix A.

5.2 Media Planning: Data Requirements & Limitations

The current developed Media Planning applications by Sparsity-Technologies use DEX (a.k.a. Sparksee) as main data storage: a high performance graph data store that runs on a single machine. DEX needs the whole data store to fit in memory to get the maximum performance.

Therefore, the scope requirements of this project is to change the data store to get a better solution combining the most appropriate technologies to perform the analytical queries without losing performance in loading time.

The most critical requirements that affect to data store technologies are:

Table 8: Media Planning Use Case: Critical Requirements – Media Planning Case.

Main Requirements	Name	Brief Description	Acceptable metric	Target Metric
Req.01	MAXIMUM_DOCS_PER_DAY	The system must be able to load 1 million of documents in less than 12 hours.	12hs	< 12hrs
Req.02	MAXIMUM_TIME_PER_QUERY	The system must be able to solve all analytical queries in less than 1 minute.	1 minute	2 sec.

5.2.1 Addressing The Appropriate Data Store

We have selected a set of data stores to ensure a high throughput in the selected media planning operations. These operations are:

- **OP#1: Retrieve the most influencers;**
- **OP#2: Retrieve communities;**
- **OP#3: Add new documents.**

Media Planning clients considered it was necessary to have a generic schema because documents are downloaded from different communication channels, and each of them has its own format. The contents of the designed generic schema are documents, entities, tags, topics, locations, people, groups, references, copies, participates, and many others. Figure 5 summarizes the current generic schema.

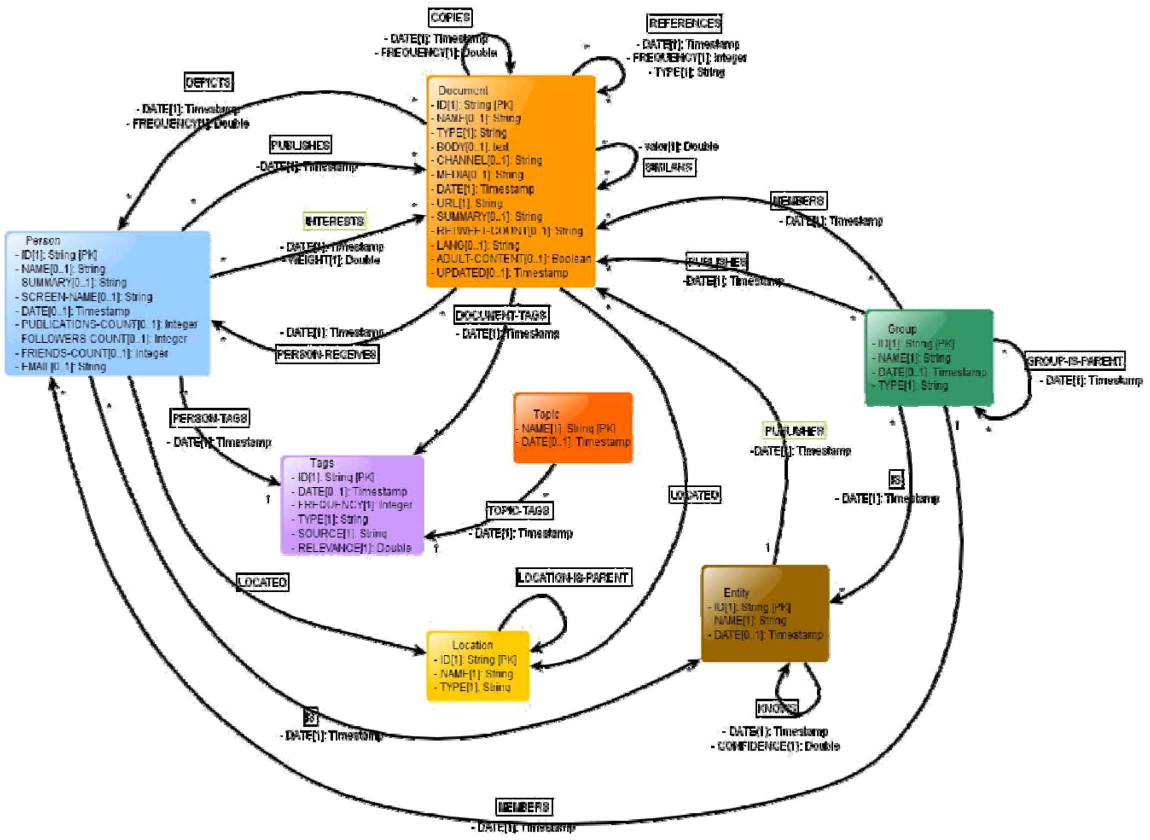


Figure 5: Media Planning Use Case: Generic Schema.

To implement an efficient behavior, this schema has been extended to index documents, entities and communities by keywords. Let's see the detail of each proposed operation for this use case and the desired data stores for each scenario.

OP#1: Retrieve the most influencers

Description: The 10 most influencers (entities) about a list of topics, sorted by the sum their probabilities in each topic. For each entity, the system must show:

- The number of (potential) influenced entities;
- The maximum propagation depth;
- All their basic fields: id, name, date;
- The last published document: id, text, date, #propagations, channel, #copies, #references.

Currently, to solve this query efficiently, we need to extend the generic media planning schema indexing the entities by keyword through the INFLUENCES_ABOUT relationship. Moreover, we need to store the relevance of a given keyword for the indexed element.

Another extra information is stored to perform efficiently the loading process to compute INFLUENCES_ABOUT. These relationships are:

- INFERRED_KEY (weight: Double): Indexes documents by keyword;
- PROPAGATES_TO (weight: Double): Stores which document is a propagation of another one (with a given probability);

- INFLUENCES_TO (weight: Double): Stores which entity influences to another one (with a given probability).

The following picture shows the extended schema with the explained relationships.

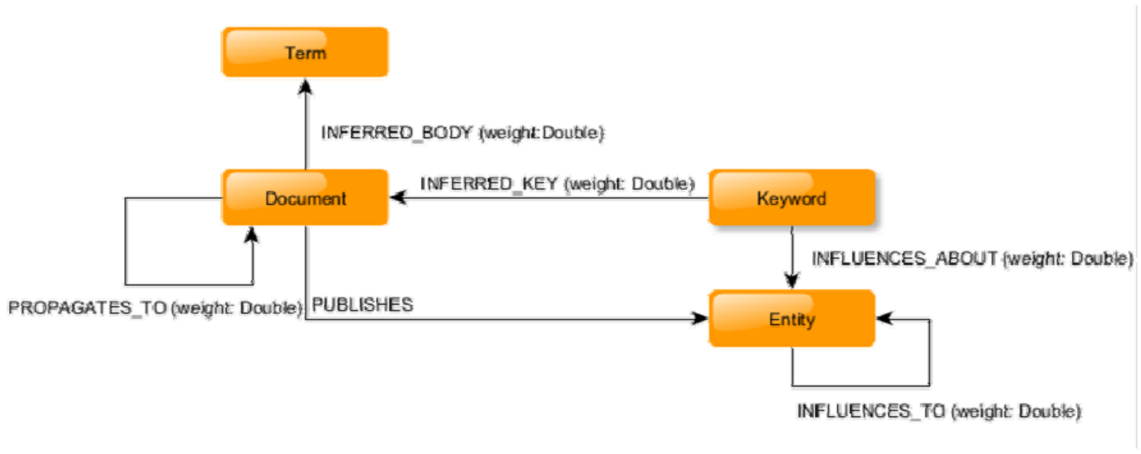

Term
INFERRED_BODY (weight:Double)
Document
INFERRED_KEY (weight: Double)
Keyword
INFLUENCES_ABOUT (weight: Double)
PROPAGATES_TO (weight: Double)
PUBLISHES
Entity
INFLUENCES_TO (weight: Double)

Figure 6: Media Planning Use Case: Extended Schema to retrieve influencers.

We suggest to apply this extended schema with the following data stores:

Table 9: List of data stores suggested to apply the extended schema for OP#1 – Media Planning Case.

Type	Data store	Resolves	Structure
INFLUENCES_ABOUT (weight: Double)	Key Value	The most influencers of a given keyword	Key: keyword Value: SortedSet<{entity.id, weight}>
INFLUENCES_TO (weight:Double)	Graph	The maximum propagation depth (DFS) and the number of influenced entities	Nodes: Entity (id) Edges: INFLUENCES_TO (weight: Double)
PUBLISHES Entity (id, name, date)	Relational Database Management System (RDBMS)	The documents of an author and the basic field information of an author.	2 tables: PUBLISHES (enity.id FOREIGN_KEY TO ENTITIES.id, doc.id), ENTITIES (id PRIMARY_KEY, name, date)
Document (id, date, channel, text)	Document oriented	The last document of an influencer with the basic field information of the document.	Document (id, date, channel, text)

An influence between two entities is produced because they have documents related between them through other relationships such as COPIES, REFERENCES, PROPAGATES or DEPICTS. For example, in Twitter, a retweet is produced when one user copies a tweet from another user and publishes it in his timeline, so the second user influences to the first one.

Thus, it is necessary to store previously some basic relationships to infer influences.

Table 10: List of basic relations to store for OP#1 – Media Planning Case.

Type	Data store	Resolves	Structure
COPIES	Graph	All documents that are exactly the same as another ones (ex. Retweets)	Relationship
REFERENCES	Graph	All documents that are referenced by another one.	Relationship
PROPAGATES (weight)	Graph	All documents that are propagations made by other documents	Relationship
DEPICTS	Graph	All entities referenced from a document.	Relationship

<u>OP#2: Retrieve communities</u>

Description: The 10 biggest communities for a given set of keywords. For each community, the system must return the 20 most influencers inside the community by microblogging documents. For each member, the system must show:

- The number of (potential) influenced entities inside the community;
- The maximum propagation depth;
- All their basic fields: id, name and date;
- The last microblogging published documents: id, text, date, number of propagations, channel, number of copies and number of references.

Currently, to solve this query efficiently, like we have explained in the previous one, we need to extend the generic media planning schema storing communities and indexing them by keyword through a new node type called CLASSIFIES and new relationships. Moreover, we need to store the members of a given community and which influences they provoke inside the community and which documents have provoked a given influence.

The extended schema is as follows:

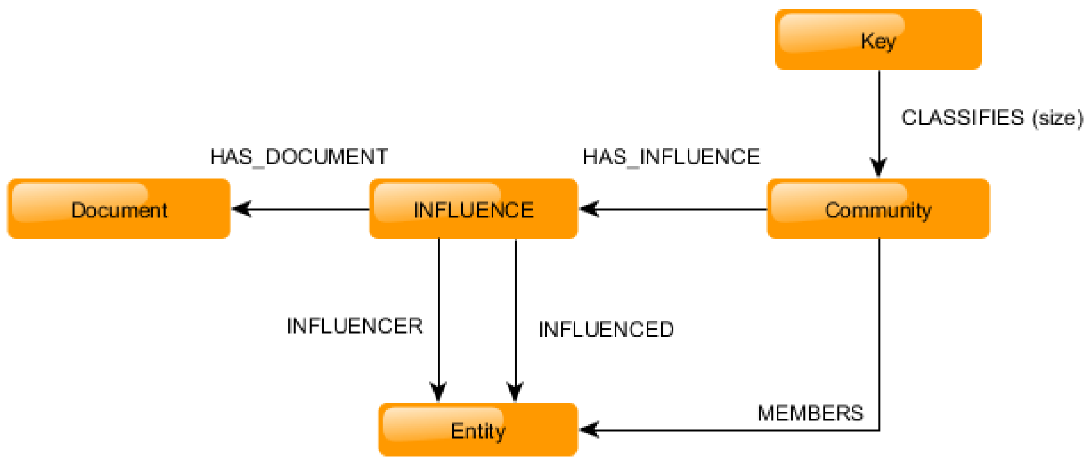


Figure 7: Media Planning Use Case: Extended Schema to retrieve communities.

We suggest to store this extended schema with the following data stores:

Table 11: List of data stores suggested to apply the extended schema for OP#2 – Media Planning Case.

Type	Data store	Resolves	Structure
INFLUENCE, MEMBERS, INFLUENCER, INFLUENCED, HAS_INFLUENCE, HAS_DOCUMENT	Graph	The most influencers within a community for a given set of keywords	**Nodes:** Influence, Community, Document **Relationships**: MEMBERS, INFLUENCER, INFLUENCED, HAS_INFLUENCE, HAS_DOCUMENT
CLASSIFIES (size)	Key value	The 10 biggest communities by keyword	**Key:** Key **Value:** SortedSet<{community.id, size}>
PUBLISHES Entity (id, name, date)	RDBMS	The document of a given influencer	2 tables: PUBLISHES (enity.id FOREIGN_KEY TO ENTITIES.id, doc.id), ENTITIES (id PRIMARY_KEY, name, date)
Document (id, date, channel, text)	Document oriented	The last microblogging document of an influencer with the basic field information of the	Document (id, date, channel, text)

		document. Document filtering by channel	

OP#3: Add new documents

Description: Add new documents produced by twitter as soon as they are published. Those documents that contain less than 3 words must be omitted. Several types are affected by this event: Document, PROPAGATES, COPIES, REFERENCES, Entity, Key, INFLUENCES_TO, INFLUENCES_ABOUT, COMMUNITY, CLASSIFIES and INFLUENCE.

No extension into the schema is needed to apply this operation, but a set of complex processes are involved. These processes are:

- Documents tokenization;
- Infer document propagations though the analysis of the similarity of their texts;
- Infer the entity influences through the detected propagations and explicit relationships such as COPIES (retweets), REFERENCES and DEPICTS;
- Infer the communities by term;
- Infer the influencers by term.

From an architectural point view, this operation, which is a complex ETL process, should be performed with Storm, a distributed real-time computation system.

5.3 Development Process

This use case implies a **reengineering** of an existing application which provides a set of analytical queries for community managers. These queries include communities and influencers. We assume the following work packages:

Loading package

Our first forthcoming work would be create a design to load tweets in real time using the Twitter API and Storm. We need to ask with subset of our clients data can be used for this use case.

Now, we get a loading process which acts as a daemon that is executed daily and it needs to be changed to apply incremental updates per tweet. However, as some processes needs a minimum set of data to someone realizes changes of the query results, they won't be executed until the minimum set of new data is reached.

Moreover, according our requirements, we need to verify that our system is able to insert 1 million of daily tweets in less than 12 hours.

Queries package

The next step is to design and create the analytical operations (communities and influencers), which needs to offer fast responses (less than one minute) because they will be requested form a visual interface. These queries will be tested from a prompt.

REST package

Finally, the last step is adding an HTTP layer to execute the included analytical queries.

5.4 Guidelines For The Validation Phase

The main goal of this use case is to load tweets in real time and provide two fast analytical queries to improve the effectiveness of the designed marketing campaigns by our clients. The quality indicators, described in detail in 8.2, are correctness, efficiency, scalability and the evaluation of the optimal use of technologies. Moreover, Sparsity-Technologies has a data sample with 1 million of tweets to use for performance purposes.

The system will use many mathematical models to resolve the queries, like the TF-IDF model to measure the relevance of a term in a document. All these mathematical models, which are used to infer relationships such as propagation or influence, have been already validated by Media Planning Group and Acceso, so these models shouldn’t change.

Sparsity-Technologies will design a validation phase using a set of **unitary tests** to assure the queries’ correctness and also the correctness of each of the loading steps. Moreover, we will build an **integration test** which will simulate the Storm behavior with the testing dataset and once all documents had been consumed, the contents of the entire data store will be verified. Moreover, Sparsity-Technologies will use streaming data of twitter to test the system **scalability** and a sample with 1 million of tweets for **efficiency** purposes.

6 Real-Time Network Performance Analysis in a Telco Environment Use Case

6.1 Use Case Outline

The objective of the Real-Time Network Performance Analysis in a Telco Environment use case is to detect network problems before any degradation or unavailability of services occur, by actively supervising it. However, monitoring the whole network implies analyzing big amounts of data in real-time and the current solution does not provide the required degree of scalability that can be found in cloud environments. The existing end-to-end (E2E) system, called Altaia, actively detects deterioration in a network using almost real-time KPIs and key quality indicators (KQIs), finds the cause of performance problems and the produced data is always ready to be analyzed through reports and dashboards to check the network's performance. These reports can be tailored to further analyze the network's performance by creating ad-hoc queries and accessing the data that originated the calculated indicators

The plan is to use CPaaS to provide a rich Platform-as-a-Service that supports several data stores accessible via a uniform programming model and language. The platform will monitor the underlying resources being used by each virtual machine, scaling up and down intelligently. This will enable our developers to focus more in development and improvement of our business applications instead of reconfigurations and management of the platform backend tiers. At the same time, CPaaS will keep the needed traceability of performance bottlenecks and debugging of errors in applications.

