Automatic, multi-grained elasticity-provisioning for the Cloud

Cloud Information System

Deliverable no.: 4.3
May 28th, 2015

CELAR is funded by the European Commission DG-INFSO
Seventh Framework Programmed, Contract no.: 317790

Table of Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

DB	Database
IS	Information System
JSON	JavaScript Object Notation
MS	Monitoring System
OS	Operating System
REST	Representative State Transfer
UI	User Interface
VM	Virtual Machine
WP	Work Package
XML	eXtensible Markup Language
JAR	Java ARchive file format
WAR	Web ARchive file format
SMA	Simple Moving Average

Deliverable Title	**Cloud Information System**
Deliverable No.	**4.3**
Filename	CELAR_D4.3_finalrelease.docx
Author(s)	Athanasios Foudoulis,Demetris Trihinas, Demetris Andoniades, Nicholas Loulloudes, Stalo Sofokleous, George Pallis, Marios D. Dikaiakos, Daniel Moldovan, Hong Linh Truong, Vangelis Floros , Giannis Giannakopoulos
Date	29.05.2015

Start of the project: 1.10.2012
Duration: 36 Months
Project coordinator organization: ATHENA RESEARCH AND INNOVATION CENTER IN INFORMATION COMMUNICATION & KNOWLEDGE TECHNOLOGIES (ATHENA)

Deliverable title:	Cloud Information System
Deliverable no.:	4.3

Due date of deliverable: 30 May 2015
Actual submission date: 29 May 2015

Dissemination Level

X	PU	Public
	PP	Restricted to other programme participants (including the Commission Services)
	RE	Restricted to a group specified by the consortium (including the Commission Services)
	CO	Confidential, only for members of the consortium (including the Commission Services)

Deliverable status version control

Version	Date	Author
0.0	04.05.2015	Athanasios Foudoulis
0.1	11.05.2015	Athanasios Foudoulis
0.2	13.05.2015	Athanasios Foudoulis, Demetris Trihinas
0.3	16.05.2015	Athanasios Foudoulis, Demetris Trihinas, Demetris Antodiades
0.4	20.05.2015	Athanasios Foudoulis, Demetris Trihinas, Demetris

		Antodiades, Nicholas Loulloudes, Stalo Sofokleous, George Pallis, Marios D. Dikaiakos, Daniel Moldovan, Hong Linh Truong
1.0	28.05.2015	Athanasios Foudoulis, Demetris Trihinas, Demetris Antodiades, Nicholas Loulloudes, Stalo Sofokleous, George Pallis, Marios D. Dikaiakos, Daniel Moldovan, Hong Linh Truong, Vangelis Floros, Giannis Giannakopoulos
1.1 (minor update)	07.01.2016	Demetris Trihinas

Abstract

This deliverable presents a thorough report on the design, research and implementation status of the final version of the Cloud Information and Performance Monitor layer, which consists of three, integrated, components: (i) the CELAR Monitoring System; (ii) the Multi-Level Metrics Evaluation Module; and (iii) the CELAR Information System. Specifically, this document focuses on the second version (v2.0) of the CELAR Information System, describing the final design, documenting the implementation decisions taken during the last year, and any performance improvement actions that were taken. Additionally, it presents the visualization interface provided to the user. Finally, it provides an extensive evaluation study, documenting how the Information System requirements were verified to address their assigned purposes.

Keywords

Cloud Computing, Elasticity, Cloud Monitoring, Real-Time Monitoring, Application Monitoring, Cost-Evaluation, Information System, JCatascopia, MELA

1 Introduction

CELAR enhances the state-of-the-art in cloud computing by providing service providers with a fully-automated platform, which incorporates intelligent decision-making algorithms to support multi-dimensional and multi-grained elasticity control. CELAR evaluates elasticity at multiple levels of the cloud stack and considers the impact of enforced actions in relation to cost, quality and allocated resources for elastic cloud services. However, to support such a complex, automated and intelligent platform, as well as, provide this functionality in a vendor-neutral manner, real-time monitoring and elasticity behaviour analysis of heterogeneous types of information collected from different and multiple data sources, is required [Trihinas, 2014a]. This important role, in the CELAR software stack, is the job of the Cloud Information and Performance Monitor Layer which is comprised of the three components developed under the umbrella of WP4.

The Cloud Information and Performance Monitor Layer runs alongside all the layers of the cloud infrastructure, as depicted in Figure 1, which presents the CELAR System architecture, in order to provide the intelligent decision making mechanisms of the CELAR System with real-time, cost-enriched, monitoring metrics.

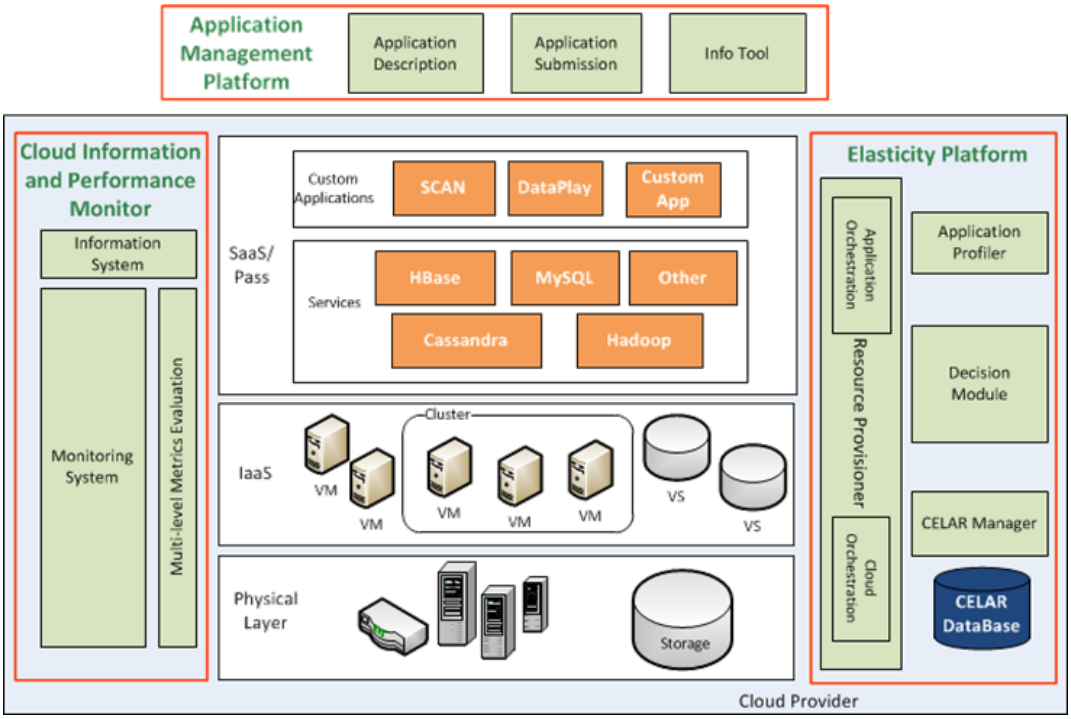


Figure 1: CELAR Architecture

The 3 components comprising the Cloud Information and Performance Monitor Layer, are: (i) The **CELAR Monitoring System**, which is a fully-automated, multi-layer and platform independent monitoring system that runs in a non-intrusive and transparent manner to any underlying virtualized infrastructure; (ii) The **Multi-Level Metrics Evaluation Module**, which provides composable cost-related monitoring information, by mapping collected monitoring metrics to the structure of the application and then enriching cost-wise the obtained metrics based on the cloud provider's pricing schemes; and (iii) The **CELAR Information System**, which provides users with high-level analytic insights to their deployed cloud applications based on collected

monitoring information, cost-enriched metrics, as well as, cloud-platform specific information, which is used to evaluate the behaviour of a specific deployment and even compare multiple deployments in relation to performance, quality and cost.

During the span of the second year of the CELAR project, the development of the CELAR Monitoring System and the Multi-Level Metrics Evaluation Module has ended with all the requirements for both systems, being successfully accomplished and evaluated, as documented in Deliverable D4.2 [D4.2]. Thus, this Deliverable focuses on the second and final version of the CELAR Information System, which, in turn, concludes the work of WP4.

1.1 Purpose of this Document

In this deliverable, we introduce the second version of CELAR Information System (Task 4.3) which documents the effort in WP4 from month 24 to month 32. We present the newly implemented features, the updates to the existing ones and we provide a final documentation report. Moreover, we provide an evaluation study for the CELAR Information System, which documents how we verified the listed requirements to address their assigned purposes.

Note: As this deliverable documents the second version of the CELAR Information System, which was initially described (design and implementation) in D4.2 [D4.2]. We refer readers with no previous knowledge of the CELAR System in general, to D4.2. To enhance readability, a short overview of the CELAR Information System is provided in Section 3. Furthermore, when referring to concepts, which were described in D4.2, a reference is provided to the respected section in D4.2 where it was first introduced.

1.2 Document Structure

The rest of the deliverable is structured as follows: Section 2 presents the updated study of the State-of-Art in Cloud Information Systems, which was initially presented in D4.2. In Section 3 we give an overview of the CELAR Information System and its features. Section 4 presents the work done during the second year (month 24 to month 32) of the task 4.3, the updates and the new functionality we have implemented. At Section 5, we provide the documentation report of the CELAR IS, explaining the distribution packages we use and the installation process. The evaluation study of the CELAR IS is presented in Section 6. Section 7 concludes the deliverable and outlines our future work.

2 State of the Art

In [D4.2] we presented both commercial and scientific works that target cloud resource usage and cost analytics, and information tools. In the duration of the second year, we have expanded our bibliographic research including papers or systems that try to solve related problems of data analytics and time series, are focused on the importance of various metrics towards cloud application optimization and works on time series visualizations. We document our findings below.