The Altaia system (which the current architecture is shown below) is composed of two data store layers, DBN0 and DBN1, which store raw data collected from the network and the calculated KPIs and KQIs, respectively.

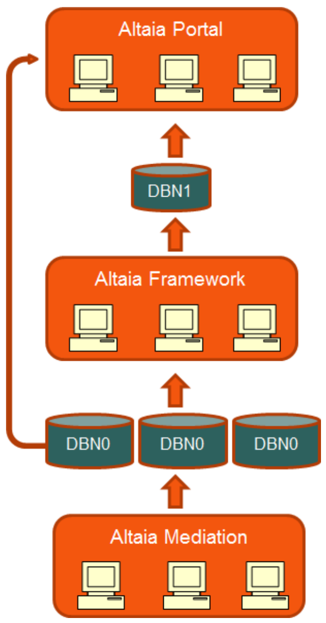


Figure 8: Altaia System Architecture.

The system is also composed from following modules:

- **Altaia Mediation:** The Altaia Mediation module is a scalable and robust abstraction layer that allows the Altaia system to access the data from various sources in the network (nodes, servers etc.), manufacturers and technologies (SNMP, SIP, XML, CLI etc.) in a standard way. The module uses a set of plugins that read the data from each data source, transform it into the normalized formats used by the Altaia system and ensure data quality (e.g. data must comply to a list of possible values of a field and block or correct the values that do not fit). It is also responsible for enriching the data with dimensional information (e.g. use cell ID to define record call the geo-location or infer classification of “on net” or “off net” from the calling and called number), performs the data parsing to identify errors and to log them and is able to perform self-discovery of network entities;
- **Altaia Correlation:** The Altaia Correlation module is under development (some of its functionalities are currently being performed inside the Altaia Framework) and is currently using Esper (CEP engine) to correlate data provided by the Altaia mediation module and assemble records. The event correlation can be used to, for example, infer that calls are being dropped in the network, by correlating multiple call events in a short period of time. Correlated events generate new complex events that are stored in the DBN0 and can be used by the Altaia Framework and Portal. The generated complex events can also be fed directly to the Altaia Framework to generate alarms;
- **Altaia Framework:** The Altaia Framework module is the point where all the KPIs and KQI are produced and transformed for several types of analyses. It reads the data stored in the DBN0s and other metrics stored in the DBN1, in batch in well-defined periods of time and the produced KPIs and KQIs are then stored in batch into the DBN1, where they are accessible to the Altaia Portal module. During KPIs and KQIs calculation, this module is also responsible for evaluating pattern deviations through fixed and dynamic thresholds using the Thresholds component. If new data arrives at the DBN0 for a time period already processed, it detects it and calculates and resubmits the new KPIs and KQIs to the DBN1. It also has a dashboard to configure the data stores, provided by the Framework Configuration, and the Metrics component allows defining new KPIs and KQIs. The SLA Management component manages monitors and reports on agreed SLA compliance. The planning component is responsible for predicting future data behavior or indicator trends and the QoS Alarm Management detects alarms and notifies other systems of these.
- **Altaia Portal:** The Altaia Portal module is responsible for analyzing the key indicators from the DBN1 data stores and presents them in reports (already defined - basic reports - or custom made with ad-hoc queries - enhanced reports) or in real-time dashboards. Through the Navigator component, it also enables drill-to-detail querying of DBN0 data stores when the analyst needs to consult the data that generated the key indicators. The Geo Indicators component is responsible for displaying geo-referenced performance indicators.

For further information about the business orientation and the functional description refer to Section 9.3.1 in Appendix A.

6.2 Real Time Network Performance Analysis: Data Requirements & Limitations

The current system uses two different sets of data stores: DBN0 and DBN1. The DBN0 is used to:

- Stage the raw information that is collected from the network and the network nodes in a common format;
- Retain the raw information for drill-to-detail queries and possible resubmission when new data arrives for a block already being processed;
- Store the results of the Altaia Correlation in a common format.

At this level, it is critical to read from and insert to the DBN0 a large number of records, which will imply that the throughput is more important than the delay.

The DBN1 is different as it is used to store the calculated key indicators from the data stored in the DBN0, which means that a large set of record from the DBN0 results in a smaller number of records to store in the DBN1. Since the data in the DBN1 is prepared for analysis, the queries an analyst might produce generally return a small number of records, but the delay must be very small because of the human interaction.

Currently, the data stores provider for the DBN0 and DBN1 is Oracle but, with the integration of Client Quality Management (CQM) in the system, the amount of records are growing exponentially, resulting in the need of using several scalable technologies in order to support the requirements in terms of insertion and selection of records.

A current limitation in the threshold evaluation is that only materialized data (data stored) is being evaluated against thresholds. CEP could prove to be useful too by allowing non-materialized data to be evaluated as well.

The requirements that affect our data stores are divided into the following clusters:

Table 12: List of Requirement Clusters – Network Performance Analysis Case.

Requirement Cluster (RC)	Name	Brief Description
RC-A	Technological	Requirements regarding the support of several technologies.
RC-B	General	Requirements regarding the system's general requirements.
RC-C	Delays	Requirements regarding the maximum delay across the system.
RC-D	Throughputs	Requirements regarding the minimum throughput across the system.

For the Delays and Throughputs requirement clusters, the defined metrics are not independent, i.e. the delay values are affected by the throughput of the system and vice-versa. Each requirement is mapped to the numbers identified below, in the image of the future architecture in Figure 9 (the arrows indicate the flow of information), which are key points of interaction between the system and the data stores, which will be henceforth referred as connectors.

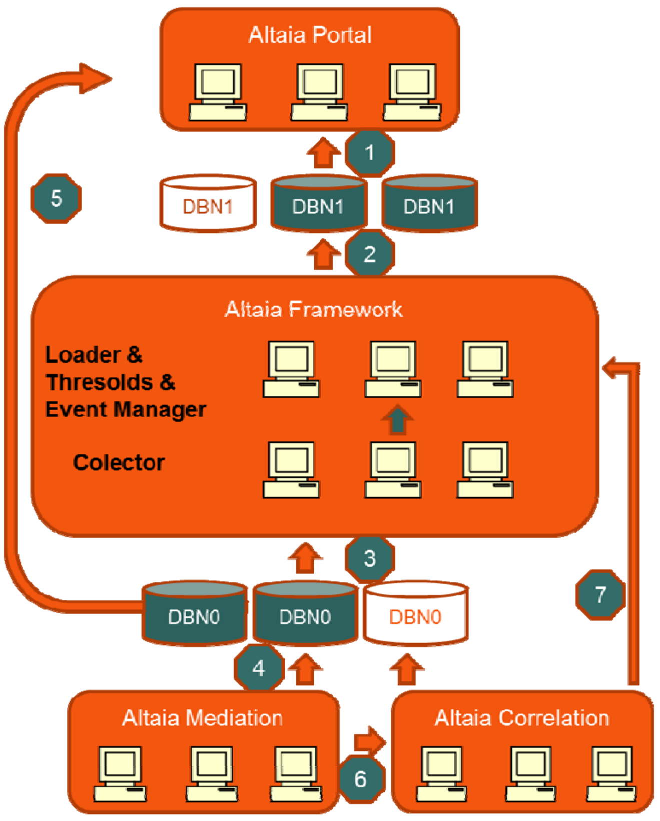


Figure 9: Mapping Requirement onto Future Altaia Architecture.

Table 13: List of Technological Requirements – Network Performance Analysis Case.

Technological Requirements	Name	Brief Description
ReqA.01	DBN0_TECH	The system must be able to support several technologies at the DBN0 layer with different trade-offs.
ReqA.02	DBN1_TECH	The system must be able to support several technologies at the DBN1 layer with different trade-offs.
ReqA.03	EVENT_CORRELATION	The system must support event correlation.

Table 14: List of General Requirements – Network Performance Analysis Case.

General Requirements	Name	Brief Description
ReqB.01	DBN0_SCALE	The system must support several instances at the DBN0 layer.
ReqB.02	DBN1_SCALE	The system must support several instances at the DBN1 layer.
ReqB.03	DBN1_INSERT_TRANSACTIONAL	The system requires atomicity at the record level.

Table 15: List of Delay Requirements – Network Performance Analysis Case.

Delay Requirements	Name	Brief Description	Acceptable Metric	Target Metric
ReqC.01	DBN1_OLAP_QUERY_DELAY	The system must be capable of completing Online Analytical Processing (OLAP) queries to DBN1 in less than 1 second.	2 seconds	1 second
ReqC.02	DBN1_INSERT_UPDATE_DELAY	The system must be capable of inserting and updating the DBN1 metrics in less than 120 seconds.	300 seconds	120 seconds
ReqC.03	DBN0_OLAP_LIKE_QUERY_DELAY	The system must be capable of completing OLAP like queries[1] to DBN0 in less than 120 seconds.	300 seconds	120 seconds
ReqC.04	DBN0_INSERT_DELAY	The system must be capable of inserting data to DBN0 in less than 120 seconds.	300 seconds	120 seconds
ReqC.05	DBN0_OLAP_LIKE_DRILLDOWN_QUERY_DELAY	The system must be capable of completing OLAP like queries[1] to the DBN0 in less than 1 second on drilldown.	2 seconds	1 second
ReqC.06	EVENTS_INPUT_DELAY	The system must be capable of providing events in less than 1 second.	2 seconds	1 second
ReqC.07	EVENTS_OUTPUT_DELAY	The system must be capable of processing events in less than 1 second.	2 seconds	1 second

[1] OLAP like queries are OLAP queries that return from thousands to millions of records.

Table 16: List of Throughput Requirements – Network Performance Analysis Case.

Throughput Requirements	Name	Brief Description	Acceptable Metric	Target Metric
ReqD.01	DBN1_OLAP_QUERY_THR	The system must be capable of reading over 1.000.000 records per hour (approx. 300 records per second) from DBN1.	150 records / second	300 records / second

ReqD.02	DBN1_INSERT_UPDATE_THR	The system must be capable of inserting over 500.000.000 records per hour (approx. 150.000 records per second) into DBN1.	60.000 records / second	150.000 records / second
ReqD.03	DBN0_OLAP_LIKE _QUERY_THR	The system must be capable of reading over 1.000.000.000 record per hour (approx. 300.000 records per second) from DBN0.	120.000 records / second	300.000 records / second
ReqD.04	DBN0_INSERT_THR	The system must be capable of inserting over 2.000.000.000 records per hour (approx. 600.000 records per second) into DBN0.	240.000 records / second	600.000 records / second
ReqD.05	DBN0_OLAP_LIKE _DRILLDOWN _QUERY_THR	The system must be capable of reading over 1.000.000 record per hour (approx. 300 records per second) from DBN0 on drilldown.	150 records / second	300 records / second
ReqD.06	EVENTS_INPUT_THR	The system must be capable of consuming over 100.000.000 events per hour (approx. 30.000 events per second).	15.000 records / second	30.000 records / second
ReqD.07	EVENTS_OUTPUT_THR	The system must be capable of producing over 100.000.000 events per hour (approx. 30.000 events per second).	15.000 records / second	30.000 records / second

6.2.1 Addressing The Appropriate Data Store

Unlike the other use cases, where different sections of a data store are being mapped to different alternatives, this use case has two data store layers that are not so easily divided. The DBN0 is composed of non-normalized tables that are used to store the same type of data, i.e. raw data, and the DBN1 is used to store only aggregated data, i.e. calculated KPIs and KQIs, using a star schema. Instead of trying to find different data store alternatives like the other use cases did, we will analyze the DAP to our data and propose alternatives that might handle those better.

The image of the future architecture presented in the parent section illustrates the flow of information in our future architecture. Between each module and a data store layer

(DBN0 or DBN1), there is a connector with a specific DAP that may be addressed better by a certain type of data stores than others. The type of data store technology currently used on both the DBN0 and DBN1 is relational.

To address the appropriate data store we will start by analyzing the DAP of each connector:

Table 17: List of DAPs – Network Performance Analysis Case.

Connectors	DAP	DB Technology Alternatives	Key Point
1	Sparse Data Retrieval Aggregations • Sums; • Averages; • Max/Min. Ordering Filtering • Time range.	Columnar • Data access enhancement; • Data compression. In-Memory • Data access. NoSQL w/aggregation support	Indexing/Partitioning BASE is sufficient Massive Parallel Processing Proactive caching
2	Batch Inserts/Updates in stable time periods Time contiguous (non sparsed)	Columnar • Optimized data insertion. Row oriented	Resilience through in-memory replication rather than disk fsync Massive Parallel Insertion Insertion time is critical
3	Periodic Batch Read Time contiguous	Relational Distributed File Systems • Hadoop.	Indexing/Partitioning Queue Messaging Proactive caching Map-Reduce
4	Batch Inserts Time contiguous	Relational Distributed File Systems • Hadoop.	Indexing/Partitioning Queue Messaging Map-Reduce
5	Block read with refining Sparse Data Retrieval	Key-Value Distributed File Systems Hadoop.	Indexing Raw Data Retrieval Proactive caching

The key points represent certain aspects that we find important in the underlying data store to respond to the DAP of each connector (only some are currently being explored and used).

- Connector 1 - usefulness and adequacy of massive parallel processing and proactive caching;
- Connector 2 - adequacy of massive parallel insertions and resilience of data through replication in memory (rather than flushing to disk);
- Connector 3 – adequacy of proactive caching as the Altaia Framework normally requests blocks of time contiguous data at constant intervals;
- Connector 5 - using Hadoop as the underlying technology could be interesting and, once again, proactive caching could improve performance by prefetching data.

At both data store levels, to address the needed DAP in each connector, the current data store solution is Oracle's relational data store, given the adequacy of its indexing and partitioning features.

However, and upon analysis of the DAP in the different connectors, we can conclude that in each case, some data store technologies are better suited than others:

- Key-Value data stores are well designed for the connector 5, which requires fast response times as defined in the requirement ReqC.05, due to the specific DAP - getting single values using a key.;
- Columnar data stores are also a possibility for the connectors 1 and 2 as, due to the usage of a star schema and the low delay accepted in the requirement ReqC.01, it may improve the performance. Other options are in-memory data stores with eventual consistency, due to the requirement ReqB.03 (only atomicity is needed) or other NoSQL data stores that have data aggregation support;
- Distributed File Systems with added layers of functionalities, like Apache Hadoop, is another option as an underlying technology to handle the DAP of the connectors 3, 4 and 5.

6.3 Development Process

This use case requires **reengineering** an existing application – Altaia – which provides real-time network performance analysis in a telco environment. We assume the following work components:

Design & Experimentation

Aims at designing in higher details the mixture of data stores that are judged as necessary in order to satisfy the listed requirements, as well as exploring the different available technologies to identify the best suited to perform the different functionalities.

Data Collection Processes

Refers to the group of tasks that result in collecting network and client raw data and load it to DBN0. It will explore the different available technologies which will be validated accordingly to the efficiency, scalability and adequacy to the different required tasks.

Correlation Processes

Refers to the tasks needed to optimize event correlation processes, main focus on validating efficient and scalable technology implementation.

Indicator Analysis Processes

Create and test analytical processes performed in Portal component, with particular focus on implementing efficient and performing drill-to-detail queries (Portal -> DBN0).

Query Implementation

The next step is to create the workflow, queries and operations which are necessary to collect data from DBN0, process - calculate KPI and KQI, threshold violations, data correlation – and load to DBN1. All queries will be tested and their performance will be measured for validation purposes.

Some of the above work-components (especially, “Design” and “Query implementation”) will be iterated based on requirements to be met, measurements and findings during testing. We assume that significant changes and reengineering will be required for:

- DBN0;
- DBN1;
- Correlation.

while the following components will remain relatively unchanged and will only be adjusted to become compatible with the components that will change considerably:

- Mediation;
- Framework;
- Portal.

6.4 Guidelines For The Validation Phase

Validation will be done in a systematic way so that to check that the new implementation of the use case:

- Functions well, is correct and complete, comparing always with the previous implementation;
- Performs within acceptable levels, or even reaching the goals set;
- Has utilized the best mixture of technologies from what is available through CPaaS and;
- Cost effective in what concerns Total Cost of Ownership and impact on application reengineering effort.

It must seek ways to ensure that the use case is implemented in a way to remain :

- **Correct:** Previous architecture (already and extensively tested) delivers specific output for a given input of usage. Same data will be given as input to new implementation;
- **Efficient:** Measuring performance will show at which level the desirable (either at the level of acceptance or at the level of ultimate target) metrics are reached. Metrics are given in Section 6.2;
- **Scalable:** Moreover, stress-testing should be applied. Input sizes should be modified (either number of scenarios, or number of billing cycles or number of

customers or usage levels or all possible combinations) and system behavior will be assessed into different input levels. Finally, it should be clear how system must scale in order for the use case to remain acceptable as load increases;

- **Optimal Use of Technologies:** Combining validation results with X-ray platform findings will drive decisions towards improving and/or correcting the CPaaS platform or selecting another design choice for data storing and analytics.

7 BIBLIOGRAPHIC SEARCH USE CASE

7.1 Use Case Outline

The Bibliographic Search use case aims at collecting a feed of tweets about science, filtered and enriched with the help of the used ontologies. These tweets will be automatically linked with European projects (through Cordis), bibliographies (through DBLP, PubMed) and patents (through Wipo) related to the search keywords or the extracted keywords from the feeds. It will also find relationships and social metrics of scientists through the bibliographies or through the social information obtained from twitter feeds. The goal is to find the best reviewers for a given European project and determine which European projects are more related to a given article.

In order to provide this functionality, the project will use different data store technologies and queries in CPaaS. On the transactional side, the feed from Twitter will be used to synchronize data from different repositories and provide accurate and timely answers to the users. From the language point of view, a single point of entry will be used, easing the application developers' task to create new functionalities from different data stores.

The system has the following modules:

- **Crawlers:** Perform periodically crawling processes for CORDIS and DBLP;
- **Loader**: Performs a periodically loading process for new E.U. projects and papers. This component might use storm to provide a distributed and incremental computation system. The loader will store the basic information related with a project and will index it in the most appropriate way to ensure a high performance in the analytical part;
- **Analytical layer**: It performs the analytical queries using the common query engine. These queries are recommendation engines to resolve reviewers and related European projects. We will use the XWork / Struts2 because it is a rich MVC framework with an IoC container;
- **REST/Web layer**: It exposes the analytical layer to the web through a REST API. Multiple servers with the same REST api will be running and a single load balancer will distribute the requests. The results will be integrated into Sciencea web application.

Figure 10 summarizes the physical and logical view of the described modules.

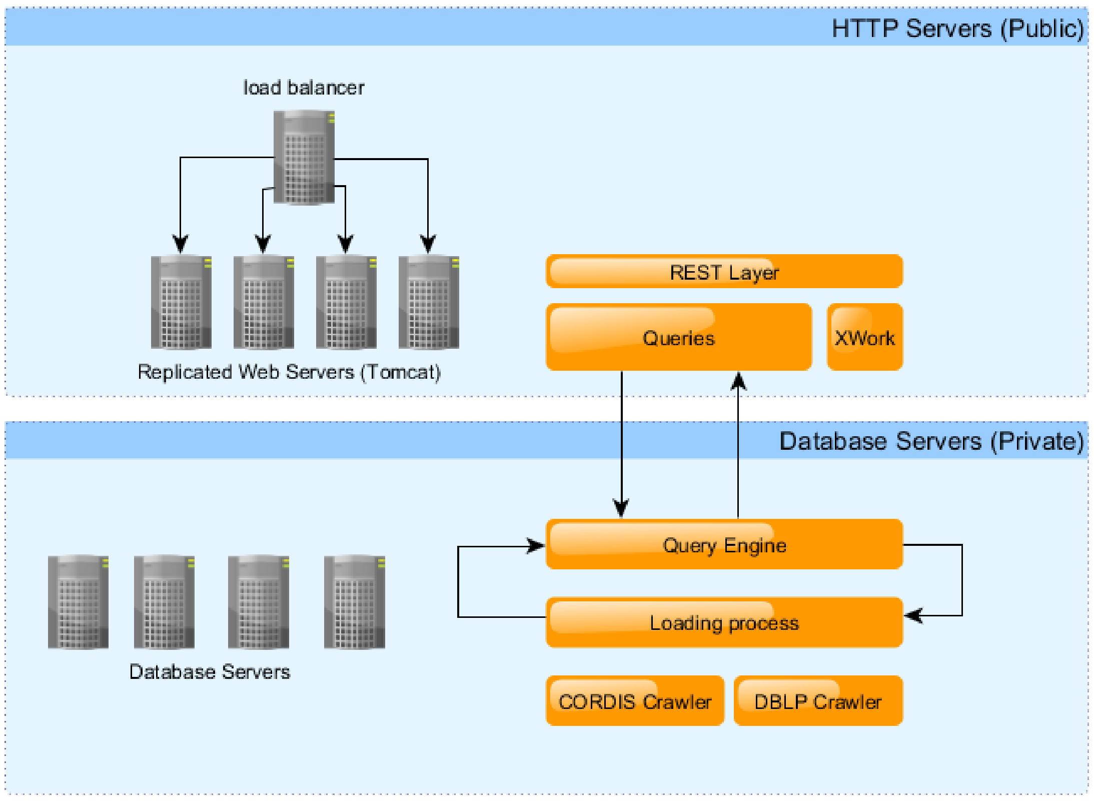


Figure 10: Bibliographic Search Use Case: Physical and Logical View.

For further information about the business orientation and the functional description refer to Section 9.4.1 in Appendix A.

7.2 Bibliographic Search: Data Requirements & Limitations

The scientific production is not as fast as the social networks because the author needs to produce high quality content. Therefore, according to our previous experience there are less than one thousands of documents per month to load. So, trending topics or the researchers' public reputation do not change in real time. However, the contents of projects and papers are longer than microblogging channels such as twitter and thus, the computation of analytical queries become more complex.

Our bibliographic clients only demand fast queries to get results in real-time. We have rewritten this requirement with the following description:

Table 18: List of Requirements – Bibliographic Search Use Case.

Requirements	**Name**	**Brief Description**	**Acceptable metric**	**Target metric**
Req.01	MAXIMUM_TIME	The system must be able to solve all analytical queries in less than 1 minute	1 min	2sec.

7.2.1 Addressing The Appropriate Data Store

We have selected a set of data stores to ensure a high throughput in the selected media planning operations. These operations are:

- **OP#1: Find reviewers for a project;**
- **OP#2: Related European projects with a given paper;**
- **OP#3: Add a new potential reviewer without publications;**
- **OP#4: Adding new EU projects or papers.**

Sparsity-Technologies and INRIA have experience building bibliographic applications such as Sciencea, a web application to search and analyze European projects that allows to know related DBLP papers with a given set of keywords. Let's see the concepts used in CORDIS and in a bibliographic data store before explaining the detail of each suggested operation for this use case and the design decisions related with the data stores.

CORDIS is a very rich data store of types. It contains calls, programs, projects, categories, institutions, the relationships between them.

The CORDIS schema needs to be necessary extended to materialize those relationships or types that maximizes the performance of the queries.

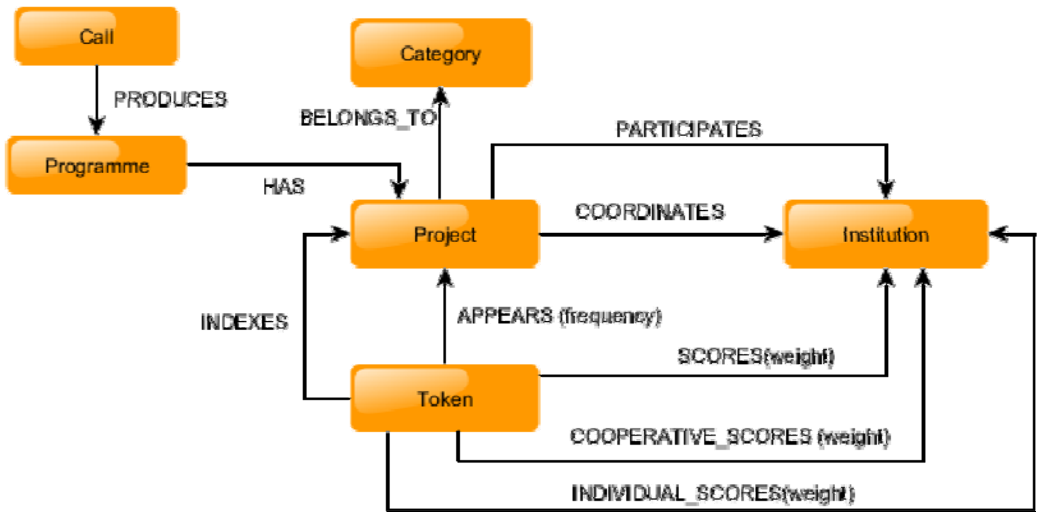


Figure 11: Extended Schema to support CORDIS.

For example, the previous diagram on Figure 11 shows the SCORES and APPEARS relationships that are created from inferred information from projects' abstract. Particularly, the SCORES relationship stores which are the expert institutions for a given token and the APPEARS relationship serves to store the tokens frequency in a project.

Some of the basic fields stored in the CORDIS schema are summarized as follows:

- Projects have title, acronym, title, description, starting date, ending date, cost, funding, url, year and last updating date;
- Institutions have code, name, contact information, country, region, city, url and address;
- Programs have acronyms;
- Calls have identifier, title, program, theme, type, publication date, deadline, budget and modification date.

A pure bibliographic data store, such as DBLP, is simpler than CORDIS because it just contains documents, authors and references (sometimes).

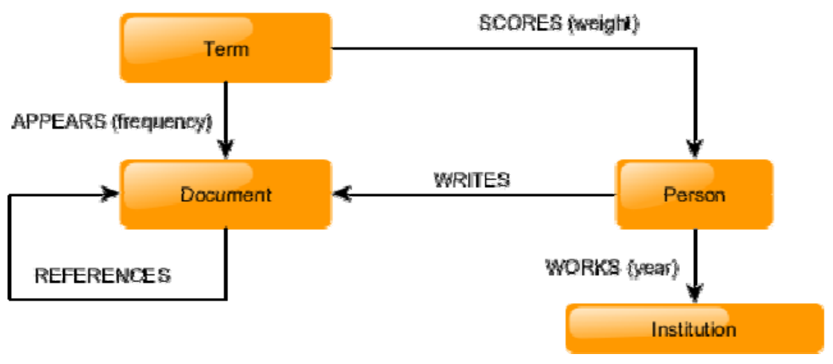


Figure 12: Extended Schema to support DBLP.

Some of the basic fields stored in a bibliographic schema are summarized as follows:

- Documents might have id, title, description, state, date, URL, pages, first page, last page, ISBN, number, volume, book title, publisher, updating date, school, editor, address, cite, chapter and type;
- Person might have a code, name, contact information, country and language;
- Institutions have id, name and country.

Now, let's see the detail of the selected operations for this use case.

<u>**OP#1: Find reviewers for a project**</u>

Description: The 10 most scored people in a project expertise area with have not been working in any of the participant institutions. The result must be sorted by the score. For each person, the system must show:

- His / Her current affiliation;
- His / Her last paper;
- His / Her basic fields: code, name, contact information and number of publications.

As it has been explained before, to solve the query efficiently, we need to extend the schema indexing people by keyword through the SCORES relationship. Moreover, it is necessary also to store the relevance of a given keyword for the indexed element.

We suggest to store the involved information with the following data stores:

Table 19: List of data stores suggested to apply the extended schema for OP#1 – Bibliographic Search Case.

Type	Data store	Resolves	Structure
SCORES (weight)	Key Value	The most experts of a given keyword	Key: keyword Value: SortedSet<{Person.id, weight}>
WORKS / <u>PARTICIPATES</u>	Graph	The common institutions between a person and a project. Requires to merge the schemas.	Node types: Institution, Person and Project. Edge types: WORKS, PARTICIPATES.
Person	Relational data	The required information of each	A table for type indexing each

Document WRITES LAST_AFFILIATION	store	potential reviewer. The LAST_AFFILIATION is a new inferred type to store the current author institution.	element by its primary key.

OP#2: European projects related with a given paper

Description: The active European projects in a given paper publication date, which share some participants with the paper affiliations and share some of the three more relevant words of the paper. For each European project, the system must return:

- The project basic information.
- The most relevant keywords of the project.
- The project coordinator and the other participants.

We suggest to store the involved information with the following data stores:

Table 20: List of data stores suggested to apply the extended schema for OP#2 – Bibliographic Search Case.

Type	Data store	Resolves	Structure
RELEVANT_WORDS	Key Value	It is a new structure to store the 3 most relevant words in a given paper.	**Key**: document.id **Value:** SortedSet<{word, weight}>
APPEARS WRITES WORKS PARTICIPATES	Graph	The projects' filters: Those which share institutions and words.	Nodes: Document, Institution, Person, Project Edge types: WRITES, WORKS, PARTICIPATES
Project (without description) COORDINATES PARTICIPATES Institution	Relational data store	The required information for each potential project	A table per type. Projects and Institutions indexed by their primary key. COORDINATES and PARTICIPATES have foreign keys to Project and Institution.
Project (id, description)	Documental	Required information for each potential project	A document indexed by its id.

OP#3: Add a new potential reviewer without publications

Description: Add a new person with all the institutions where he/she previously has worked and a set of expertise areas (keywords). These are the affected types by this event: Person, Institution, SCORES and WORKS, which are stored as explained in previous operations.

OP#4: Adding new EU projects or papers.

Description: DBLP periodically publishes the updated version of the whole data store in a single XML file, but there isn't other way to get the new data in real-time. Therefore, the system will need to add those new documents executing a DBLP loader periodically.

CORDIS, neither has a streaming API, such as twitter, where the new published projects are pushed to the listening applications. However, they have an HTTP service where people can ask for those new projects published in a time interval. Sparsity-Technologies has a crawler, specifically built for Sciencea, to ask for these new projects using public CORDIS' HTTP services. This crawler it is periodically executed by a Cron task.

Once the data becomes updated, we will proceed a complex ETL process, which should be performed with Storm, a distributed real-time computation system.

7.3 Development Process

This use case implies a partial **reengineering** of an existing application, which provides a set of analytical queries for bibliographic data stores and CORDIS, called Sciencea. These queries include reviewers and related projects analysis. We assume the following work packages:

Loading package

Our first forthcoming work is the creation of a design to load CORDIS projects and DBLP papers periodically using Storm, and the insertion of new projects and papers in different data storages. Some loading processes, such as those which store rankings (e.g. SCORES) may require a minimum set of data to get quality values because they need a representative dataset. In this case, these processes won't be executed until a minimum set of new data is reached.

Queries package

The next step is to design and create the analytical operations (i.e. reviewers and related projects), which needs to offer fast responses (less than one minute) because they will be requested form a visual interface. These queries will be tested from a prompt.

REST/HTTP package

Finally, the last step is to add an HTTP layer to execute the included analytical queries and integrate them into the current Sciencea application.

7.4 Guidelines For The Validation Phase

The main goal of this use case is to load periodically EU projects and DBLP papers and provide two fast analytical queries to improve enrich an existent application called Sciencea. The quality indicators, described in detail in 8.2, are correctness, efficiency, scalability and the evaluation of the optimal use of technologies.

The system will use many mathematical models to resolve the queries, like the TF-IDF model to measure the relevance of a term in a paper or project. All these mathematical models, which are used to infer relationships such as scores, has been already based on existent technical papers.

Sparsity-Technologies will design a validation phase using a set of **unitary tests** to assure the queries' correctness and also the correctness of each of the loading steps. Moreover, we will build an **integration test** which will simulate the Storm behavior with the testing dataset and once all documents had been consumed, the contents of the entire data store will be verified. Moreover, INRIA and Sparsity-Technologies will use the whole contents of CORDIS and DBLP to test the system to validate the **scalability** and **efficiency** of the new system.

8 Overview Of Requirements & Concluding Remarks

8.1 Executive Summary of Data Requirements Of The Use Cases

The tables below summarise the list of requirements, data stores engines, and DAPs for each use case.

Table 21: Cloud Telecom/M2M - CDR analytics.

Data Requirement	Data Store Engine	DAP
Margin analysis and efficient analytics	Columnar	Aggregation of data
Store raw and after processing (rated) data.	Key-Value Hadoop like distributed file system	Read single values using a key.
Analytics on CDRs and performing what-if analysis.	In-Memory	Data aggregation and compression. Data analysis.

Table 22: Cloud Telecom/M2M – M2M.

Data Requirement	Data Store Engine	DAP
Scalable and fault – tolerant data reception	CEP	Reception and processing of messages at high throughput
Low latency message handling	In-memory	Alarm handler definitions need to be fetched rapidly for each vehicle message in order to in real-time identify possible alarms.
Real-time views	In-memory	The most recent data must be cached for quick views on present vehicle status.
Store Remote data sent by vehicle devices	Hadoop like distributed file system	Massive insertions to an append-only immutable

		data set
Get a snapshot view of past vehicle status	Columnar	Data analysis Aggregate information needs to be fetched and presented

Table 23: Media Planning.