While there are many available commercial tools, to the best of our knowledge, there is no work originating from the scientific community that specializes in cloud application analytics. There are several ones trying to solve related problems of data analytics and time series and works focused on time series visualizations. Rainmon [Shafer] is a framework proposed by Shafer et al. which handles bursty monitoring data and calculates analytics over it. Rainmon's analytics engine is based on python and RDDtool[1] and can provide results to the user in a timely manner. The user, through the GUI can view the data of any available machine and whatever metric has been collected for it. An additional analysis procedure running in the background finds and gives access to the user to machines that have similar behavior as the one the user currently views. To support the time series analysis, many libraries and frameworks have been proposed which are not limited to monitoring data. Kurbalija et al. present in [Kurbalija] a Java library capable of handling the most common statistical procedures over time series data. TSAaaS [Xu] is a general purpose time series analytics as a service tool which is not limited to cloud service data. The aforementioned frameworks, target mostly the analysis process of the time series data. Another important part of the analytics is the visualization of data. Kerren et al. [Keim] present a comprehensive research on visual analytics and their characteristics. From a less scientific point of view, Few et al. [Few], give an overview of possible visualization techniques over time series data. The methods presented in [Steinarsson][Jugel] are trying to optimize the visualization process by reducing the amount of data the tool needs to digest using data-reduction and down-sampling techniques. In CELAR IS we are using some of these techniques in order to improve the process of the visualization rendering and provide better visual results to the Application Users [Section 4.1].

[1] https://oss.oetiker.ch/rrdtool/

3 Overview

In Deliverable 4.2 [D4.2] we introduced the CELAR Information System, which is capable of providing, per cloud application, usage analytics by processing information regarding resource utilization and cost-enriched monitoring data, collected during the lifecycle of an elastic cloud service deployment. Moreover, the CELAR IS enables Application Users with the ability to query, explore, compare, and visualize historic data collected from past deployments and previous versions of a cloud service. Table 1 provides an overview of the CELAR IS features and functionality, as defined in the previous deliverable.

<table>
<tr><th colspan="2">Explore and navigate through CELAR data</th></tr>
<tr><td colspan="2">Raw Access and Visualization of data stored in CELAR
Application and Deployment information
Monitoring Metrics
Resizing Actions Enforced</td></tr>
<tr><th colspan="2">Cloud Usage and Cost Analytics</th></tr>
<tr><td rowspan="3">Performance</td><td>The usage of an entity[2]</td></tr>
<tr><td>The state of the entity over time and whenever is applicable</td></tr>
<tr><td>Visualizations intercepting the data input rate (e.g. requests/sec) and the entity usage (e.g. CPU, memory usage).</td></tr>
<tr><td rowspan="2">Cost</td><td>The per hour/month/year cost of an entity,</td></tr>
<tr><td>The change of an entity’s cost over the time.</td></tr>
</table>

Table 1: CELAR Information System Features

The CELAR IS components, reside on the CELAR Server alongside with the CELAR Manager Component. As we depict in [D4.2, Section 3.4] the IS in most of the cases, interacts with the CELAR Manager to obtain data both for calculating analytics and presenting it to the Application User. Moreover, CELAR IS in several occasions interacts, with the underlying CELAR Monitoring System (JCatascopia) and the Multi-Level evaluation module to access monitoring and enriched monitoring data for a running deployment. Figure 2 presents the High-Level architecture of CELAR IS.

[2] As an entity we imply one of the following: (i)a resource (ii) a component, (iii) a group of components or (iv) a deployment

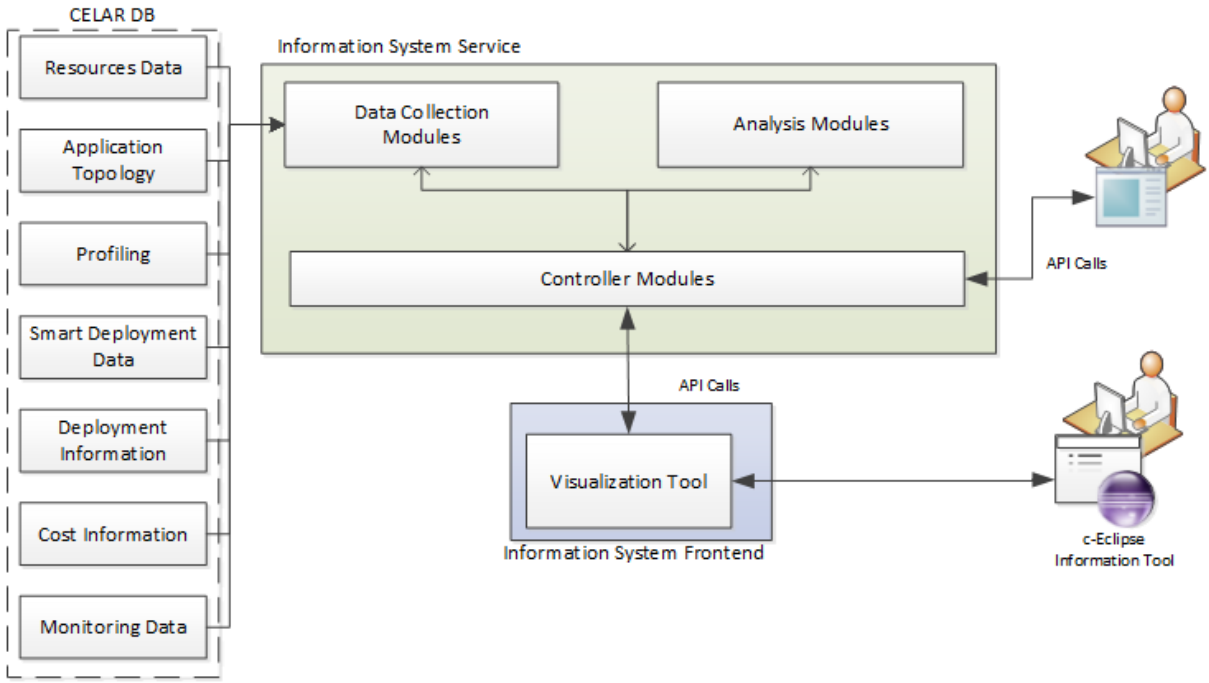


Figure 2: CELAR Information System High-Level Architecture

The CELAR IS consists of two high-level components, the Information System Service and the Information System Frontend. Each one is a separate application, which is distributed in its own package. The communication between these components is achieved through the CELAR IS REST API that the first one exposes. Furthermore, any other interested party can use the REST API to get data from the CELAR IS. The Information System Service handles the computational tasks of the CELAR IS and it includes the (i) Data Collection Modules, (ii) the Analysis Module and the (iii) Controller modules. Each module is further separated, to sub-modules and units, as we describe in Section 3.3 [D4.2], which are responsible for the actual Information System Service functionality. The Information System Frontend is responsible to visualize the data and provide the means for the user to interact with it. The Application User can access this functionality through the c-Eclipse Information Tool.

4 New Features and Improvements

Within the duration of the third year of the CELAR project, we made several changes to the CELAR IS implementation. We describe the most significant changes in the following sub-sections. Additionally, we made minor modifications targeting to solve issues / bugs that aroused, as well as, improve the performance of individual components and the system as a whole.

The most significant modification relates to how the CELAR IS Service runs on the host system. In its first version, the CELAR IS Service was running as a web application (.war file) and it required, for the host system, to have installed the appropriate container (e.g. Apache Tomcat). We changed that and now the CELAR IS Service runs as a native Java application (.jar). The only requirement for the host system is to have Java 7 or greater installed. The CELAR IS Service runs as a daemon, listening for user requests using its REST API. For the REST API to work and expose the desired functionality through the HTTP interface for the users to connect, the CELAR IS contains an embedded tomcat server. The main advantage of having an embedded httpd server is that is not required from the system administrator to install one on the targeted system. This enhances our tool in terms of portability since its only requirement is Java. A second advantage is that the tool is self-contained and it has full access and the ability to modify the tomcat parameters as per its needs. The drawback is that the tool's installation package grows a lot in size.

The CELAR IS Service is stateless by design. The CELAR IS Service does not maintain any state between user requests. This enables the server to be included in a "Server Farm" where a Load Balancer is responsible to distribute the users' requests. Such operation increases the availability of the CELAR IS and makes it less prone to 'hardware' errors. In the rare case of virtual machine failures, the load balancer will distribute the traffic to another server. Depending on the incident, the CELAR IS may recover almost seamlessly or require from the Application User to re-issue the task that failed. For example, if the host fails the CELAR IS needs to recalculate the whole procedure from the beginning the Application User will be informed that "Something Went Wrong" and she will have to take action (e.g. request again the calculations). If there is an error in an internal module, the CELAR IS will catch the error and auto recover (e.g. repeat some of the calculations). The latter operation will be transparent for the Application User, beside some minimal increase in the response time.

4.1 Analytics Control and Analysis Module Improvements

During the first year, we have implemented a prototype analytics engine (Analysis Module) in order to provide cloud usage analytics to the Application User. The Analytics Control module occasionally invokes this engine, in order to fulfill the Application User requests in analytics. The analytics Module provides basic mathematical and statistical functions based on Apache Commons Math library [Commons Math]. In this second version of the Analysis Module, we improved, in terms of performance, the trend calculation tasks and we added an implementation of a down sampling function.