Data Requirement	Data Store Engine	DAP
The most influencers given a keyword.	Key-Value	Read collections using a key.
The maximum propagation depth, the number of influenced entities, and document related information (copies, references, propagations and depicts)	Graph	Apply a DFS algorithm and complex filters and joins over relationships
The documents and the basic information of an author	Relational	Read a set of basic fields of a set of registers.
The last document of an influencer.	Document-based	Documents are stored in a Document-based data store because they may have a long abstract.
The most influencers within a community for a given set of keywords	Graph	Because the influence relationships are stored in a Graph to find communities
The 10 biggest communities by keyword	Key-Value	Read collections using a key.
The document of an influencer.	Relational	Read a set of contiguous stored fields.
The last microblogging	Document-based	Documents may have a

document of an influencer.		long abstract.

Table 24: Summarised list of requirements, data store engines and DAPs - Real-Time Network Performance Analysis.

Data Requirement	Data Store Engine	DAP
OLAP queries on DBN1	Columnar NoSQL w/aggregation support	Aggregations, ordering and filtering
OLAP queries on DBN1	In-memory	Sparse data retrieval
Store calculated KPIs and KQIs on DBN1	Columnar Row-Oriented	Batch inserts/updates of time contiguous data in stable time periods
Read raw data for KPI and KQI calculation from DBN0	Relational Hadoop like Distributed File Systems	Batch read of time contiguous raw data.
Store raw, time contiguous data on the DBN0	Relational Hadoop like Distributed File Systems	Batch inserts of time contiguous data
Drill-to-detail for the data that was used to calculate the KPIs and KQIs	Key-Value Hadoop like Distributed File Systems	Block read of sparse data with refining

Table 25: Bibliographic Search.

Data Requirement	Data Store Engine	DAP
The most experts of a given keyword	Key-Value	Read collections by key.
The common institutions between a person and a project.	Graph	To apply efficient joins and filters.
The required information of each potential reviewer.	Relational	Read a set of contiguous stored fields with a fixed length.
The three most relevant	Key-Value	Read collections by key.

words in a paper.		
The papers that share institutions and words.	Graph	To apply efficient joins and filters.
The potential information for each potential project.	Relational	Read a set of contiguous stored fields with a fixed length.
The required information for each potential project.	Document-based	Projects may have a long description.

8.2 Overall Validation Guidelines

Validation will be done in a systematic way so that to check that each use case exhibits the following qualities when compared to the previous implementation:

- Is correct and complete. The previous architecture (already and extensively tested) delivers specific output for a given input of usage. Same data will be given as input to new implementation as part of unit and integration tests;
- Performs within acceptable levels and reaches the target goals. Measuring performance will show at which level the desirable (either at the level of acceptance or at the level of ultimate target) metrics are reached as defined in each use case. In addition, input sizes should be modified and the system behavior must be assessed. Finally, it should be clear how each use case must scale its system in order for it to remain acceptable as load increases;
- Uses the best mixture of technologies from what is available in CPaaS. Combining validation results with X-ray platform findings will drive decisions towards improving and/or correcting the CPaaS platform or selecting another design choice for data storing and analytics.
- Reduces Total Cost of Ownership and application re-engineering effort.

9 Appendix A

9.1 Cloud Telecom/M2M Use Case – CDR analytics

9.1.1 Business Orientation

Extensive analytics on usage data of telecom operators' networks can be considered as a very specific case of an M2M application. Machines are the communications network devices, while M2M data are representing the voice and data usage details, namely the CDRs.

For each communication event one or more records are produced by the network equipment. These are typically in binary format. A process called mediation processes these records and creates a number of CDRs, typically in ASCII format. Voice Call Records contain limited information, typically call time, originating number, terminating number, country code, call duration and other, operator-specific fields. Initial parsing of CDR (rating process) produces "rated CDRs", which are CDRs with associated costs, prices or other elements.

Finally, more CDR analytics processes follow in order to perform several important for telecoms tasks, including a) billing operations beyond rating, i.e. invoicing and reporting, b) marketing operations i.e. customer base segmentation, pricing simulation (what-if analysis), social network analysis, margin analysis and c) fraud detection and prevention

In the project use case, we are focusing onto pricing simulation case and margin analysis, as realized by Neurocom's product TariffSuite and its analytics applications on top of it. TariffSuite is a pricing simulation and analysis tool that helps single or multiple play telecom providers to:

- Develop new bundles of products and tariffs;
- Define customer retention strategies;
- Manage costs;
- Analyse margins.

TariffSuite offers the capability to rapidly execute several pricing scenarios on several traffic patterns. Scenarios are combinations of rateplans, addons, discounts and promos. Every event is rated according to one or more scenarios (tariff plans). Results are compared against each other in order to benchmark instantly the impact on revenue and margin of one or another tariff plan applied to a specific set of traffic. It is based on the processing and analysis of real traffic data rather than aggregates. Its goal is to shorten the cycle:

"define experiment (alternative tariffs) →→ execute experiment (re-rate traffic) →
→ analyse results".

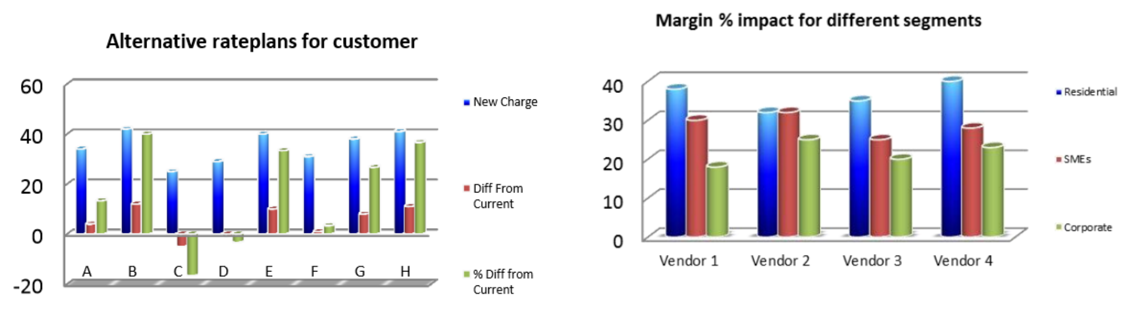
Figure 13: Price Simulation Flow

This gives telecom operators the precise information they need to react quickly to any developments in the market. Questions and problems that are tackled through tariff simulation and margin analysis are mentioned below:

- **Keep telecoms customers happy.** Tariff Suite gives pricing and product managers the ability to benchmark their new offers against the competition by calculating what customers would pay if they switched to alternative plans. It enables them to fine-tune the offer by using iterations to produce the "best" offer for attracting the customer - lowering their invoice - without giving away all business margins.
- **Analyze Competition - Predict Impact on Revenues.** New plans and options proposed by a competitor can be immediately analysed. TariffSuite calculates what customers would pay in these plans and the differences in billing charges based on real traffic. Product development teams are having a tool to do preliminary analysis before introducing new plans. They can:
  - compare new plans to offers made by the competition;
  - identify the best strategy to defend;
  - check if new offers are really attractive;
  - find the best alternative offer;
  - predict impact on revenues.
- **Moving customers** to new plans - Tariff Suite calculates what customers would pay for each rate plan/add-on combination available in the company for a specific sales channel. This makes it easy to analyse alternative bills and decide which new offer should be proposed to the customer. Rate plan management teams can make better decisions :
  - detect the "best deal" telecom companies can offer to their customers;
  - control consolidation of customers into the most important rate plans;
  - move customers to the best rate plan according to their usage profile.
- **Reconciliation of vendor bills -** Tariff Suite enables procurement teams to check the bills produced by their service providers. The calculation is based on the traffic details generated by internal logging systems (e.g. switches) or the traffic files received from service providers. With TariffSuite it is easy to detect errors in the bills of service providers, such as differences in minutes, the wrong pricing applied, etc.
- **Margin Analysis for optimised sales.** It allows management to allocate resources to commercial offers meeting company strategy by:
  - accurately predicting future margins when introducing new plans and add-on services;
  - identifying customers, plans and profiles that produce the highest and lowest margins.

- **Margin Analysis for optimised cost control.** Dealing with several countries/providers usually means dealing with different and quite complex pricing models. Once the different models are uploaded, comparing the costs of different service providers with actual traffic becomes simple. Cost and procurement managers can determine and optimize their purchasing strategies. Margin analysis can give a very clear understanding to cost control and procurement teams of what would the effect be in margin from switching traffic to another provider

Customer Retention	Margin Management



Pricing Management.

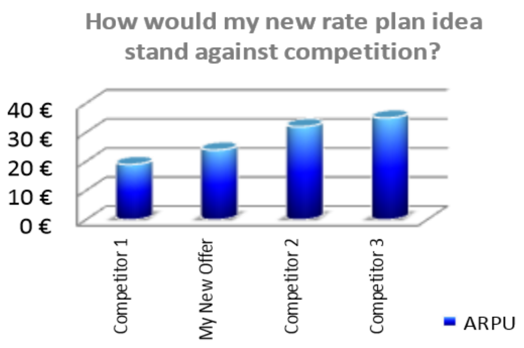


Figure 14: Examples of Tariff Simulation Applications Results.

Tariff simulation and margin analysis is designed to support a three-step pricing assessment methodology:

Step One : Define Scenarios
• Select rate plans, add-ons, discounts and promotions • Select customer and contract information • Provide the traffic data you want to include in the simulation
Step Two: Re-rate & Calculate alternative bills
• On average, a billing run of 35,000,000 CDRs is re-rated 10 times (10 alternative bills calculated) in about 30 minutes.
p Three: Analyse Results
• Per rate plan and service • Per invoice and customer • Per month • Numbers of calls, volume, charge, free units • Data available in a form suitable for reporting (data store tables)

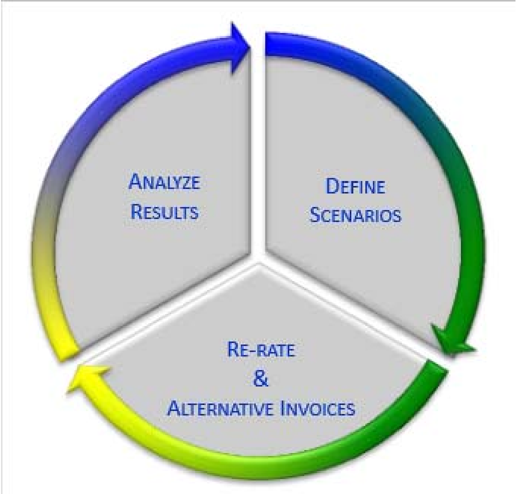


Figure 15: Three-Step Pricing Assessment.

Our use case will be based on the **Margin Analysis application,** the most important functionality of tariff simulation, where requirements are extremely demanding and scaling issues are faced. We are focusing the use of the margin analysis application to the larger customer accounts, mainly the corporate ones. The application is realized by a web application customized to the needs of corporate accounts management. It groups data by large accounts and allows sales persons to try different scenarios so that to get estimations on financial offers for existing and potential customers and produces margin analysis reports. A scenario is a combination of a baseplan with one or more addons and promotions. Sales efforts are aiming to identify how revenue is shaped when different scenarios are applied to customers' usage and whether a new offer (and which offer) is attractive. Level of attractiveness depends on its value distance from current pricing or from hypothetical (i.e. competitor's) pricing.

The main actors involved are:

- **Sales managers,** the account managers who take care for the management of relation with the customer and have to analyse revenue per customer, customer satisfaction and provide alternative offerings when necessary;
- **Product managers,** who need to analyse competitors' offers and define new packages or new discount options/addons etc;
- **Rating system provider** who is providing the core calculation engine to rate call usage data.

Tariff simulation and margin analysis can be differentiating tool for operators that brings benefits towards market leadership, enhanced productivity and profitability, quality and flexibility.

While all jobs run in batch mode right now, it is not scalable and demand for real-time execution and response is emerging. For instance, tariff simulation takes on average 2 hours per billing cycle; margin analysis in total is now taking 36 hours on average (data loading needs 12 hours while tariff simulation analysis on 3 billing cycles needs 24 hours), for 5.5 billion records (invoices). The total data store size reaches 320 Gbytes for tariff simulation data and 500 Gbytes for margin analysis. This is the situation for 300 scenarios which through algorithmic enhancements can support the decision for up to 1200 scenarios. These figures are expected to grow when the service will be provided over the cloud as a service, for many more customers and multiple needs. In that case more invoices need to be calculated for more scenarios (e.g. 5000) for more billing cycles and larger customer segments. Moreover, instant answers per customer will be necessary which demands the ability to isolate single customer data at once, rate and analyse them.

9.1.2 Functional Description

The Basic questions answered by the margin analysis use case for corporate accounts management, fall into the following cases:

Per customer

- Find customers with the biggest delta (distance) from their best plan;
- Find the best plan for each customer;
- Find the best offers that bring the same revenue.

Accumulative

- Find average distance of the invoices from the corresponding best plan;
- Find average distance of the invoices from the corresponding best plan per plan;
- Perform segmentation of customers based on usage, what they pay, etc.

Use Case Example: Below, there is a typical use case for a mobile operator sales person trying to create an offer for a customer. The basic moves taking place are:

- Select a customer;

- Select a period of time for which usage profile is considered representative e.g. last 3 months;
- Among a large pool of baseplan/addon/promotion compatible combinations, filter in/out the ones that seem ‘appropriate’ according to mobile operator’s targets and end user’s needs;
- Find the baseplan/addon/promotion combination that best meets some target.

Our price simulation and margin analysis application has the following high-level functional requirements:

REQ#1: The system must be able to load call usage data related to a specific set of contracts
Description: Usage data can either correspond to whole customer base, or a specific customer base segment (i.e. corporate customers) or a single customer (with a single or multiple contract numbers)

REQ#2: The system must be able to load usage data for a given time period (in billing cycles)
Description: Usage data corresponds to calls and network usage made during a specific period. The period should always fit to an integral number of billing cycles.

REQ#3: The system must be able to rate all usage data according to numerous scenarios that contain compatible combinations of baseplan/addons/discounts/promos
Description: The need is that the user can describe multiple scenarios; each scenario is a combo of base plan, addon(s), discount plan(s) and promo(s). The scenario also includes some conditions that determine if the scenario is applicable or not (e.g. rate with this scenario only if the user doesn’t have an UNLIMITED contract, or keep addons that give at least 500 MBs free data or exclude all discounts if access charge is less than 25 euros). Considering such conditions, scenarios that are not applicable (they do not have compatible components to the described conditions) are not applied through the rating process that will follow. Rating engine will be invoked and should process all usage data for each scenario. Finally, invoice lines will be produced that correspond to all possible scenarios that need to be analysed.

REQ#4: The system must be able to find the scenario that best meets a target in real time
Description: The system is given some target defined on the billed amount and then it must find the baseplan/addon/promotion combination that best meets this target. Examples of such targets are listed below: • the billed amount is the lowest possible • the billed amount is less by 10% (or other) with respect to what customer currently pays • the billed amount similar to what customer currently pays but access

charge is at least 5 euros higher
The target may not be known beforehand. The business user may want to try different constraints and targets on the fly. This is one of the most challenging features because it refrains system users from deleting data that do not participate in optimal results if the targets and constraints were known beforehand.

REQ#5: The system must be able to perform all functions also at customer level
Description: The system should be able to perform all mentioned actions (usage data collection, scenario building, optimal scenario finding) for a set of numbers, belonging to the same customer. This is the case of a corporate customer paying the bills of several individual subscribers (i.e. the case of a firm with its employees). The optimal billed amount is found in the calculated sums of billed amounts which are produced after applying a combination of scenarios applied all individuals. Usually, there is no constraint that all numbers belonging to the same customer should get the same rateplan/addons/discounts/promos. The only parameter that could potentially make sense to keep the same across all of them are company-level discounts and promos which need to be applied on the whole company and may considerably alter the end result. Number of contract numbers (MSISDNs) can vary from 3 up to a thousand or even more, with an average of 10.

REQ#6: The system must be able to apply dynamically modifications on a given combination
Description: The system should give the ability to select some scenario parameters as constants (for a specific large account), for instance keeping a fixed baseplan, or an addon, or applying a special discount) and then find the best in the remaining scenarios.

9.1.3 Technical Framework

Price simulation is dependent on the rating engine performance upon multiple rate plans. Rating is the application that performs rating of CDRs based on a single contract scenario which includes the base plan, the addon(s), the discount plan(s) and the promo(s). Price simulation uses rating engine further on to apply what-if scenarios that involve different scenarios. It consists of two phases: rating and post-processing. During the rating phase it takes as input raw files of CDRs and rates them using different combinatory plans. Plans are stored in a SQL data store and loaded at the beginning of execution into an in-memory data structure. The output of the rating phase is raw files of CDRs rated based on the different plans. It finally exports N+1 rated CDRs for each input CDR, where N is the number of different plans and 1 is the original CDR. After the rating phase, it stores in memory aggregated results. Using this data, it executes a post-processing phase and computes final results to report. These include more aggregations, selections based on thresholds, possible discounts etc.