The trend calculation as it was presented in [D4.2] was based on the implementation of a Simple Moving Average (SMA) function. We improved that implementation providing an algorithm that can run in parallel, using threads. Moreover, to lift some burden from the Application Users, we automated some processes and the Analytics Control can calculate the input of the algorithm automatically. The SMA windows parameter, which defines the type (long / short term)

of the calculated trend, had to be either inserted, as a parameter from the user, or the system would use a fallback parameter defined in the configuration files. We changed that and now there is the option to estimate automatically that window parameter. The logic behind this is based on the deployment duration and the amount of the metric values for which the trend needs to be calculated. The purpose of the SMA is to smooth the irregularities of the input data and thus reveal the trend. As the timeframe of analytics (e.g. the whole deployment duration or the last three hours of the deployment) we are producing longer-term trend, since we assume that the Application User does not want to see, at least in a first stage, a too detailed trend. However, even when we increase the level of detail, this detail will get lost due to the big amount of data points(metric values). In the same essence, we are also increasing the SMA window, as the amount of data grows larger.

A new feature named **sampling** is new functionality that we added to the second version of the CELAR IS. This functionality aims to improve the visualization rendering by reducing the amount of data points the Visualization tool has to render, but without losing (almost) any visual information. Jugel in [Jugel] identified that there is no need to send a million data points over the internet in order to render a chart, which cannot visually represent all this information. Thus, we can speed up the calculations and the charts rendering, by only focusing on the data points that best represent the overall trends in the data. To reduce the actual amount of data, in an almost lossless way, many techniques such as aggregation, binning and sampling can be realized. In the CELAR IS, we choose to use an implementation of *Largest-Triangle-Three-Buckets* down-sampling algorithm presented in [Steinarsson]. This algorithm separates the raw data in almost equal size buckets and selects from each one the most representative point, based on the previously selected sample. We choose to use sampling instead of regression techniques, for data reduction since it uses observations from the actual data and does not create new points as representatives like the second one does. By using points from the initial data, we do not lose some of the additional information such as the time series trend. Having this in mind, we can apply the down-sampling technique before the calculation of trend, optimizing in this manner even more the performance of our tool during the analysis process. We need to point out that these data reduction techniques are most suitable to increase the performance of visualizations and not the performance of the analysis process itself.

Implementation wise, according to the down-sampling implementation, we are using, the Analysis Module needs to ensure that cases of extreme down-sampling do not exist. Extreme down-sampling occurs when the calculated sample uses less than 5% of the original points. We avoid this programmatically by not letting the analysis module to apply sampling if the amount of data does not fit our requirements.

4.2 Data Collection Module

The Data Collection module, as it was presented in D4.2, is the module of CELAR IS that collects data from any underlying configured data source. This data is further used by IS in its calculations. As we mentioned before, the CELAR IS mostly communicates with CELAR Manager to access the data stored in CELAR DB. Most of the changes we did, in Data Collection Module, are under the hood and they are not visible to the Application User. These changes aim to improve the interaction between the IS and the modules, of CELAR, from where IS requests data.

In the first version of our modules, we were parsing the CELAR Manager response as clear text. During the second year, ATHENA built a helper Java library, namely celar-

beans [CELAR Beans], capable of parsing the CELAR Manager REST API response into Java objects. This is easing the integration process between IS and CELAR Manager and makes the IS less prone to errors, due to changes to REST API response format. Our task was to make all the appropriate changes to our code to encapsulate that library. Moreover, we have upgraded our initial code in terms of error handling.

4.3 Information System Visualization Tool

In the implementation of Visualization Tool, included in the first version of CELAR IS, the user was able to search for application versions and deployments on the Search Page, to view summarized information on the Dashboard Page and to view auto generated analytics reports for a deployment in the Deployment Page. In the second version of CELAR IS, (i) we have updated the Search Page, including all the functionality presented in [D4.2]; (ii) we have built a new view under the Deployment Page, where we are displaying analytics mapped graphically onto the application topology and (iii) we have implemented the Comparison Page as we introduced it in [D4.2].

Topology Explorer

In the topology view, we leverage the graphical representation of application topology to provide a better view on how one's component load / performance affect the others. In Figure 3, we can see the Playegen's DataPlay [D7.2] application as it can be presented through the topology view. The Application User, using the topology view UI [Figure 4], has selected the metric, which she wants to inspect. The user of our scenario has selected, (i) cost and standard deviation (stadev) metrics for master nodes; (ii) pgBlocksDiskRead for Postgre SQL component; (iii) cacheHits for Redis in-memory storage and (iv) writeLatency for Cassandra cluster. Through the time bar (at the top of the topology view page [Figure 3], the user has also defined a time window in where she wants to apply the analytics procedures. For example, imagine that the aforementioned application deployment has run for over a year. The user wants to focus her analysis only in the summer months. She can do that by specifying that time window in the time

Figure 3: Analytics Mapped on the topology blueprint

bar. In the case of an elastic application, the user will also have access to the elasticity decisions [Figure 4] that have been enforced for every component. By selecting one of them, the system will automatically calculate a time window, adding a predefined period before and after the moment of the selected action. The calculated time window in not binding since the user can adjust it to her needs using again the time bar. After this parameterization step, the user can issue the calculations by clicking the respective control. The visualization tool will display the calculated information in charts near to each component. The user can visually see, through the charts, how a tier or a component affects any other.

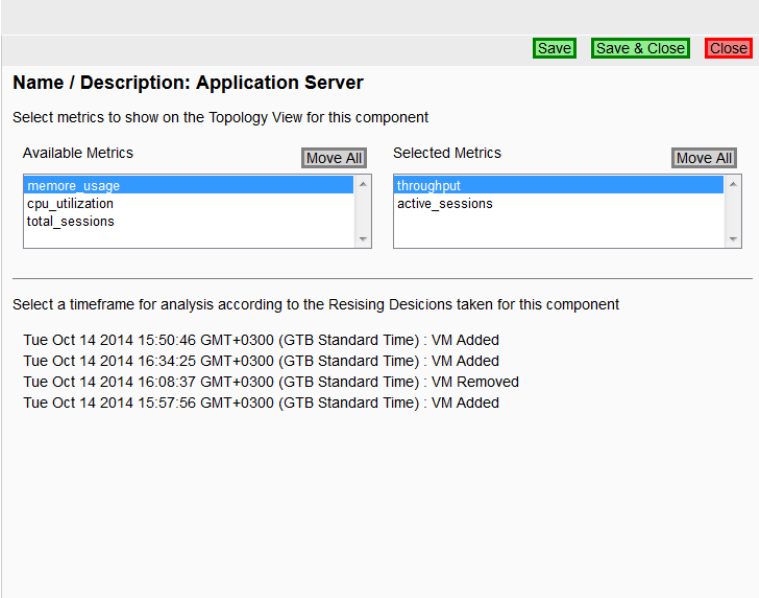


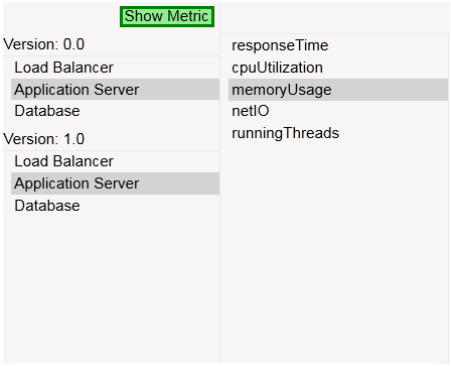
Figure 4: Selecting Metrics for Analytics or navigating to a Resizing Action event

Comparison Page

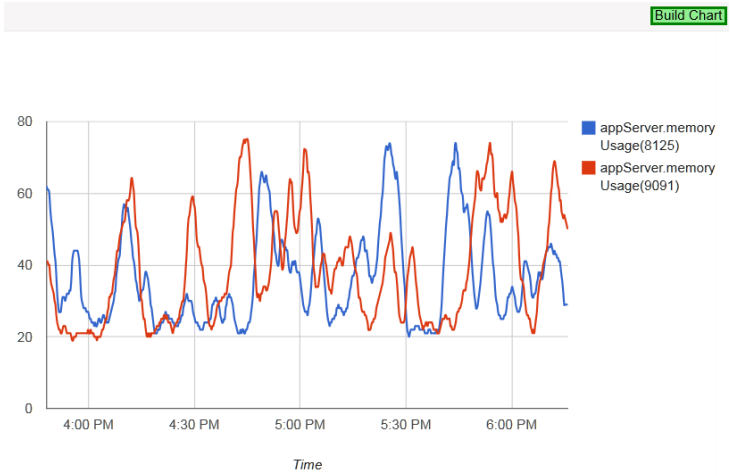
As we defined in [D4.2], the Application User will have the ability to compare several deployments previously run on CELAR. The Application User can compare the monitoring and cost data collected and the calculated analytics at component level. Figure 5depicts the comparison user interface offered by the CELAR IS Visualization tool.

Before issuing the comparison and building the chart, the Application User needs to guide the process, by giving some input to the tool through the User Interface. The Application User needs first to specify which versions of an application she wants to compare. She can do that through the “ADD” control on the top-right corner of the page. After adding two or more versions and clicking the “COMPARE” control, the topology of each deployment is fetched, from the CELAR IS Service, and displayed as a list at the left side of the Comparison Page. When the Application User selects, one or more component to participate in the comparison process, the CELAR IS Service will calculate and display to the user, the common metrics of these components. As a last step, the Application User needs to select the metrics she wants to visualize and initiate the chart rendering by clicking the “Build Chart” control.

(a) User selects the Application versions for comparison



(b) User Targets components and specifies metric



(c) CELAR IS Visualize the comparison according to the User's parameters

Figure 5: Comparisons Page

To briefly sum up, a common use case scenario can follow the following sequence of actions

1. Select application version for comparison (Figure 5a)
2. Specify the component of each version, that will take place in comparison (Figure 5b)
3. Select the metrics you want to visualize (Figure 5b)
4. Issue the chart rendering (Figure 5c)

5 Documentation

5.1 Information System Artefacts and Installation Process

The CELAR Information System, as stated previously, consists of two different components, each of them is packaged, distributed and can be installed differently. Both of them are written in Java, but their distribution packages include also some Bash\Shell scripts that guide the installation and execution processes. While it is possible to install each of the components to a different host, for the purposes of CELAR, both of them are installed at the CELAR Server. The CELAR Information System components are distributed both as (i) TAR archives and (ii) RPM packages. The RPM packaging is the default packaging used for CELAR and uses the CentOS Linux distribution [CentOS] as its operating system. The RPM distributables are available through the CELAR Repository. The following subsections give specific details on how one can install and configure these components.