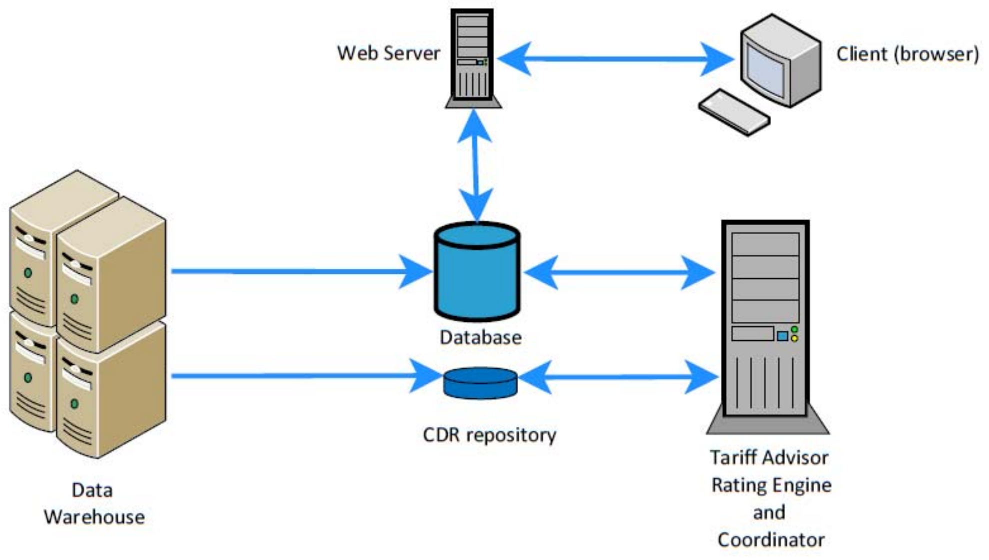


Figure 16: Current Physical Architecture of Price Simulation.

The current price simulation implementation faces two main challenges:

- **Scaling:** It is a big-data application in that it needs to process large amounts of records and plans and, depending on the use-case, provide results at real-time or near real-time;
- **Extensibility:** It needs to adapt to different use-cases as business requirements change. Therefore the functionality provided by the post-processing phase and to some extend the rating phase itself need to change over time and for different applications. Currently, both phases are written in Java, making even small extensions both error-prone and time-consuming. Instead, implementing each phase in a high-level framework, such as a query language or MapReduce can significantly reduce the time and effort required for adapting the application to new use cases;

Rating is performed on an AIX server with 16 cores and 64 GB memory. Neurocom has already commenced a project, exercising price simulation over MapReduce. Margin analysis is a Java on Oracle Data store running on a server with 8 cores and 16 GB memory.

9.2 Cloud Telecom/M2M Use Case – Machine-To-Machine

9.2.1 Business Orientation

M2M is a broad label that is being used to describe any technology that enables networked devices to exchange information and perform actions without the manual assistance of humans. It forms the basis for a concept known as the IoT. Key components of an M2M system include sensors, RFID, a Wi-Fi or cellular communications link and autonomic computing software programmed to help a networked device interpret data and make decisions. Recent advancements in technology and standards (such as 4G LTE, IPv6, big data management approaches like Apache Hadoop HDFS, scalable cloud architectures) and reduced costs in telecommunications and cloud deployment, have expanded the role M2M from the very well-known telemetry to everyday use in products like home heating units, electric meters and Internet-connected appliances. Products built with M2M communication capabilities are often marketed to end users as being “smart.” Advances in technologies have enabled the emergence of Big Data. Without the highly acclaimed Hadoop-based platforms, processing semi-structured and unstructured data would still be virtually impossible. Without reduced processing and storage costs, Big Data would remain an expensive and highly experimental endeavour for a few leading edge enterprises. Without improved networks and cheaper data traffic rates, data from individual data sources would remain isolated and unavailable for data mining. Much the same developments are beginning to drive the implementation of machine-to-machine solutions in an ever expanding range of situations. In short, the technological landscape has developed to the point where it is clear that Big Data presents tremendous opportunities for organisations and enterprises to develop and improve customised services and experiences delivered to customers by capturing, processing and analysing data from an increasing range of sources. Additionally, it is clear that data gleaned from M2M devices must also be incorporated into Big Data analyses.

Business orientation of M2M deployments spans different application domains, forming a highly fragmented environment as far as concerns business development concerns. One can easily identify such domains, the fields where M2M applications and IoT has already showcases in place. These are industry, environment, transport, energy, smart home / city, health, retail, finance, defense, sports, entertainment, etc.

All areas are demanding

- **Higher productivity and operational efficiency:** predictive maintenance policies, lower operation costs, faster response, Auto re-plenish, simpler integration, new products;
- **Valuable Creation :** new revenue stream, new business models, revenue uplift, better experience, new bundled products, more interaction.

M2M solutions can offer the means to change rapidly operational procedures and business models, delivering major benefits and abilities. A route of continuous growth is already in progress, led by variant actors, including communication service providers, device/sensor manufacturers, application providers, connection managers, application enablement providers, addressing the needs towards wide M2M deployment.

Identified as a significant new revenue opportunity for communications service providers, M2M has already become an enterprise oriented staple of many communication service provider product and services portfolios, and also as a key capability of a range of consumer propositions. Many leading operators have developed extensive M2M capabilities which can extend to a one-stop shop for M2M. Such operators can provide customers with reliable and experienced contacts for devices, modules, applications development, and systems integrations. Strong support by a PaaS platform being able to store, distribute and process securely information from many different devices, of different types with multidisciplinary requirements in terms of speed of process, size, structure, significance.

In essence, the telecoms industry already has an established M2M role which can provide a suitable entry point to introduce benefits of Big Data to their customers. In a situation where a communication service provider is already managing the connectivity component of an M2M solution, and potentially the management platform components, then that communication service provider will have already established business relationship with the customer in question. In many ways, then, offering Big Data solutions can be seen as a natural on-sell from providing M2M solutions.

A use case of a communication service provider who offers PaaS within M2M propositions can include many different applications. However, we will focus into one real application case, that of a VTP, bringing together multiple types of operations and data structures, to a sufficient level so that we can judge the abilities of the platform to accommodate the needs of more M2M deployment.

In the area of transport and vehicle telematics, many enterprises have built telematics solutions aiming at global improvement of operation efficiency and costs reduction of organizations who involve transport operations in their processes. Basic and abstract architecture components include:

- Vehicles equipped with devices (GPRS and GPS) or sensors connected to the Internet;
- Data gateways, collecting raw data before they are routed to the application for further processing;
- Data stores, where all data are consolidated;
- Applications which handle all messages and data collected from the vehicles and the associated gateways. They produce reports, real-time monitors, alerts, notification related to transport operations.

Vehicle telematics application can be used to indicate driver behaviour, vehicle status, oil consumptions, maintenance needs, assess risks, measure timeliness of transport tasks, quantify insurance risks, provide notifications to drivers and fleet operators (i.e. traffic notices, notifications for transport network disrupts). Our Use Case provider involves such application functionality offered today in a both license mode and SaaS mode.

The main actors involved in the use case are:

- **Communications Service provider (CSP)** who is providing the PaaS within an overall M2M service proposition that avails a solution for Big Data analyses to its customers. The solution can be optionally bundled to the communication network of the CSP;
- **VTP** who is providing the core applications to assist drivers and fleet managers, namely the end customers. VTP is a customer of the CSP;
- **Drivers** who are receiving continuous assistance. They are customers (or end users) of VTP:
- **Fleet Managing Organisations** are also customers of the VTP. They are getting reporting, online monitoring and alerting services through variable means of visualization.

Driver assistance and fleet management are of valuable importance for the transport industry and for any industry that includes transport operations into its critical path of operations, affecting quality and cost (i.e. car rental, freight transport companies, courier services, catering companies, taxi networks, retail chains).
Within our use case, number of devices to be monitored and managed by telematics applications is increasing rapidly. Their functionality and complexity are also increasing. Data analysis complexity is increasing in parallel. The performance of telematics applications is far too low to be able to support such growth in devices, given current SaaS architecture.
In addition to the points mentioned above it should be also noted that:

- Telematics application tends to integrate more than just one device in the vehicle e.g. RFID or fuel meter and correlate them;
- Vehicle devices which used to analyse data to provide measures to the main system are now sending raw data to be analysed by the main system;
- The large volume which is sent to the main system is also re-analysed to provide relevant information;
- This analysis which is done at single vehicle level could be extended to multiple vehicle and multiple devices by vehicle;
- For some application area the system will require real time, highly available versions.

In this context, it is evident that the VTP should seek ways and technology platforms to multiply the capacity and ability of its services. As the amount of devices increases from years to years, he needs to improve this infrastructure in order to support scalability, high availability, stability and performances. While a NoSQL solution has been considered for the future, the provider wants to elaborate a use case in order to validate this assumption among other ideas.

9.2.2 Functional Description

The Telematics application description hereunder defines the place in the architecture of the PaaS and sketches the possible areas to be covered by the use case. It is expected that the PaaS will store and distribute securely information produced by a variety of devices. As we are in the telematics area, important information is the geographical coordinates and the time tag attached to an entity e.g. a vehicle. Entities can be grouped in different ways and hierarchies, these associations need to be managed according to privileges granted to the users of the platform.

Management and contextual time bound objects will also be stored in the PaaS like:

- Maps;
- Geofencing zones;
- Schedules and tasks;
- Measures like safety-driving scores and eco-driving scores;
- Alerts and notification management data (e.g. thresholds);
- Prices plans;
- Invoices – credit notes;
- Customer contract and billing information;
- Events like accident, dangerous manoeuvers, start, stop…;
- Trip;
- Metadata and methods that can be migrated from the application to the PaaS, candidates for this migration are: reports, measure calculation, alert and notification mechanism.

The use case we are defining should be based on an architecture where messages are decoded and routed to the back-end logic process (as in the architecture drawing in later subsection 4.1.3) where they are processed and inserted in the PaaS. A web application will then access the PaaS and will provide following services:

- User authentication and authorization;
- User management;
- Entity and device management;

- Alerts and notification management;
- Alerting device management;
- Customer management;
- Prices and Invoice management;
- Dynamic on-line and off-line reporting;
- Dynamic dashboards;
- Real-time presentation of entity behaviour;
- Reconstruction of entity behaviour;
- Task and schedule optimization taking into forecasted/real traffic;
- Eco score calculation, reporting and alerting;
- Safety score calculation, reporting and alerting;
- Fuel management;
- Bus entity management;
- Car renting management;
- Fuel transportation management.

As the amount of devices increases from years to years, we need to improve this infrastructure in order to support scalability, high availability, stability and performances.

Our vehicle telematics application has the following high-level functional requirements:

REQ#1: The system must be able to receive and decode messages at a high frequency

Description: The tracking frequency tends to be higher and higher. Assuming we have 100.000 sensors (vehicle, persons, temperature sensors, seals …), sending one message per second, the platform must handle these messages in nearly real time. One message contains more or less 200 bytes. Every message must be acknowledged by sending a 40-byte message to the device. Decoding of the message provides the following raw information (Vehicle data):

- GPS position (longitude/latitude)
- Timestamp (date time in GMT+00)
- Speed
- Course
- Altitude
- Engine on/off
- Cumulated distance
- Input state
- Driver identification

REQ#2: The system must be able to aggregate and consolidate information and at the end visualize them over dashboards (in real-time)

Description: These information must be consolidated to produce dashboard like:

- Cumulated distance
- Mean speed, max speed
- Driving time, stop time, idling time, …
- Fuel consumed
- Safety scoring

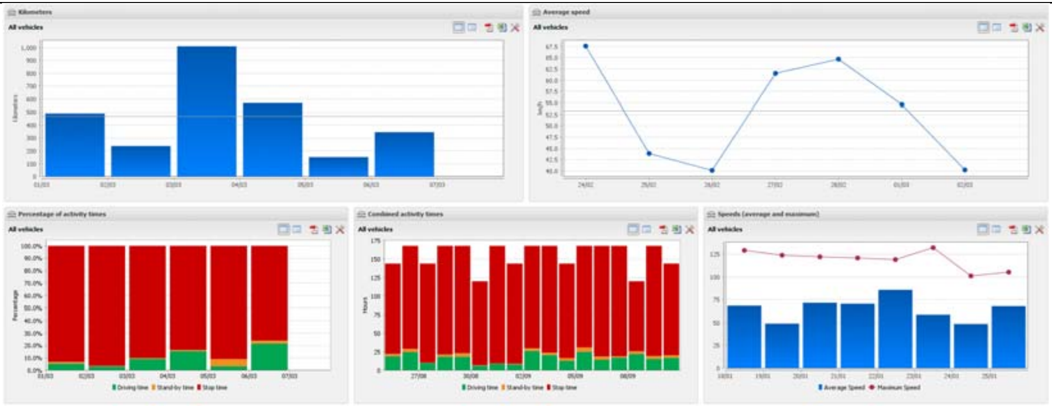


Figure 17: the Vehicle Telematics M2M Case – Dashboard.

Dashboards can be built:

- for all vehicles, a group of vehicles, one vehicle
- for a fixed period (this day, this week, this month, this quarter, …) or for a dynamic period, with a breakdown by hour, by day, week, month, …

Dashboards are summary information that need to popup rapidly. However, dashboards contain aggregate data which are produced asynchronously. In order to produce dashboard data, another component / service runs every night and aggregates information (kms, running hours …). However, dashboards must be updated and available in nearly real time.

REQ#3: The system must be able to deliver the positions to the web interface and reproduce the trips in real time.

Description: The geographical positions must be delivered to the Web UI to produce real time map with the trip of the sensor. The positions are updated in a map as soon as they are received

Many users can be connected to the platform and receive the new positions

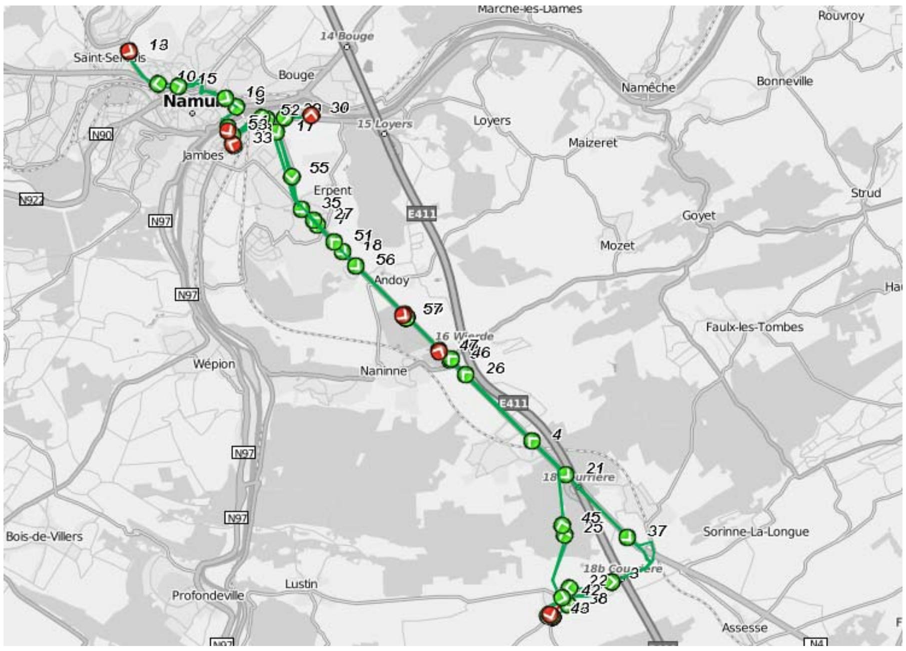


Figure 18: the Vehicle Telematics M2M Case - Mapping New Positions.

REQ#4: The system must be able to monitor sensor status in real-time

Description: The activity (stop, driving, idling, …) of the sensor must also be updated on real time

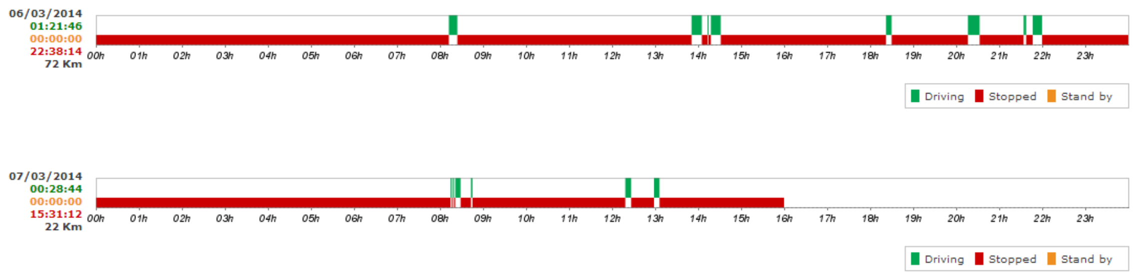


Figure 19: the Vehicle Telematics M2M Case - Monitoring Sensors.

REQ#5: The system must be able to detect alarms and generate notifications

Description: Based on the information contains in the messages, alarms and notifications must be generated in real time.

Alarms can be triggered by the device (speeding, unauthorized movement, unauthorized identification, panic button …) or by the back office based on a combination of criteria's. These ones can be:

- Speed of the vehicle
- State of an input
- Geographical zone (entering, leaving)
- Period of the day

Alarms can also be triggered by comparing historical data with the latest received one, for example:

- The state of an input change from on to off between to messages
- A position is outside a geographical zone, while the next one is in the zone

REQ#6: The system must manage the entities and maintain a hierarchical structure grouping them in logical structure

Description: Entities will be grouped in a hierarchical structure to allow reporting by groups e.g. vehicles grouped by cities, region, country.

9.2.3 Technical Framework

Our current infrastructure is based on these building blocks:

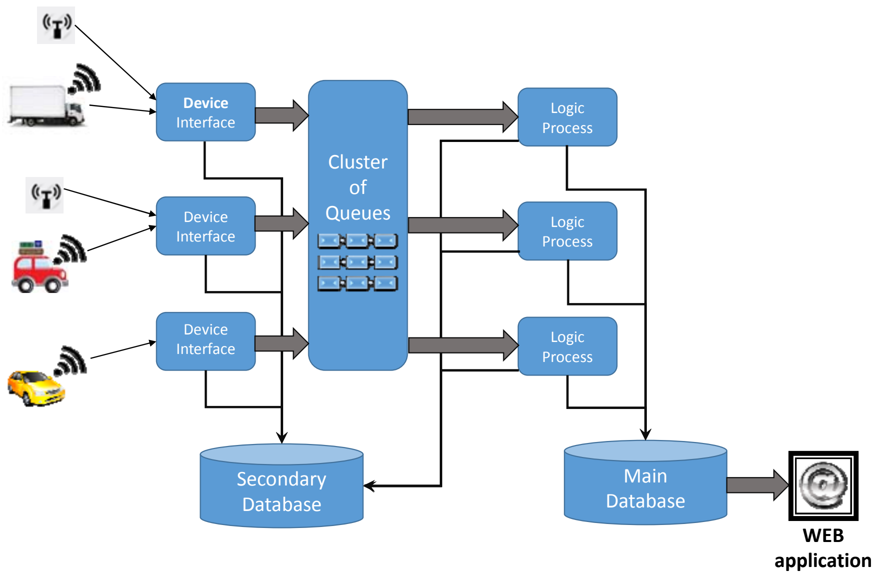


Figure 20: Building Blocks for the Vehicle Telematics M2M Case.

1. Vehicles equipped with devices (GPRS and GPS) or sensors communicate to a TCP application server (Device Interface block in Figure 20)
2. Device interface application handles messages:
    a. Acknowledges the messages based on specific protocol definition
    b. Stores messages in a secondary database (SQL Server) for specific consolidation
    c. Puts the result in queues managed by a cluster of brokers
3. A second application server (Logic Process block in Figure 20) takes messages from the queues:
    a. Decodes the messages based on specific protocol,
    b. handle some logic based on the secondary data store
    c. Stores positions in the main database (SQL Server), creates notifications, alarms, …

The main DB is also used by the web application in order to deliver information to the end user.

Main technologies currently used by the VTP are:

- Broker : ActiveMQ;
- TCP Server : based on Grizzly Framework (Java NIO);
- Enterprise Integration : Apache Camel;
- Dependency Injection : Spring Framework;
- DB : JPA + Hibernate;
- RDBMS : SQL Server.

9.3 Media Planning Use Case

9.3.1 Business Orientation

Nowadays, millions of people communicate each other using social networks (e.g. Twitter, Facebook, WhatsApp, blogs, forums, e-mail, and so on). Daily, these social platforms produce huge amount of data that is very interesting for **marketing purposes**, but cannot be analyzed with traditional technologies.

In the most popular social networks, the analysis of registered messages allow to know the public brand perception and then, perform corrective actions through the social networks to change it, such as designing effective marketing campaigns, analyzing your target audience classifying it by roles or predicting the impact of a given communication with the most influencers in a given topic/community. To sum up, the social networks' data is very important for any company to increase the client conversion rate and sales.

New different data stores have been appearing to support these data magnitudes and perform analytical queries in few seconds. Moreover, a distributed environment has also become necessary to provide a scalable architecture to support daily loads.

Data providers, whose role consist of collecting and selling social data to third parties, are another type of stakeholders involved in this use case. Sometimes, social networks sell their own data, but other times third parties collect data from different communication channels and deliver it with the same format.

So, we can summarize the following actors involved:

- **Community managers**, who analyze social networks and design effective marketing campaigns;
- **Software** (marketing oriented) **vendors**, who design analytical software to improve the decision criteria before starting a new marketing campaign;
- **Cloud systems' providers** to install big data architectures and offer their renting services;
- **Data store's providers** to create and design efficient and scalable solutions to store and retrieve data.

The following picture resumes the described interaction among the actors involved in this use case.

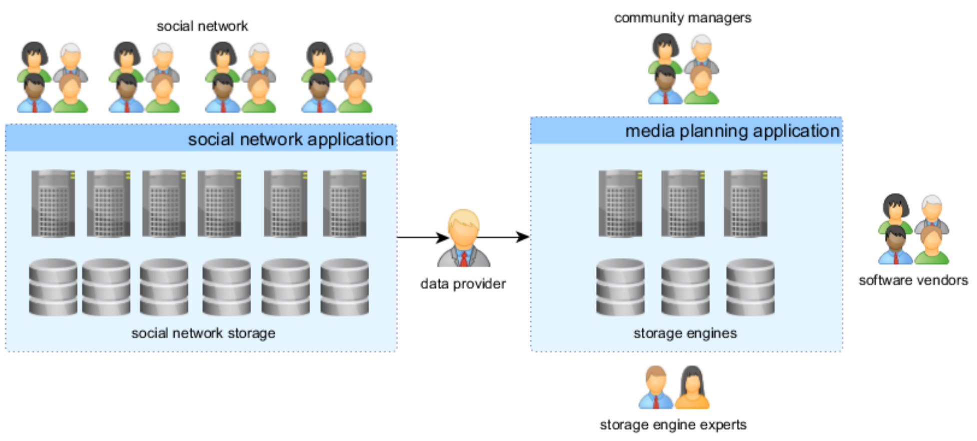


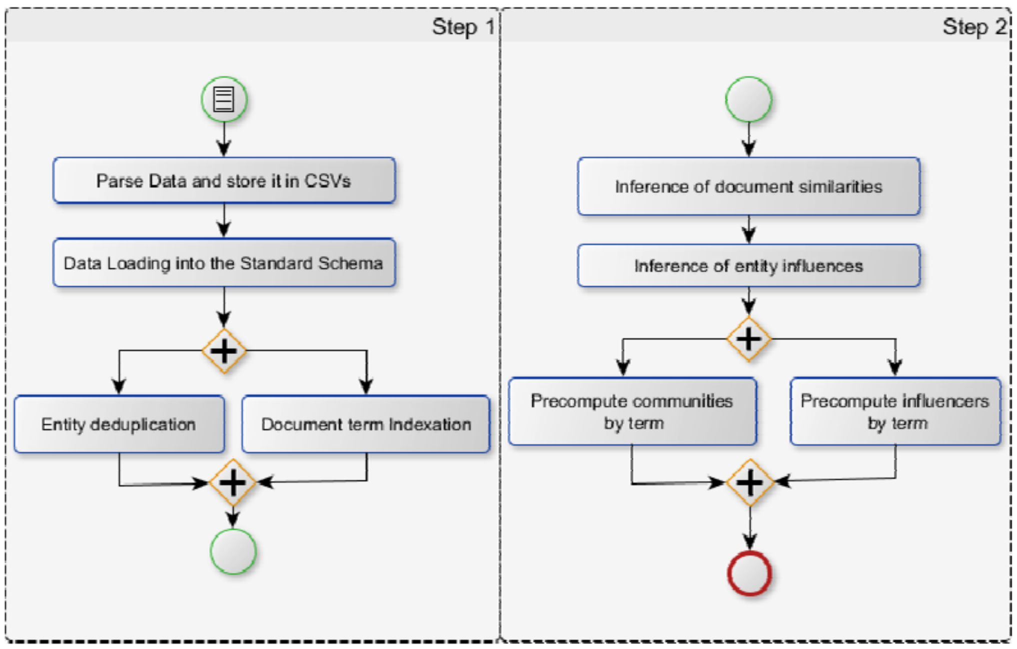
Figure 21: Media Planning Use Case: Interaction among actors.

Sparsity-Technologies has been working for nearly three years with two specialized and international marketing companies with more than 20 years of experience. These companies are **Acceso** and **MPG**. Sparsity-Technologies has transferred to them a deep knowledge to manage huge amounts of data and efficient algorithms to solve common and difficult marketing issues related with social networks using Sparksee. Indeed, Sparsity-Technologies has needed to create a specific technology to support complex media planning queries that are able to work with some millions of documents. Now, CPaaS will offer the chance of having more efficient loading and querying processes distributing the work load on specialized data stores.

Finally, it is interesting to highlight the relevance of the Media Planning clients of Sparsity-Technologies. Acceso is an international company based on Spain and Latin America, which provides: tracking services, content analysis and communication control services. MPG is also an international company based in Spain, France, UK and EEUU, which designs and manages marketing campaigns for important companies such as Nike, Mac Donald's, Coca-Cola or Hunday. At his moment, MPG is working in more than 100 countries.

9.3.2 Functional Description

Any of our media planning application has the following high-level requirements:

REQ#1: The system must be able to load the daily data provided by specific communication channels (i.e. twitter) into a generic schema in less than 12 hours.
Description: Every day our clients usually need to process less than 1 million of documents and their custom queries require to generate and store more inferred information to provide efficient response times. Thus, there is a tradeoff between the queries response time and the required data loading time. The following diagram summarizes some of the processes involved in loading time.
 Step 1 Parse Data and store it in CSVs Data Loading into the Standard Schema Entity deduplication Document term Indexation Step 2 Inference of document similarities Inference of entity influences Precompute communities by term Precompute influencers by term **Figure 22: Media Planning Use Case: Processes Involved.**

REQ#2: The system must recognize which author accounts represent the same entity.
Description: A person might have more than single social network account (e.g. twitter and Facebook) and the system must recognize which accounts are related with the same entity. The analytical queries just will work with entities instead of user accounts.

REQ#3: The system must infer PROPAGATION relationships among documents through their text similarity
Description: A propagation is a fact that happens whether a document contains part of the message of an older document. For example, a tweet created from a partial copy-paste action instead of making an explicit “retweet”. This detection needs to be executed in loading time because is a hard problem to solve and might be thousands of documents to compare.

REQ#4: The system must support incremental loading processes.
Description: Every day, the system must load data and it is necessary making a design of incremental and efficient processes that ensure high quality analytical results. For instance, the word weight for a document never changes although new documents with the same word are inserted. Indeed, it has sense because the document may be very relevant for that day.

REQ#5: The system must resolve which are the most influencers for a set of keywords.
Description: In order to give strategic special offers to clean or improve the brand opinion, it is important to know who are the most influencers because this information gives the ability of change the people’s feelings more effectively through them. Our Media Planning clients measure the influence in terms of how many copies, references, propagations or mentions an entity has been received. The system must rank them for a given topic. The following screenshots show the propagation chains of the most influencers in the topic “social media”.

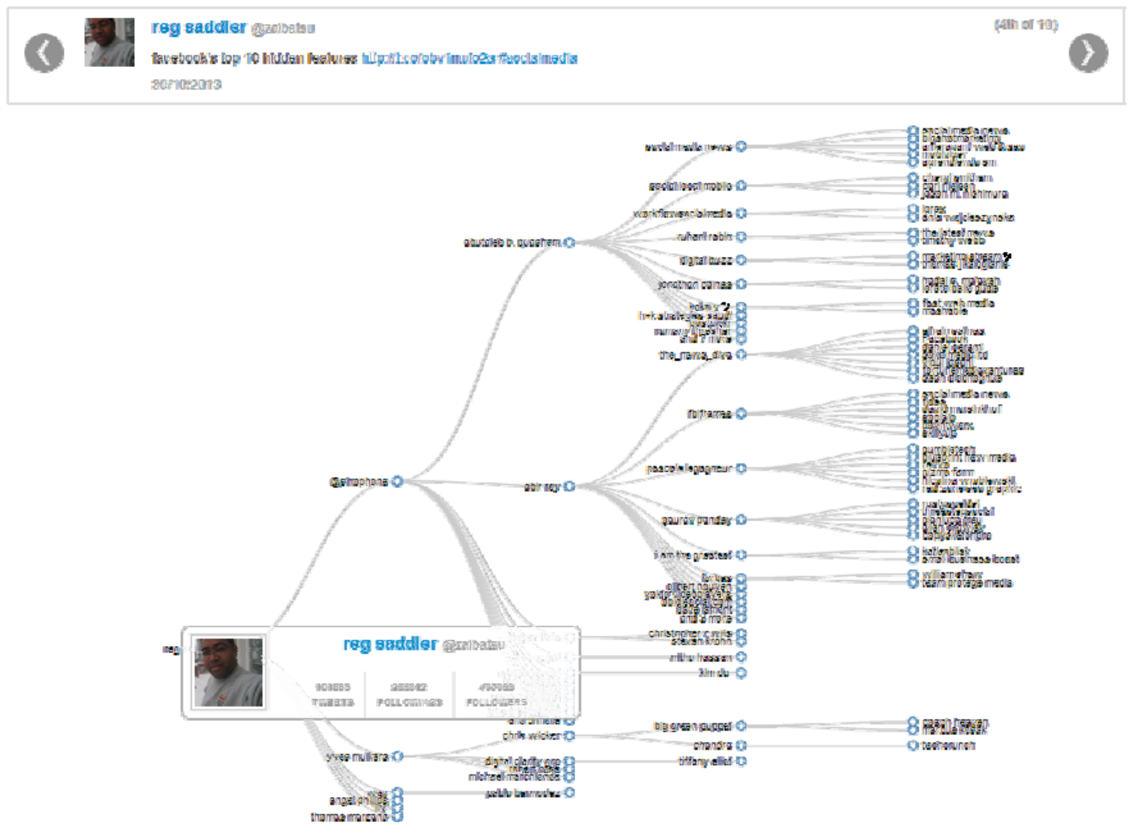


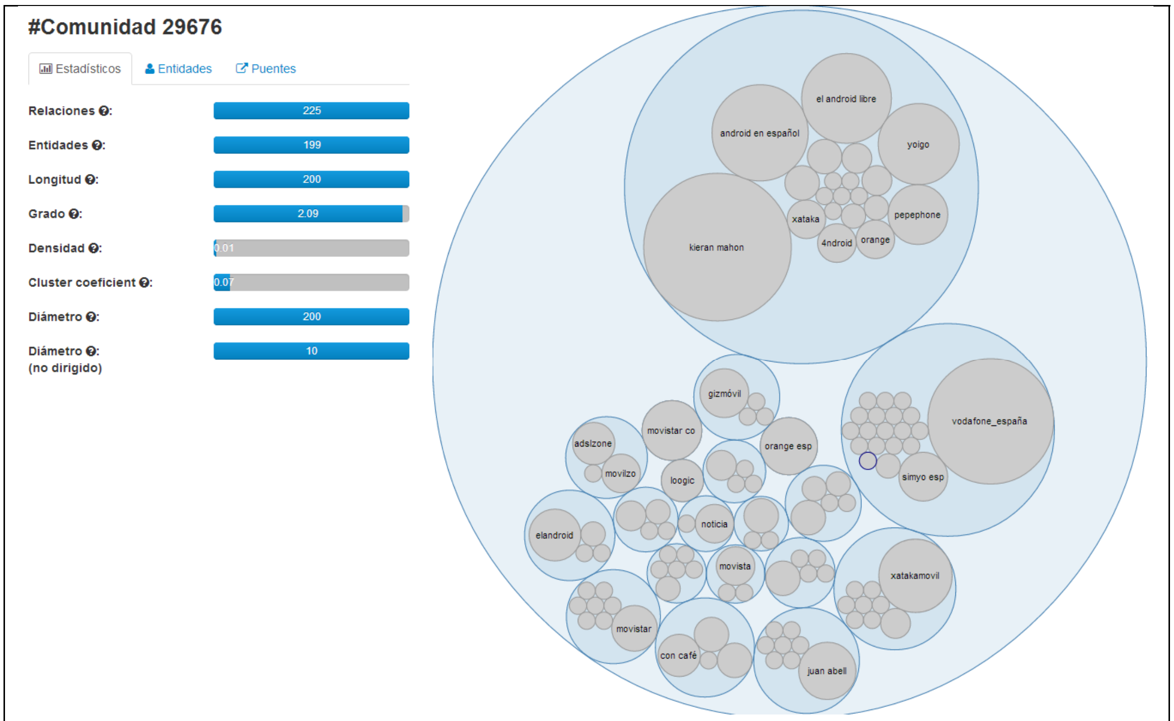
Figure 23: Propagations Chains of influencers.