5.1.1 Information System Service

Both TAR and RPM distribution methods install the Information System Service, using the included Bash\Shell scripts and register it to run as a system service. In addition, the user can directly instruct the Information System Service to run directly its Java runnable archive (JAR). Listing 1 bellow shows the command that a user can use, for such purposes.

```
java-jar celar-is-core-${VERSION}.jar ${PATH_TO_CONFIG_FILE}
```

Listing 1: Executing CELAR IS Service via JAR Packaging

The packaging (depicted in Figure 6) of the CELAR Information System Service contains the Information System Service as a runnable JAR, an installer, service scripts and the CELAR Information System Service default configuration files (under the config folder).

```
CELAR Information System Service/
|-- celarIS-Server
|-- celarISServerDir
|   |-- celarIS-Server-start.sh
|   |-- celarIS-Server-stop.sh
|   |-- cloud-is-core.jar
|   `-- resources
|       `-- config
|           |-- celar
|           |   |-- endpoint.celarmanager.properties
|           |   |-- endpoint.jcatascopia.properties
|           |   `-- services.properties
|           |-- server.properties
|           `-- standalone
|               `-- endpoint.jcatascopia.properties
|-- installer.sh
`-- LICENSE.txt
```

Figure 6: CELAR Information System Service Packaging Content

During the deployment process, the downloaded tarball needs to be extracted first. The installer script copies the previously extracted artifact to the appropriate folders and sets the required permissions. Moreover, the Information System Service is registered as a Linux service (daemon) using native OS libraries (/etc/init.d) which allows it to run as a background system process and is automatically started/terminated when the virtual instance, in turn, is started/terminated.

In any case, the user can change the default values in the configuration files to customize the Information System Service as per her requirements. The Table 2 bellow lists the available configuration properties. Excluding the *.port properties, any other properties can be changed at the runtime.

Property Name	Default Value	Type	Description
General Properties			
common.mode	multi	String	The property indicates whether the IS server will run in 'single' or 'multi' mode 'single': 1 user, 1 application, 1 deployment. 'multi': Multiple users, applications and deployments. When operating in 'multi' mode an extra data source endpoint is needed to provide this information. For the purposes of CELAR the IS can only operate in 'multi' mode
common.collector	celar	String	Indicates the 'bundle of' connectors that will be used to obtain the needed data
dev.debug	true	Boolean	If this option is 'true' the service with log additional information for debugging purposes
log.location	/	String	The path where the log files will be saved
srv.port	8282	Integer	The port which the service will listen to.
mgm.port	8383	Integer	Management Interface / Socket Properties.
Analytics Module Properties			
sampling.threshhold	0.9	Double	Sampling threshold defines the portion of the initial data that will be used as the sample. Accepted Values [0.0,...,1.0] 0: automatic 1: sampling if off
sampling.presampling	false	Boolean	Indicates whether the sampling will be applied before the statistical operations or after.
trend.sma.window	10	Integer	Sampling Moving Average window defines the smoothing windows for creating the trending line 0: automatic

trend.parallel.threads	4	Integer	The number of parallel that will be used during the trend calculation. 0: automatic

Table 2: CELAR Information System Service, configurable properties

5.1.2 Information System Frontend

The Information System Frontend executable distributed in the Web Application Archive (WAR) format and needs to be deployed on an Apache Tomcat or similar web container. The Information System Frontend distributable contains the aforementioned WAR executable and an installation script. The installation script during the deployment copies the WAR executable into the appropriate web container's folder. The only parameter that needs to be configured in the Information System Frontend is the Information System Service address (`isserver.ip`) in order for those two to communicate. For the purposes of CELAR, the Information System Frontend is installed alongside with the Information System Service, at the CELAR Server. Thus, the default value of the '`isserver.ip`' is '`localhost`'.

5.2 CELAR Information System Source Code

The source code for the CELAR Information System (v2.0) can be found on GitHub under the following URL:

https://github.com/CELAR/cloud-is

6 Evaluation

In this section, we present our evaluation report based on the CELAR Information System requirements. The requirements of the CELAR IS were initially presented in D1.2 [D1.2] (R4.20 – R4.28) and later were updated in the D4.2 [D4.2] (Section 3.1).

The basic CELAR IS functionality (Table 1) derives from the R4.20, R4.21, R4.22 and R4.26 requirements. Therefore, in order for the CELAR IS, to advance the state-of-the-art, meeting these requirements successfully is crucial. The R4.23 and R4.24 are marked as "ON GOING", since further improvements are yet to come.

Req. No.	R4.20, R4.21	Access to Application Versioning and Deployment Data
Status	DONE	
Means of Assessment/Verification		

The CELAR Information System needs to have access on application version and deployment data in order to provide both the search and the analytics functionality. The application and the deployment meta data information as well as the monitoring, cost and enforced resizing action data generated during a deployment are stored in the CELAR DataBase. The live data (e.g. monitoring metrics, composite metrics) reside at the orchestration VM in the JCatascopia and MELA databases. In both cases, the CELAR Information System uses its DataCollection module to access this data.

Figures Figure 7 and Figure 8 are screenshots taken while we were demonstrating the CELAR Information System. Figure 7 depicts the search functionality of the CELAR IS and verifies that it can access the application versioning data. Figure 8 depicts the topology explorer view, where the Application User can map the calculated analytics on the application topology.

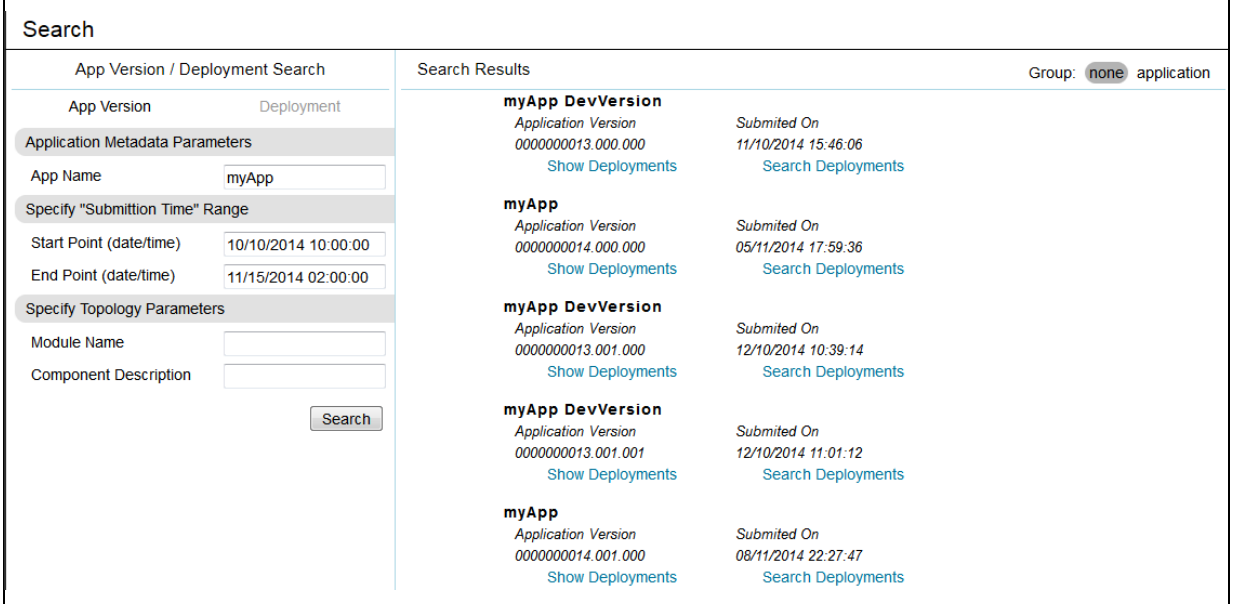


Figure 7: Search Page

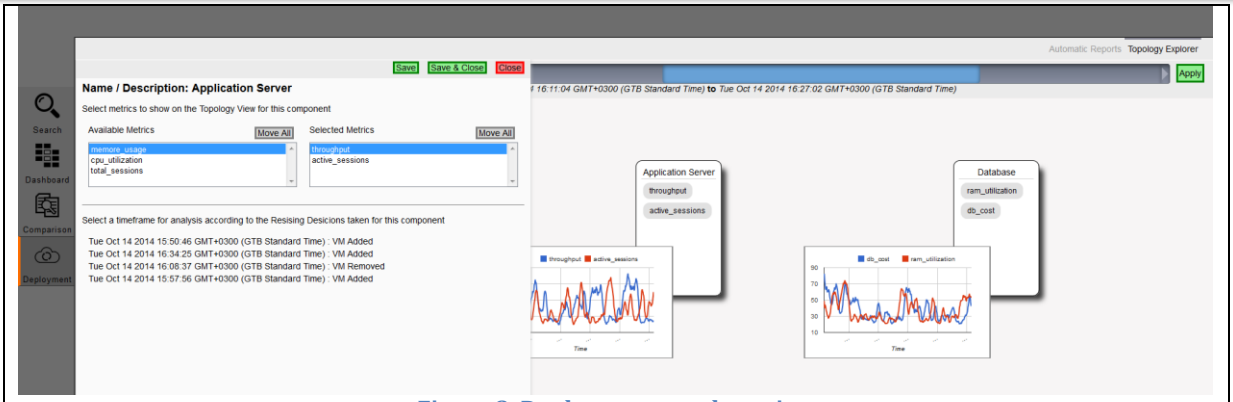


Figure 8: Deployment topology view

Req. No.	R4.22	Cloud Resource and Usage Analytics
Status	DONE	

Means of Assessment/Verification

One of the key features of the CELAR Information System is to use the collected data for a deployment and provided Cloud resource and usage analytics.

The analytics are calculated from the Analytics Controller and the Analysis Module. Figure 9 below depicts some exemplar analytics (metric values trend), which are calculated for display in the auto-generated analytics reports for a deployment.

Figure 9: Deployment Page, auto-generated analytics reports

Req. No.	R4.23	Express Complex Condition Based Queries
Status	ON GOING	

Means of Assessment/Verification

As an Information System, the CELAR IS needs to provide the user with the means to search and find the desirable information stored in the CELAR DataBase.

To hide the complex procedures of two-step queries and data filtering or grouping the CELAR IS provides the Search UI for searching in the Application Version and the Chart Builder UI for access previously collected raw metrics or analytics in a condition based way. The Figure 10 shows the parameters that the Application User can specify while she is searching the CELAR DB for application version and deployment information. Two exemplary queries can be the following:

- What application were submitted during 2:00AM and 9:00PM on the 13th of May
- What deployments started running between 12:00PM and 5:00PM on the 14th of May

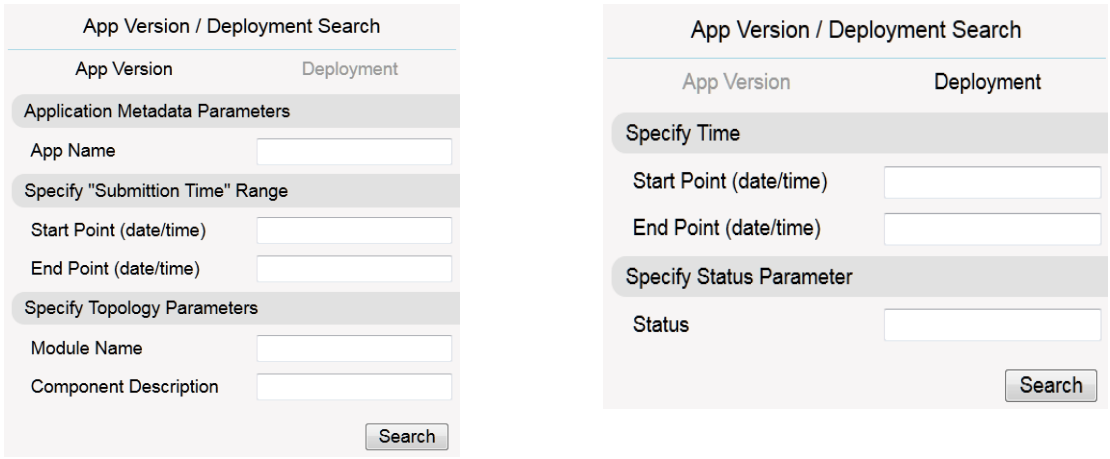


Figure 10: Search for Application Version and Deployments Information stored in CELAR DB

In the Figure 11 we present how the Application User can "ask" queries like *"what was the memory usage percentage of resource X, when CPU usage percentage was over 80%"*. The Application User selects the metrics, of one or more components, which she wants to participate in the query and specifies the condition through the User Interface (e.g. text boxes and drop-down lists). For the presented case, the pie chart shows that from the total time (~2h) that the CPU was over 80%, the memory usage was between 50% and 80% for most of that time. In the current version, the user can express very simple queries, but we are working to extend the functionality of this feature.

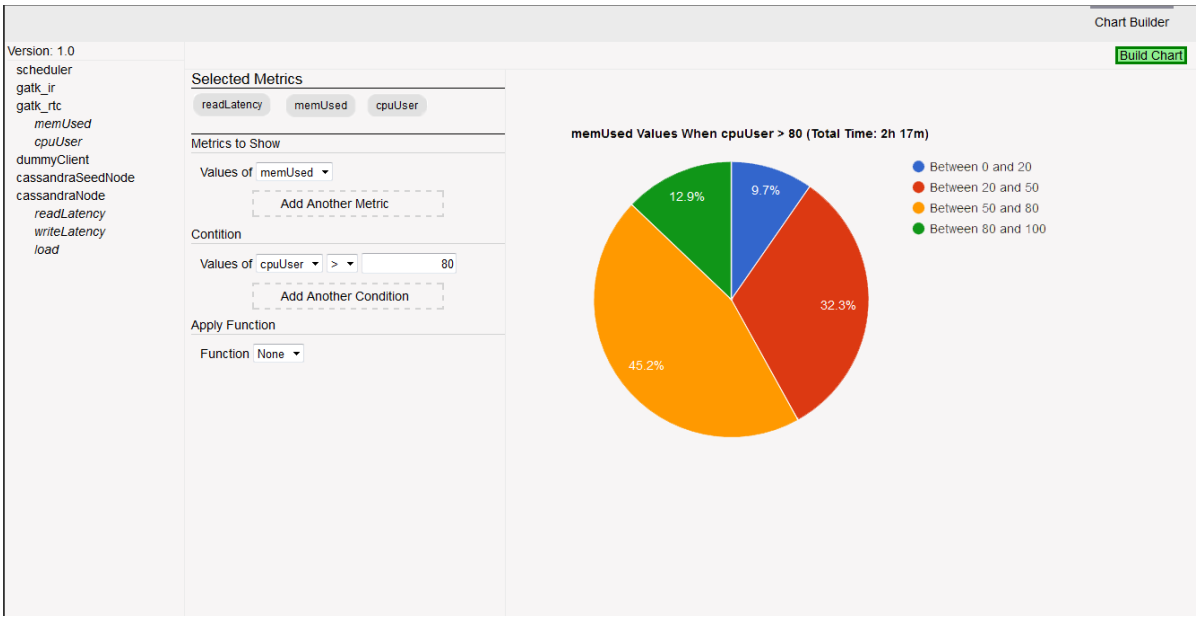


Figure 11: Condition Based Chart Builder

Req. No.	R4.24	Compare Different Deployment Configurations
Status	ON GOING	
Means of Assessment/Verification		

As described in [D4.2] the developer goes through a circle of development, evaluation and optimization of her cloud application. When using the CELAR Platform, the data generated from these iterations, is kept and stored in the CELAR DB. We assume that the Application User deploys each version at least once. The data collected (e.g. monitoring) during the deployment is also stored in CELAR DB. To facilitate the user's need for optimization and help her choose the application version that better fits her expectations; the CELAR IS provides a view for the Application User to compare the data collected during two different application version deployments.

The Comparison User Interface, shown in Figure 12 and described previously in Section 4.3 verifies that we successfully meet our requirement.

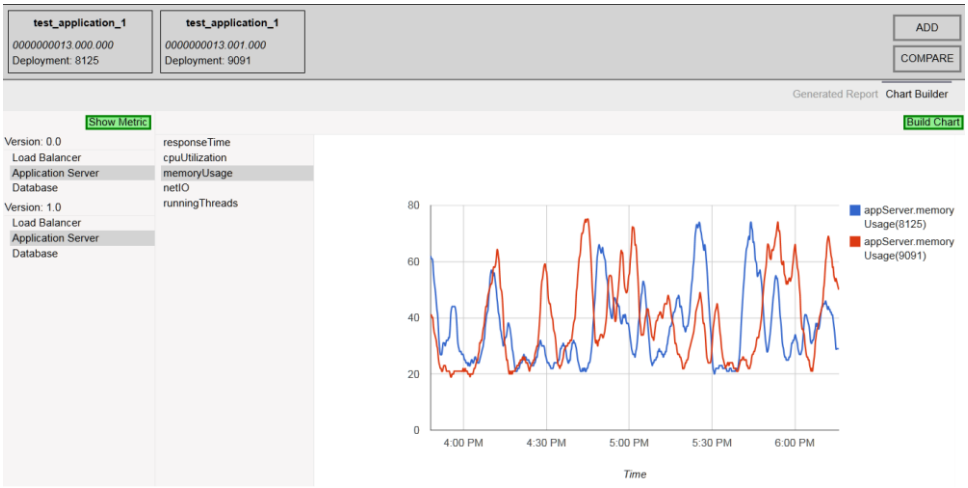


Figure 12: Compare Different Deployment Configurations

Req. No.	R4.25	Be Agnostic to Underlying Data Sources and Cloud Infrastructure
Status	DONE	
Means of Assessment/Verification		

The DataCollection Module as it was described in D4.2, introduces a set of interfaces and abstractions that help *the CELAR IS to be unbounded to the data sources that it connects.* Implementation wise, every module of the CELAR IS knows and uses the Interface specification that the DataCollection Module exposes. For example, if we change the monitoring system the CELAR uses, the only additional step we have to do for the CELAR IS to work is to configure it to use the monitoring data source implementation for the new monitoring system. To further test that the CELAR IS, can work with different data sources without changes in its code, we have implemented and included in our repository two additional implementations of the data collection interfaces that the CELAR IS can work with.

- The 'standalone' variation is configured to use JCatascopia Monitoring System, as the data source for both live and historic monitoring information and to read the application topology and other deployment specific data (e.g. cost, elasticity actions enforced) from files
- The 'dummy' variation is configured to generate dummy data for all the available interfaces and is primarily used for testing reasons.

Req. No.	R4.26	Elasticity Awareness
Status	DONE	
Means of Assessment/Verification		

The related cloud usage analytics tools, as presented in D4.2, do not consider elastic applications. In order to advance the state of the art the CELAR IS, needs to be aware of the cloud application elastic nature.