REQ#6: The system must resolve which the communities for a set of keywords are.

Description: Efficient marketing campaigns are designed for people connected through common interests. Indeed, it is what Facebook did starting from one university to another one. Resolve which are the communities around a topic, allows to classify and understand the companies' target market.

There are many different algorithms for community detection, but in our case, the client fixed which algorithm we had to implement using the influence network. The members of a community must be sorted by the number of produced influences.

The following screenshot shows the communities around the keyword "android" and who are the most influencer entities in each one.



#Comunidad 29676

Estadísticos | Entidades | Puentes

Relaciones:	225
Entidades:	199
Longitud:	200
Grado:	2.09
Densidad:	0.01
Cluster coeficient:	0.07
Diámetro:	200
Diámetro: (no dirigido)	10

Figure 24: Visualisation of Communities.

9.3.3 Technical Framework

The system has the following modules:

- **Loader**: Performs a real time data collection from twitter. This component might use storm to provide a distributed and incremental real-time computation system. The loader will store the basic information related with a document and will index it in the most appropriate way to ensure a high performance in the analytical part;
- **Analytical layer**: It performs analytical queries using the common query engine. These queries are recommendation engines to resolve communities and influencers. We will use the XWork / Struts2 because it is a rich MVC framework with an IoC container;
- **REST layer**: It exposes the analytical layer to the web through a REST API. Multiple servers with the same REST api will be running and a single load balancer will distribute the requests.

The following diagram summarizes the physical and logical view of the described modules.

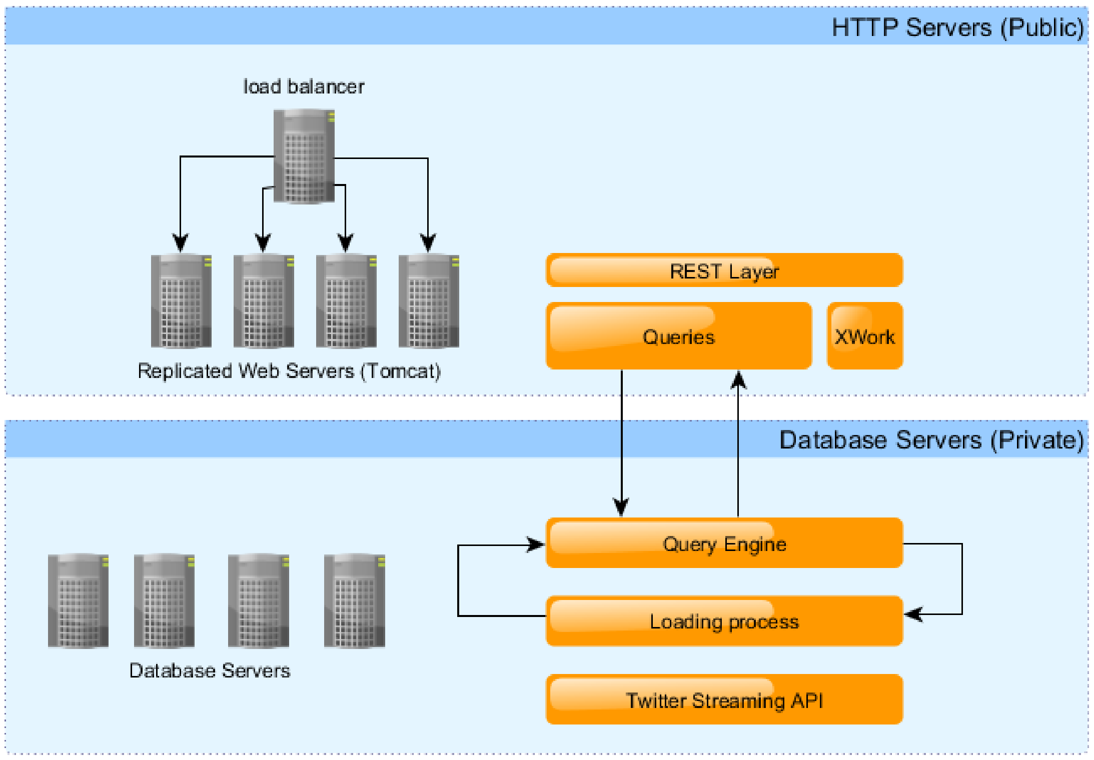


Figure 25: Physical and Logical View of Media Planning Use Case.

9.4 Real-Time Network Performance Analysis in a Telco Environment Use Case

9.4.1 Business Orientation

Any company offering products or services to its customers must compete each and every day, if it is to increase its income stream. It is crucial that companies get the most out of their own resources, so they can maximize operational returns. This is particularly true of telecommunications operators for the following reasons:

- It is one of the sectors in which technological developments have the greatest impact;
- The sheer diversity of technologies, suppliers and services creates a complex operational map;
- The rate at which services are created and then changed demands a swift and adaptive response from the supporting infrastructure;
- The quantity and range of equipment in the network mean installed resources require careful control, including detailed data (for example, on location) and assured availability.

Without any tool to monitor all this equipment and services, telecommunication operators face a difficult problem, which is trying to detect a network or service problem. When faced with one, they can spend several hours performing advanced performance reports and newsletters which lead to:

- Customer complaints;

- The usage of a large number of applications with high cost of ownership, training and maintenance;
- Difficulty to support new services or service upgrades.

To solve this problem a real-time network performance analysis framework was designed, called Altaia, which offers:

- Detection of a large number of network problems before real service degradation or unavailability in real time;
- Proactive supervision;
- A reduced number of customer complaints;
- Completely E2E capability to solve service problems;
- Unified and centralized system for performance management with smooth integration with other operation support systems (OSSs);
- Reduced costs and an increase in customer satisfaction and quality of experience (QoE).

Altaia is a centralized and unified End to End performance management system for multi-service, multi-technology and multi-vendor environment whose main function is to provide operations, marketing and planning managers with the information they need on network and service performance to be able to make technical/business decisions.

These functionalities enable the telecommunication operator to perform insightful analysis of the network, services and customer performance data, assure E2E performance management, ensure that the network and services operate at peak performance and availability, provide analytic forecasting capabilities and a powerful troubleshooting and root cause analysis of performance degradation. All these features together make Altaia a framework that is:

- An integrated enterprise wide E2E performance management system;
- An highly configurable framework;
- One of the industry most advanced threshold & metrics engine;
- Multi-vendor and multi-technology;
- Web based, failover/high-availability and with a scalable architecture.

9.4.2 Functional Description

The Altaia system is an integrated E2E performance management system, multi-vendor and multi-technology, that actively detects deterioration in a network using almost real-time KPIs and KQIs, finds the cause of performance problems and the produced data is always ready to be analyzed through reports and dashboards to check the network's performance. These reports can be tailored to further analyze the network's performance by creating ad-hoc queries and accessing the data that originated the calculated indicators. An overall view of the system is presented in an image in the end of this sub-section.

To store all the data to support these functionalities, two sets of data stores were designed: the **DBN0**, which stores the raw data collected from the network, and the **DBN1**, which stores the KPIs and KQIs calculated from the raw data.

The system's main functionalities, therefore, are:

- Collect data from various sources in the network, homogenizing and enriching the data into a standard format and storing it in batches in the DBN0 for later processing and analysis. The data is also parsed to detect and prevent errors, which are logged;

- Performing CEP on the collected data enabling time series pattern deviation detection and the assembly of records. Definition of metrics dictionaries for the whole company, threshold violation levels (SLAs) and performance and QoS metrics;
- Aggregate collected data in well-defined periods of time to calculate the KPIs and KQIs, which are used to generate alarms and detect pattern deviations over fixed or dynamically configured thresholds; computed data is then batch loaded to the DBN1;
- Define DBN0 inventory (configure the data to collect), DBN1 model and DBN0 to DBN1 mapping (map which data from the DBN0 is used, the process of calculating the KPIs and KQIs and where should they be stored in the DBN1);
- Use the data in the DBN1 to generate reports (system defined or custom built), real-time dashboards and access the data in the DBN0 for more detailed information (drill-to-detail);
- Manage, monitor and report on SLA compliance;
- Notify other systems of detected alarms;
- Display geo-referenced performance indicators to allow users to visually simultaneously analyse different indicators for various geographical entities;
- Support for the management of network upgrade or maintenance activities;
- Support for predictive analysis based on forecasts and trending techniques aimed at predicting any degradation of networks and/or services;
- The DBN0 data stores are a staging point for the collected data, which is retained for resubmission and detailed querying;
- The DBN1 data stores temporal aggregated data; data aging is used to further aggregate older data; a star schema is used for improved performance.

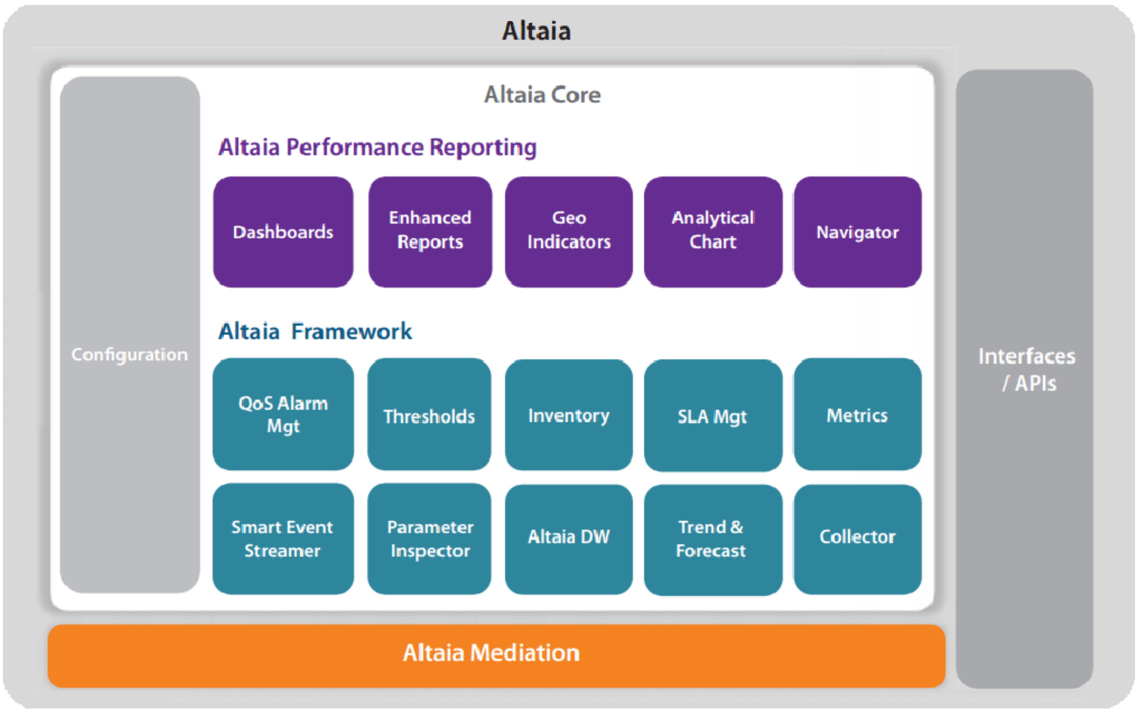


Figure 26: Altaia Functionalities.

9.4.3 Technical Framework

The Altaia system (which the current architecture is shown below) is composed of two data store layers, DBN0 and DBN1, which stores raw data collected from the network and the calculated KPIs and KQIs, respectively.

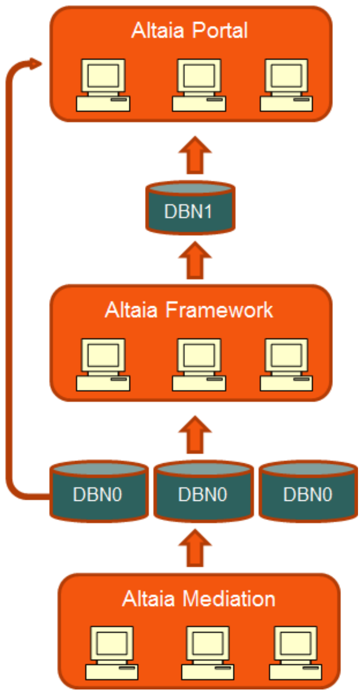


Figure 27: Altaia System Architecture.

The system is also composed from following modules:

- **Altaia Mediation:** The Altaia Mediation module is a scalable and robust abstraction layer that allows the Altaia system to access the data from the various sources in the network (nodes, servers etc.), manufacturers and technologies (SNMP, SIP, XML, CLI etc.) in a standard way. The module uses a set of plugins that read the data from each data source, transforms it into the normalized formats used by the Altaia system and ensure data quality (e.g. data must comply to a list of possible values of a field and block or correct the values that do not fit). It is also responsible for enriching the data with dimensional information (e.g. use cell ID to define record call the geo-location or infer classification of “on net” or “off net” from the calling and called number), performs the data parsing to identify errors and to log them and is able to perform self-discovery of network entities;
- **Altaia Correlation:** The Altaia Correlation module is under development (some of its functionalities are currently being performed inside the Altaia Framework) and is currently using Esper (CEP engine) to correlate data provided by the Altaia mediation module and assemble records. The event correlation can be used to, for example, infer that calls are being dropped in the network, by correlating multiple call events in a short period of time. Correlated events generate new complex events that are stored in the DBN0 and can be used by the Altaia Framework and Portal. The generated complex events can also be fed directly to the Altaia Framework to generate alarms;
- **Altaia Framework:** The Altaia Framework module is the point where all the KPIs and KQI are produced and transformed for several types of analyses. It reads the data stored in the DBN0s and other metrics stored in the DBN1, in batch in well-

defined periods of time and the produced KPIs and KQIs are then stored in batch into the DBN1, where they are accessible to the Altaia Portal module. During KPIs and KQIs calculation, this module is also responsible for evaluating pattern deviations through fixed and dynamic thresholds using the Thresholds component. If new data arrives at the DBN0 for a time period already processed, it detects it and calculates and resubmits the new KPIs and KQIs to the DBN1. It also has a dashboard to configure the data stores, provided by the Framework Configuration, and the Metrics component allows defining new KPIs and KQIs. The SLA Management component manages monitors and reports on agreed SLA compliance. The planning component is responsible for predicting future data behavior or indicator trends and the QoS Alarm Management detects alarms and notifies other systems of these;

- **Altaia Portal:** The Altaia Portal module is responsible for analyzing the key indicators from the DBN1 data stores and presents them in reports (already defined - basic reports- or custom made with ad-hoc queries -enhanced reports) or in real-time dashboards. Through the Navigator component, it also enables drill-to-detail querying of DBN0 data stores when the analyst needs to consult the data that generated the key indicators. The Geo Indicators component is responsible for displaying geo-referenced performance indicators.

9.5 Bibliographic Search Use Case

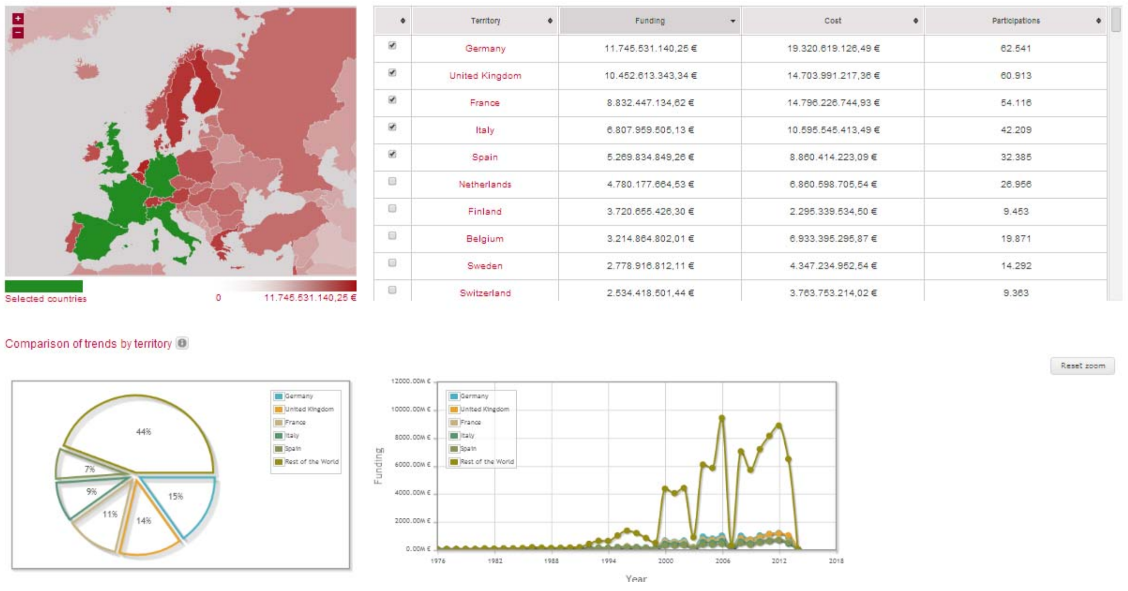
9.5.1 Business Orientation

Researchers are experts in different areas and they produce papers, patents and collaborate among them In European projects, which are economical resources to extend their knowledge in a specific research area. European projects offer chances to transfer knowledge between different organizations.