The CELAR IS uses this information, already stored in the CELAR DB, during its analytics calculations and exposes to the users as annotations to the presented visualizations. Figure 13 shows a line chart captured while visualizing the CPU Usage for a cloud application deployment. The vertical lines annotate the occurrence of an elasticity action happened, giving also additional information such as the type of the resizing action (e.g. VM Added). Furthermore, in the topology view the Application has the ability to use the Decision Taken as time bookmarks of the application deployment duration and by selecting one of them to navigate to the specific time and evaluate the resizing action according to the collected metrics.

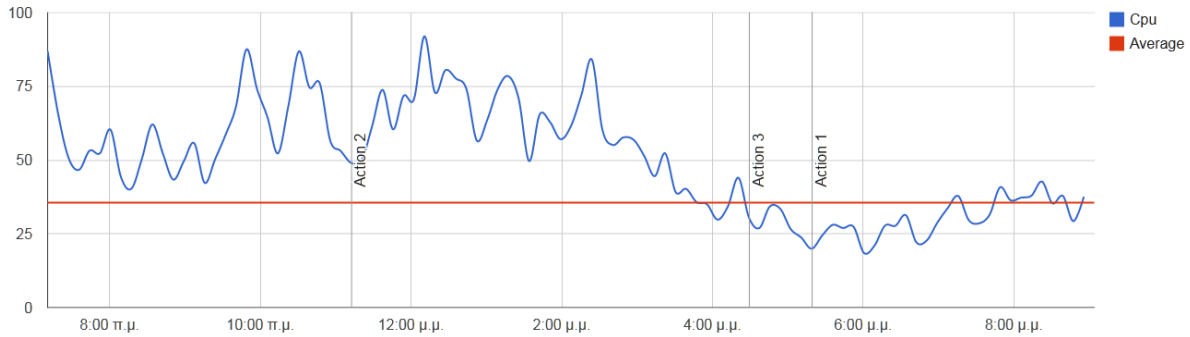


Figure 13: Elasticity Decisions annotated on Trend Line Chart

Req. No.	R4.27	Information Consistency
Status	DONE	
Means of Assessment/Verification		

We designed the CELAR IS from the beginning to connect and fetch data from CELAR Modules and not to maintain its own database. The CELAR DataBase as the central database of the CELAR Platform maintains the majority of the data.

Req. No.	R4.28	Low Latency
Status	DONE	
Means of Assessment/Verification		

In order to evaluate our tool, in terms of response time, we are defining response time as the time that our tool needs to serve a single user request. Based on the formula of [Savoia] we are calculating response time as the sum of the time the back-end services needs to process the data (e.g. calculate the trend of raw metrics), plus the time that the front-end need to load, interpret and visualize the data (e.g. build the charts).

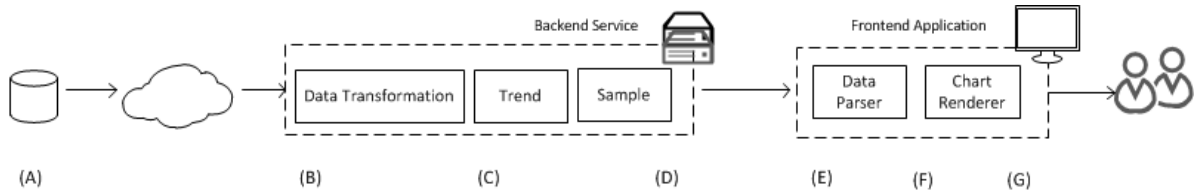


Figure 14: Sequence of distinctive tasks during an analytics process that have impact to the total

In Figure 14, we abstractly depict the sequence of distinctive tasks that run to fulfil an analytics request from the user. We are marking the start of each task with a letter A, B, C e.t.c. to further help the reference to a task's duration. Below we further explain the calculations that take place in the depicted tasks.

- **Fetch From Data Source (A – B)**
  Contact with a Data Source (e.g. Monitoring Server) and get the needed data. This can be further separated to the time that the data source needs to prepare the data and the transmission time over the network.
- **Parse to Internal Structures (B - C)**
  To ensure several abstractions, the Data Collection Module needs to parse the Data to our tool internal structures. Moreover, this task contains the time that the CELAR IS Service needs to prepare the HTTP response, in order to send data to the frontend Application
- **Analytics Calculation (C - D)**
  The time that our Analysis Module needs to calculate the trend and apply data reduction using sampling
- **Transform to Google Charts Format (E - F)**
  The frontend application, receives the data, to visualize, in JSON format. It needs to parse this data to a format that Google Charts understands in order to visualize it.
- **Render Google Chart (F - G)**
  The time that Google Charts scripts, need to read the data and 'draw' the visualization

According to the study in [Nah], end user starts noticing the delay after 4 seconds and delay of 12 seconds or more causes satisfaction to decrease. These times are valid when the user knows that she is expecting content to be loaded.

To evaluate the CELAR Information System in terms of response time, we consider the scenarios, (i) the common case scenario and the (ii) worst case scenario. For the former, we realize the 3-tier application described in [Trihinas, 2014b]. The raw metrics are stored in JCatascopia database and the user requests from the CELAR IS to present analytics for 1-day duration (approximately 30.000 raw [3]metric values). For the later, we are using a random data generator in order to generate up to 1million (~ 3 months of data) metric values and stress the CELAR IS.

In order to get more consistent values, we are repeating each of our runs twenty times and we are choosing the median as the value we will use in our evaluation process. The time that the sampling process needs to finish differs according to the type of data (e.g. how irregular the data is), so in every iteration we are using a different

[3] With the term 'raw', we mean that we use the metrics as stored in the monitoring database (e.g. one value; every 15seconds) and they are not aggregated to higher granularities.

random data set.

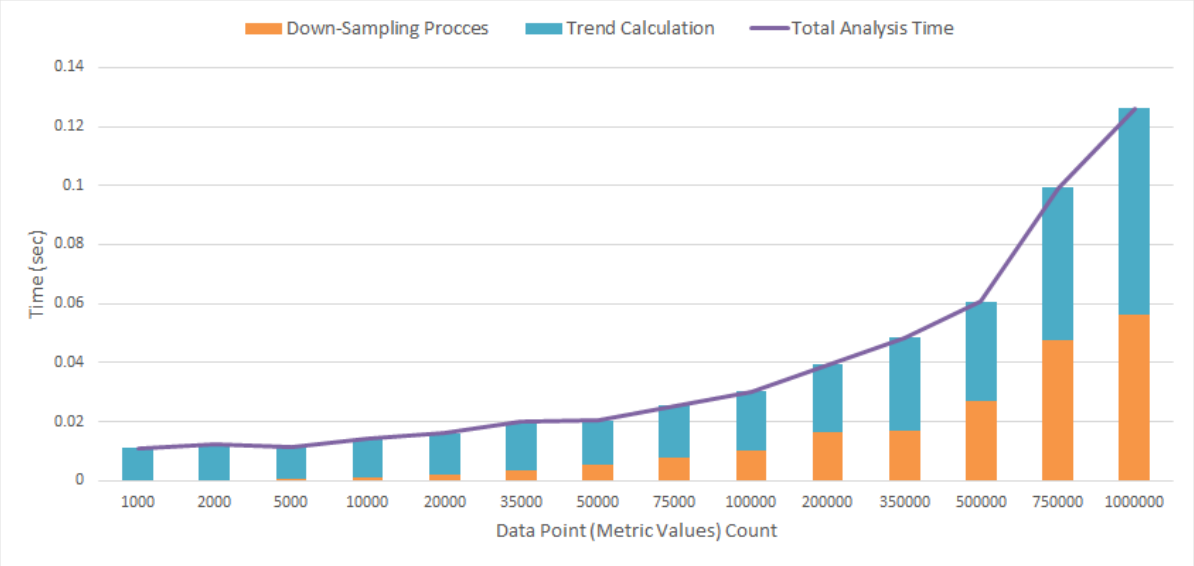


Figure 15: Duration of the Calculations (trend and sampling), running on CELAR IS Service

Evaluation Environment: For the purposes of our evaluation, we installed our analytics tool to a vanilla Virtual Machine running Centos 6.5. The system uses 2 virtual cores and 4 GB of RAM. We have set JVM (Java Virtual Machine) heap size, for the CELAR IS service, at 857 MB, which were proved sufficient to handle our calculations.

In order to present a more detailed view, we are presenting the time the CELAR IS Service needs to calculate the data and time the frontend application need to present the visualizations separately. We are concluding later, with the total round trip time. Figure 15 presents the time that the CELAR IS Service needs to calculate the trend of a metric. Each bar consists of the times that each distinctive task (trend calculation and sampling) needs to complete. The trend line summarizes how the response time is affected by the data points amount. As we described in [D4.2], our tool implements a

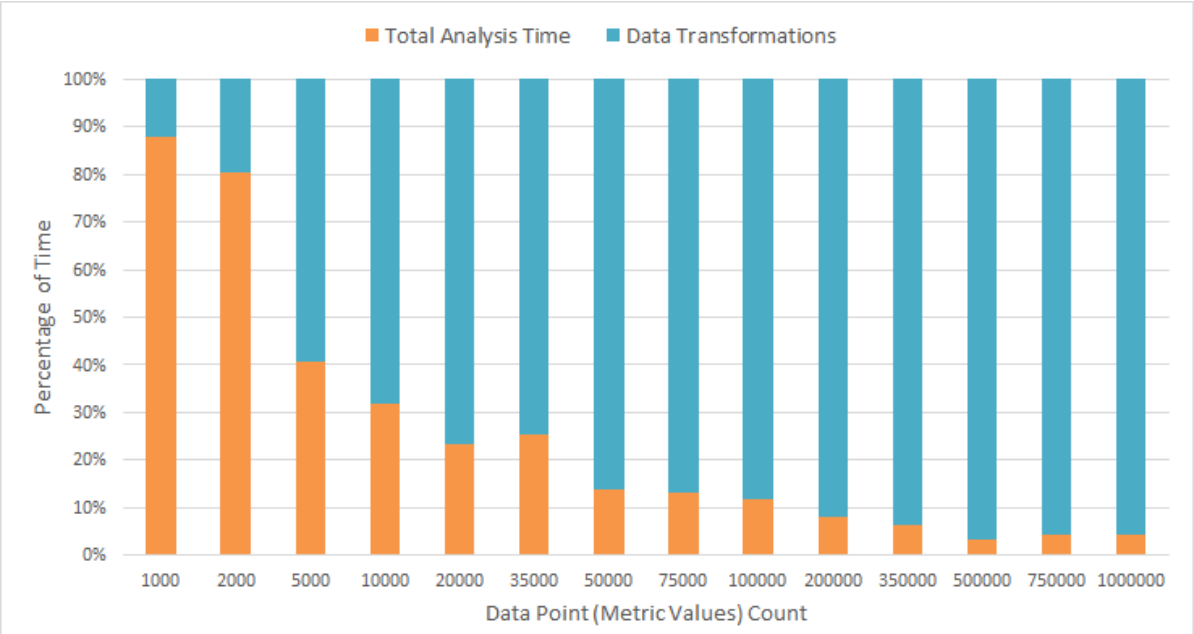


Figure 16: Duration in percentage of the total analysis process running on CELAR IS Service

set of abstractions, in order to be agnostic to the underlying service, from where it consumes data. These abstractions have the side effect that additional data transformations are needed in several cases. Figure 16 presents the percentage of time that each task of the CELAR IS Service needs to finish. It is clear that the percentage of time need for calculating the trend and sample is getting smaller, as the amount of data grows.

Unlike the experiments we run for the CELAR IS Service, in the case of the frontend application, we used a limited amount for data points. The largest data set we used contains 9000 data points. We identified that this is the 'limit' of the Google charts, exceeding this threshold our frontend application starts becoming unresponsive. However, as we described in Section 4.1 the amount of data that can be presented in a line chart through a computer screen is limited and so we consider this threshold tolerable. If we add too much data in a chart, we are concealing the useful information. In Figure 17, we present the time that CELAR IS Frontend needs to render the charts in a function of the metric values count.

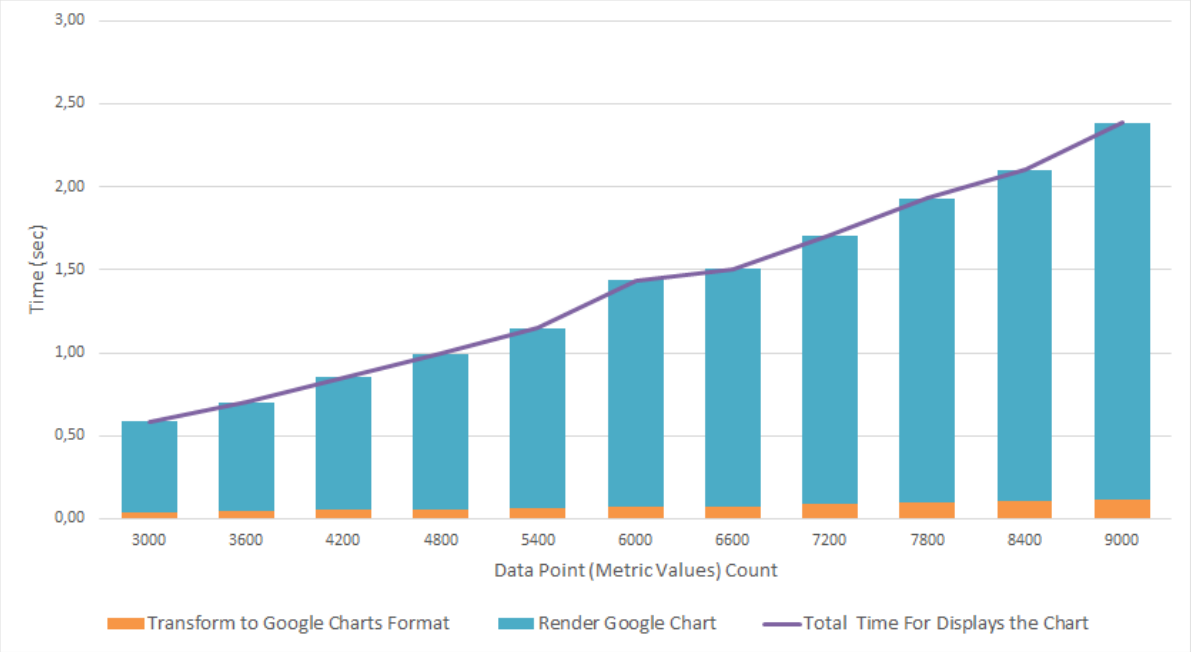


Figure 17: Duration of the Chart Rendering task, in Frontend application

In figures Figure 15 and Figure 17, we have presented the measurements of the distinctive tasks of our analytics tool for both the computational (CELAR IS Service) and the presentation (frontend) level. The response time, from the moment that the end user will request the analytics until she sees the final visualization differs a little. This is because, the module that the CELAR IS requests data from, needs to first calculate the data (e.g. query its database) and then 'push' it to the Data Collection Modules though the network. This time is also added to the total response time and corresponds to the 20% - 30% of the total CELAR IS response time as we show later in Figure 20.

Figure 19 shows the common case scenario, which was described earlier. Despite saying, that we will not allow the browser to be unresponsive, we have added in the Figure 19 the last two values that exceed the threshold of 9000 data points, to pinpoint that event.

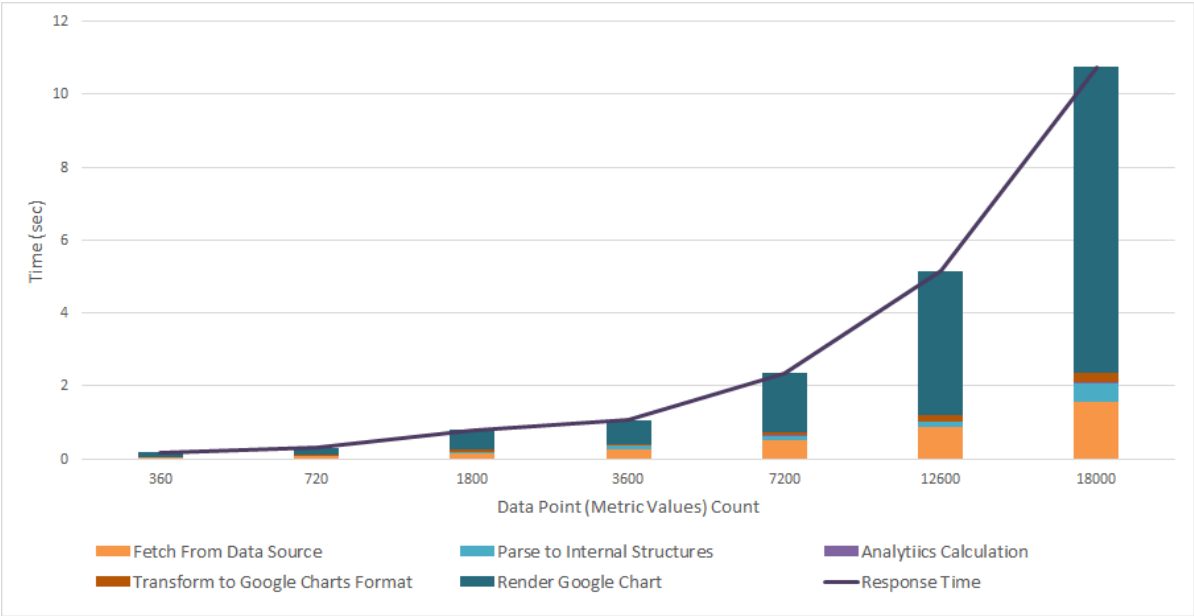


Figure 19: Common Case Scenario - Duration of each distinctive task and the total response time, of an analytics process

In the Figure 18, we present the response time for the worst-case scenario we may encounter. Here the user wants to analyze up to 500000 data points, but as we showed above in Figure 17, the Google charts have a limit of 9000 data points. In order to visualize successfully this amount of data we are realizing the methods presented in Section 4.1, to 'shrink' the data to 9000 data points. We also point, that given the literature, presented in Section 4.1, 9000 points are still too much to be rendered in a line chart on a screen, and will make the chart unreadable.