Bibliographic data stores collect the researchers' knowledge and are important resources to find experts and institutions working in specific research areas. Some of these bibliographic data stores are DBLP, which contains all papers about computer science; CORDIS which contains all European projects; or SCOPUS, which is one of the biggest bibliographic data stores. Let's see each one of these data stores in more detail:

- DBLP, the computer science data store, contains over 1.8 million publications, 1 million authors and several thousands of journal or conference proceedings series. Its data is completely open and it is provided in XML format. However, DBLP publications have got neither abstracts (because these are subject to the same copyright restrictions as full-text of research papers) nor cross references, but it has specialized people which verifies all information to assure unique authors;
- The SCOPUS (Elsevier) bibliographic data store it isn't open data solution, but contains, by far, much more publications than DBLP because it is not restricted to the computer science subject. Indeed, it is one of the most complete bibliographic data stores: it has over 50 million publications with abstracts and cross references, over 21 thousand journals or conference proceeding series. However, in contrast with DBLP publications' authors are duplicated. In other words, it would miss publications for a given author identifier;

- The CORDIS is the European Commission's primary public repository and portal to disseminate information on all EU-funded research projects and their results in the broadest sense. It contains 40 thousand projects approximately. Sparsity-Technologies has worked with its dataset and have detected some drawbacks: projects and participants are duplicated and the members of each participant are not included. Sparsity has built a web application called Sciencea which solves this problem by means an ad-hoc deduplication engine. The following screenshot allows to know the economic trends of each European country and the top companies that has acquired more funding.



	Territory	Funding	Cost	Participations
✓	Germany	11.745.531.140,25 €	19.320.619.126,49 €	62.541
✓	United Kingdom	10.452.613.343,34 €	14.703.991.217,36 €	60.913
✓	France	8.832.447.134,62 €	14.796.226.744,93 €	54.116
✓	Italy	6.807.959.505,13 €	10.595.545.413,49 €	42.209
✓	Spain	5.269.834.849,26 €	8.860.414.223,09 €	32.385
☐	Netherlands	4.780.177.664,53 €	6.860.598.705,54 €	26.956
☐	Finland	3.720.655.426,30 €	2.295.339.534,50 €	9.453
☐	Belgium	3.214.864.802,01 €	6.933.395.295,87 €	19.871
☐	Sweden	2.778.916.812,11 €	4.347.234.952,54 €	14.292
☐	Switzerland	2.534.418.501,44 €	3.763.753.214,02 €	9.363

Figure 28: Example of a Bibliographic Search.

Mainly, bibliographic data stores are used by researchers to know the state of the art of a given topic. However, they can be used for many other purposes such as looking for reviewers, collaboration, similar papers or works and so on.

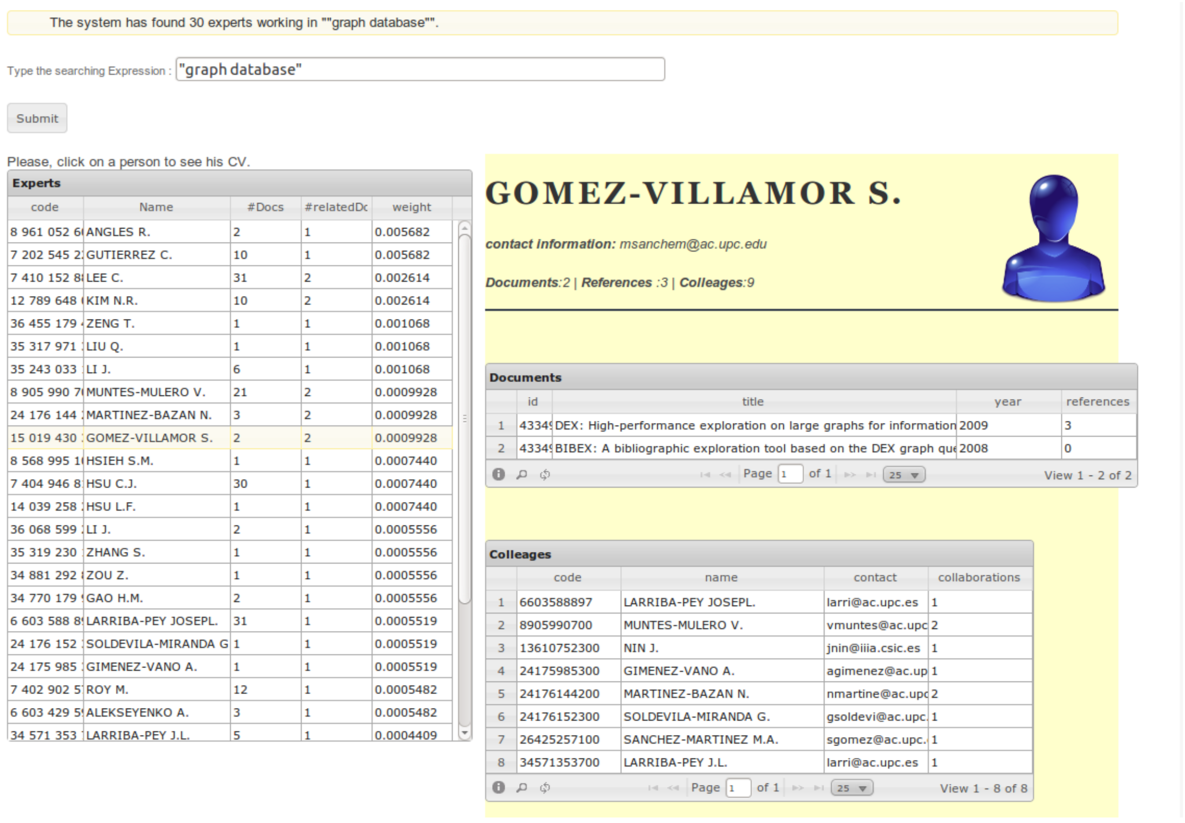
Moreover, improvements in the quality of the bibliographic data, offers a wide range of possibilities such as analysis about the evolution of a European partner or country, which are deeply interesting for the European Commission.

Sparsity technologies and INRIA want to go one step forward with Sciencea, trying to merge the contents of CORDIS and DBLP to offer new analytical queries such as: which are the projects probably related with a given article or who is a good reviewer for a given European project.

9.5.2 Functional Description

There are the main functionalities provided by the use case.

- Calculate the best reviewers for a given European project. The following screenshot shows a software previously developed by Sparsity-Technologies for the Spanish government to search experts/reviewers for a specific topic;



The system has found 30 experts working in ""graph database"".

Type the searching Expression : "graph database"

Submit

Please, click on a person to see his CV.

Experts

code	Name	#Docs	#relatedDc	weight
8 961 052 6	ANGLES R.	2	1	0.005682
7 202 545 2	GUTIERREZ C.	10	1	0.005682
7 410 152 8	LEE C.	31	2	0.002614
12 789 648	KIM N.R.	10	2	0.002614
36 455 179	ZENG T.	1	1	0.001068
35 317 971	LIU Q.	1	1	0.001068
35 243 033	LI J.	6	1	0.001068
8 905 990 7	MUNTES-MULERO V.	21	2	0.0009928
24 176 144	MARTINEZ-BAZAN N.	3	2	0.0009928
15 019 430	GOMEZ-VILLAMOR S.	2	2	0.0009928
8 568 995 1	HSIEH S.M.	1	1	0.0007440
7 404 946 8	HSU C.J.	30	1	0.0007440
14 039 258	HSU L.F.	1	1	0.0007440
36 068 599	LI J.	2	1	0.0005556
35 319 230	ZHANG S.	1	1	0.0005556
34 881 292	ZOU Z.	1	1	0.0005556
34 770 179	GAO H.M.	2	1	0.0005556
6 603 588 8	LARRIBA-PEY JOSEPL.	31	1	0.0005519
24 176 152	SOLDEVILA-MIRANDA G	1	1	0.0005519
24 175 985	GIMENEZ-VANO A.	1	1	0.0005519
7 402 902 5	ROY M.	12	1	0.0005482
6 603 429 5	ALEKSEYENKO A.	3	1	0.0005482
34 571 353	LARRIBA-PEY J.L.	5	1	0.0004409

GOMEZ-VILLAMOR S.

contact information: msanchem@ac.upc.edu

Documents:2 | *References* :3 | *Colleages*:9

Documents

	id	title	year	references
1	4334	DEX: High-performance exploration on large graphs for information	2009	3
2	4334	BIBEX: A bibliographic exploration tool based on the DEX graph que	2008	0

Page 1 of 1 25 View 1 - 2 of 2

Colleages

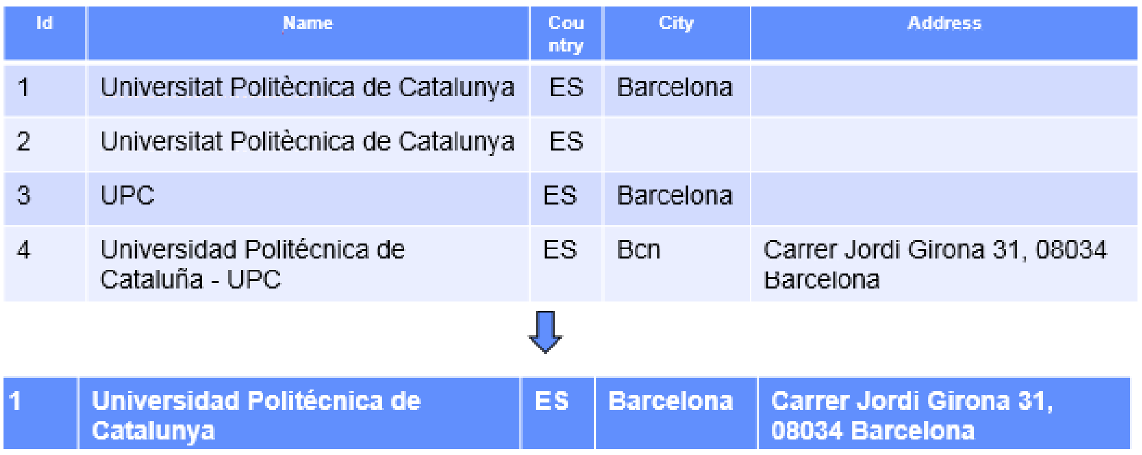
	code	name	contact	collaborations
1	6603588897	LARRIBA-PEY JOSEPL.	larri@ac.upc.es	1
2	8905990700	MUNTES-MULERO V.	vmuntes@ac.upc	2
3	13610752300	NIN J.	jnin@iiia.csic.es	1
4	24175985300	GIMENEZ-VANO A.	agimenez@ac.up	1
5	24176144200	MARTINEZ-BAZAN N.	nmartine@ac.upc	2
6	24176152300	SOLDEVILA-MIRANDA G.	gsoldevi@ac.upc.	1
7	26425257100	SANCHEZ-MARTINEZ M.A.	sgomez@ac.upc.	1
8	34571353700	LARRIBA-PEY J.L.	larri@ac.upc.es	1

Page 1 of 1 25 View 1 - 8 of 8

Figure 29: Example of a Bibliographic Search: Best Reviewers' Search.

- Calculate the most related European projects with a given article;
- The system must allow to add new reviewers (without documents) with a set of keywords to define their specialization;
- Loading process of projects and papers periodically.

To expose these functionalities, it will be necessary to de-duplicate the CORDIS projects and institutions. One example that shows how dirty the information is shown as follows:



Id	Name	Country	City	Address
1	Universitat Politècnica de Catalunya	ES	Barcelona	
2	Universitat Politècnica de Catalunya	ES		
3	UPC	ES	Barcelona	
4	Universidad Politécnica de Cataluña - UPC	ES	Bcn	Carrer Jordi Girona 31, 08034 Barcelona

1	Universidad Politécnica de Catalunya	ES	Barcelona	Carrer Jordi Girona 31, 08034 Barcelona

Figure 30: Example of "Dirty" Information.

9.5.3 Technical Framework

The system has the following modules:

- **Crawlers:** Perform periodically crawling processes for CORDIS and DBLP;

- **Loader**: Performs a periodically loading processes for new E.U. projects and papers. This component might use storm to provide a distributed and incremental computation system. The loader will store the basic information related with a project and will index it in the most appropriate way to ensure a high performance in the analytical part;
- **Analytical layer**: It performs the analytical queries using the common query engine. These queries are recommendation engines to resolve reviewers and related European projects. We will use the XWork / Struts2 because it is a rich MVC framework with an IoC container;
- **REST/Web layer**: It exposes the analytical layer to the web through a REST API. Multiple servers with the same REST api will be running and a single load balancer will distribute the requests. The results will be integrated into Sciencea web application.

The following diagram summarizes the physical and logical view of the described modules.

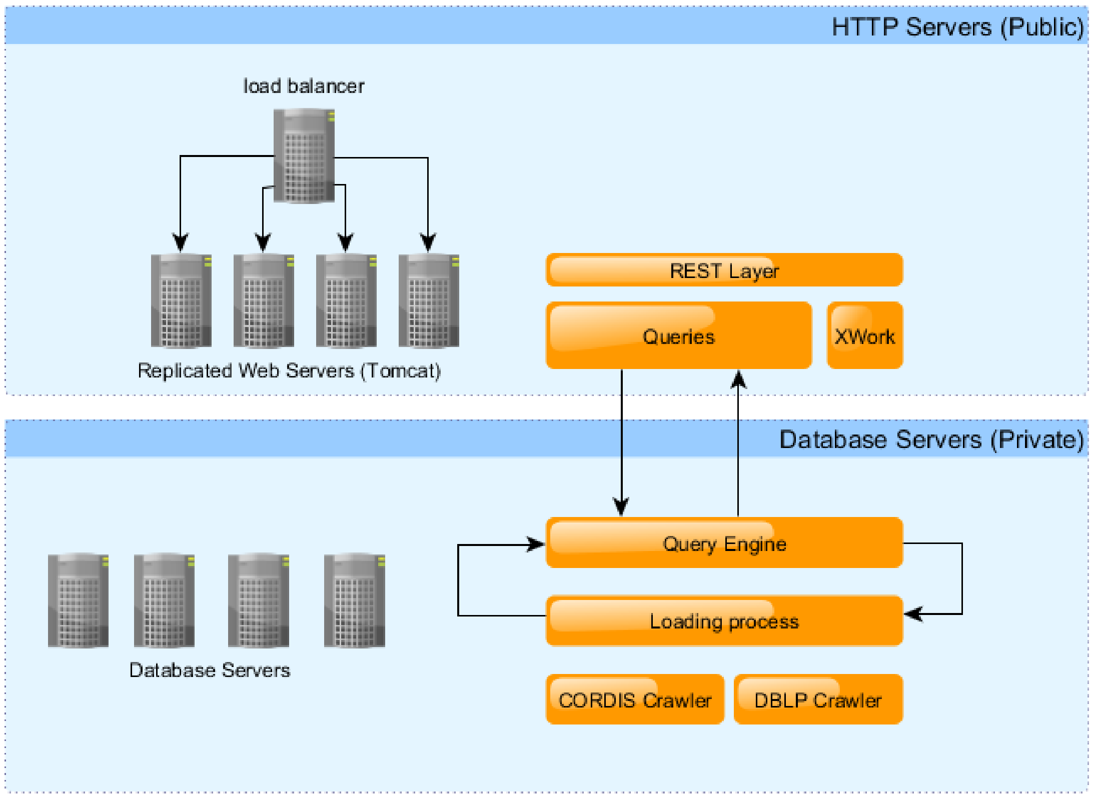


Figure 31: Physical and Logical View of Bibliographic Search.