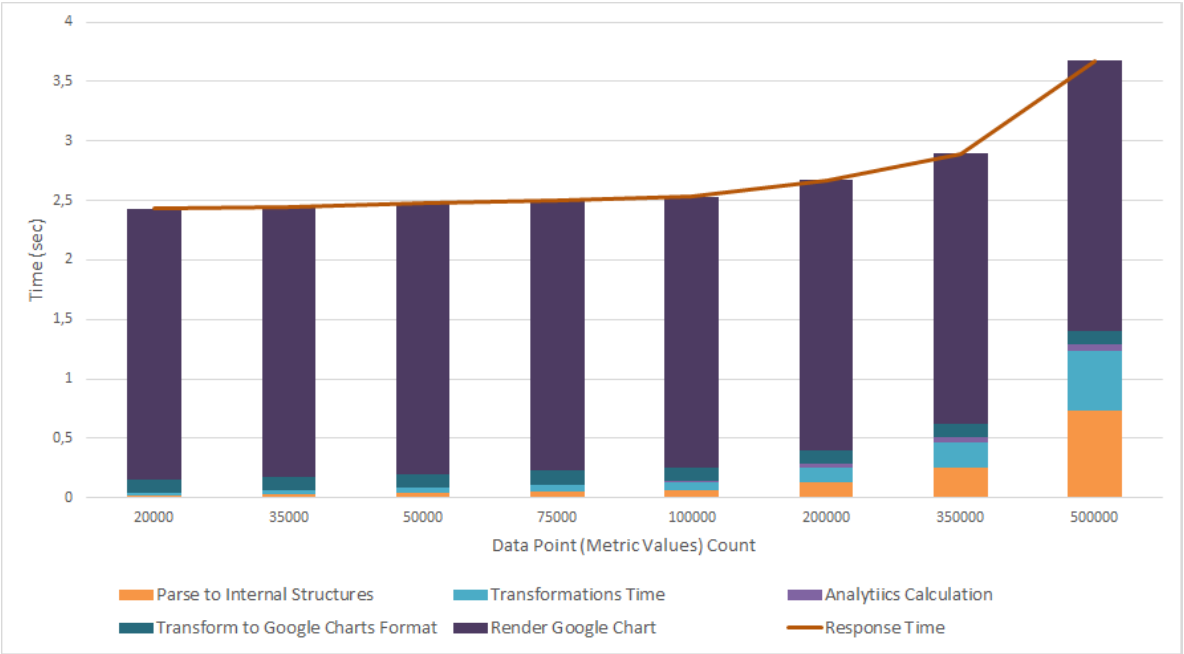


Figure 18: Worst Case Scenario - Duration of each distinctive task and the total response time, of an analytics process

Excluding the two last values in Figure 19, we can say that in both cases we successfully meet our requirements and achieve time less than our 4 seconds threshold. Moreover, we ensure that the frontend application will not feel unresponsive due to the large amount of data it has to visualize. Furthermore, we have observed that if we increase the number of cores at the host system, from2 to 4, we can achieve about 25% reduction in the time, the CELAR IS Service needs to complete each calculation. This is expected since most of our calculation run in parallel using threads. So by increasing the number of cores, we increase the number of threads that can run simultaneously. We should also point that most of the time is consumed to fetch the data from the configured data source and to visualize the data using the Google charts. We present these findings in Figure 20where we show the time that each task needs to finish as the percentage of the total response time.

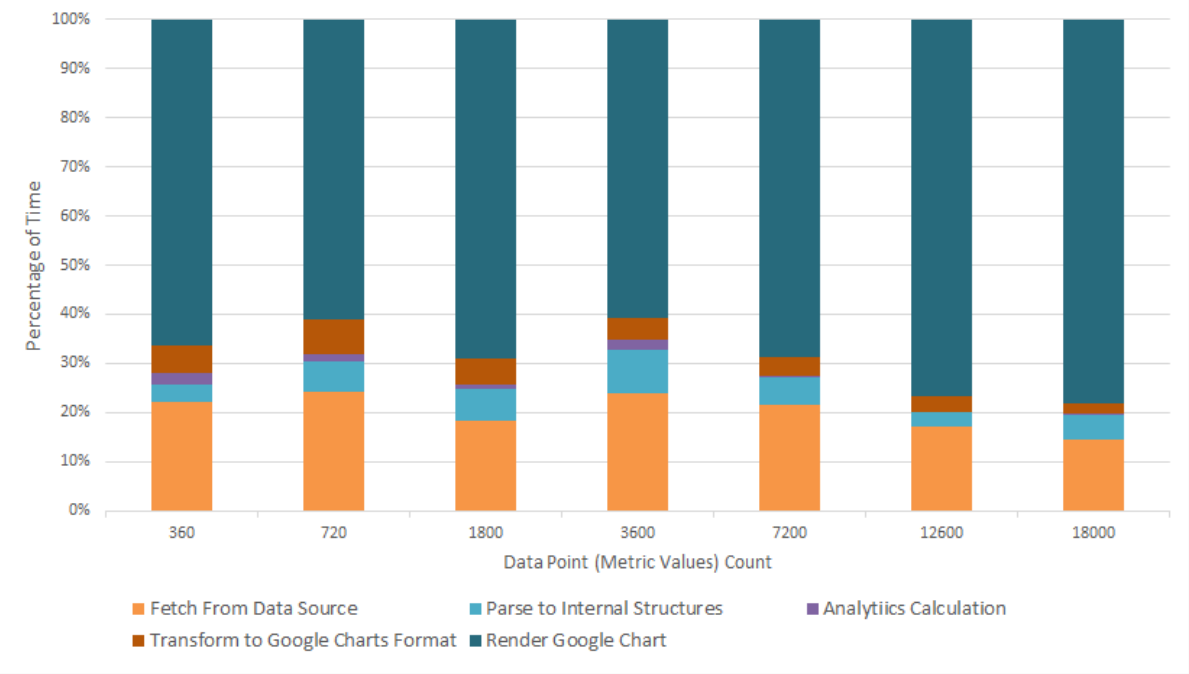


Figure 20: Common Case Scenario - Percentage of time of each distinctive task running during an analytics process

The results presented above, regard the latency / response time of a single user request(s). The calculations of the CELAR IS are mostly CPU intensive. However, an analogous to input data size (metric values count) amount of Memory is needed. In a case of a large-scale environment, the CELAR IS can scale out in order to fulfill the users' demand. As we previously mentioned (Section 4), the CELAR IS Service, is stateless and capable of operating 'behind' a load balancer. Thus on cases that the demand is increasing and the performance decades, an additional instance running the CELAR IS Service can be added.

Table 3: CELAR Information System Requirement Evaluation

7 Conclusions and Future Work

In this Deliverable, we have presented the final version (v2.0) of the CELAR Information System, which is part of the Cloud Information and Performance Monitor layer. We have provided a detailed description of the new features introduced during the third year of the CELAR project and the updates of the existing functionality. The most significant changes regard the analytics engine, where we have updated our implementation in terms of performance and automation. We have also implemented a sampling function in order to utilize data reduction techniques and help the visualization and the calculation of the data. Moreover, we introduced two new features to the Application User, which are the topology explorer and the comparison views. To help the CELAR Information System installation, we have described how the system is deployed, configured and run. Finally, we have presented an evaluation report and we showed how the implemented system meets the requirements defined in D4.2 and D1.2.

As Task 4.3 ends, focus shifts to sustainability of the CELAR Information System in the CELAR eco-system, as well as, a standalone system. The CELAR IS github repository[4] is constantly being updated with information regarding new functionality, bug tracking, issue requests, API updates, changelog and examples. The CELAR documentation website[5] now includes this information as well, with the addition of CELAR integration endpoints and configuration for providers interested in offering analytic insights to their users. Finally, we note that, as the CELAR Information System is deployable also as a standalone system with various integration points, plans to extend its adoption include adding new connectors to cloud providers (Openstack, AWS) and monitoring systems other than JCatascopia and MELA.

[4] https://github.com/CELAR/cloud-is
[5] https://docs.celarcloud.eu/index.html

8 References

[CELAR Beans] https://github.com/CELAR/celar-server/tree/master/celar-beans

[CentOS] http://www.centos.org/

[Commons Math] http://commons.apache.org/proper/commons-math/

[D4.2] D4.2: Cloud Monitoring Tool V1, CELAR project, FP7-317790

[D7.2] D7.2: Cloud Policy Game (Alpha version) and Updated Design, CELAR project, FP7-317790

[Few] S. Few, "Visualizing change: an innovation in time-series analysis," tech. rep., Perceptual Edge, September 2007.

[Jugel] U. Jugel, Z. Jerzak, G. Hackenbroich, G. Hackenbroich, and V. Markl, "M4: A visualization oriented time series data aggregation," Proc. VLDB Endow., vol. 7, pp. 797–808, June 2014.

[Keim] D. Keim, G. Andrienko, J.-D. Fekete, C. G¨org, J. Kohlhammer, and G. Melanc¸ on, "Visual analytics: Definition, process, and challenges," in Information Visualization (A. Kerren, J. Stasko, J.-D. Fekete, and C. North, eds.), vol. 4950 of Lecture Notes in Computer Science, pp. 154–175, Springer Berlin Heidelberg, 2008.

[Kurbalija] V. Kurbalija, M. Radovanovi´c, Z. Geler, and M. Ivanovi´c, "A framework for time-series analysis," in Artificial Intelligence: Methodology, Systems, and Applications (D. Dicheva and D. Dochev, eds.), vol. 6304 of Lecture Notes in Computer Science, pp. 42–51, Springer Berlin Heidelberg, 2010.

[Nah] F. F.-H. Nah, "A study on tolerable waiting time: how long are web users willing to wait?," Behaviour & Information Technology, vol. 23, no. 3, pp. 153–163, 2004.

[Savoia] A. Savoia, "Web page response time," Software Testing and Engineering Magazine, pp. 48–53, 2001.

[Shafer] I. Shafer, K. Ren, V. N. Boddeti, Y. Abe, G. R. Ganger, and C. Faloutsos, "Rainmon: An integrated approach to mining bursty timeseries monitoring data," in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, (New York, NY, USA), pp. 1158–1166, ACM, 2012.

[Steinarsson] S. Steinarsson, "Downsampling Time Series for Visual Representation," Master's thesis, University of Iceland, 2013.

[Trihinas, 2014a] D. Trihinas, G. Pallis, M. D. Dikaiakos, "JCatascopia: Monitoring Elastically Adaptive Applications in the Cloud", in 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, (CCGRID 2014), Chicago, IL, USA, 2014

[Trihinas, 2014b] D. Trihinas, C. Sofokleous, N. Loulloudes, A. Foudoulis, G. Pallis, and M. D. Dikaiakos,“Managing and monitoring elastic cloud applications,” in Web Engineering, pp. 523–527,Springer International Publishing, 2014.

[Xu] X. Xu, S. Huang, Y. Chen, K. Browny, I. Halilovicy, and W. Lu, “Tsaaas: Time series analytics as a service on iot,” in Web Services (ICWS), 2014 IEEE International Conference on, pp. 249–256, June 2014.