LARGE-SCALE ELASTIC ARCHITECTURE FOR DATA AS A SERVICE

Project Number:	FP7-ICT-318809
Project Title:	Large-Scale Elastic Architecture for Data as a Service
Deliverable Number:	D5.7
Title of Deliverable:	Full implementation of the distributed application for multiple micro-clouds
Contractual Date of Delivery:	2015.04.30
Actual Date of Delivery:	2015.05.01

Abstract

The document presents the evaluation of the multi-cloud environment with a usage of datasets, components and queries prepared by adidas, forming the representative application of the LEADS platform.

List of Contributors

Name	Organization	E-mail
Onurcan Anil	adidas	Onurcan.Anil@adidas-group.com
Hueseyin Kocabey	adidas	Hueseyin.Kocabey@adidas-group.com
Jacques Ohannessian	adidas	Jacques.Ohannessian@adidas-Group.com
Pawel Skorupinski	adidas	Pawel.Skorupinski@adidas-Group.com
Wojciech Barczynski	Cloud & Heat	wojciech.barczynski@cloudandheat.com
Marcel Gädig	Cloud & Heat	marcel.gaedigk@cloudandheat.com
Jens Struckmeier	Cloud & Heat	jens.struckmeier@cloudandheat.com
David Garcia Soriano	BM-Y!	davidgs@yahoo-inc.com
Kourtellis Nicolas	BM-Y!	kourtell@yahoo-inc.com
Timothy Potter	BM-Y!	tep@yahoo-inc.com
Hossein Vahabi	BM-Y!	puya@yahoo-inc.com
Emmanuel Bernard	Red Hat	ebernard@redhat.com
Jonathan Halliday	Red Hat	jonathan.halliday@redhat.com
Mark Little	Red Hat	mlittle@redhat.com
Mircea Markus	Red Hat	mmarkus@redhat.com
Pedro Ruivo	Red Hat	pedro@infinispan.org
Aggelos Aggelidakis	TSI	aaggelidakis@softnet.tuc.gr
Eleftherios Chatzilaris	TSI	echatzilaris@softnet.tuc.gr
Antonios Deligiannakis	TSI	adeli@softnet.tuc.gr
Ioannis Demertzis	TSI	idemertzis@softnet.tuc.gr
Minos Garofalakis	TSI	minos@softnet.tuc.gr
Odysseas Papapetrou	TSI	papapetrou@softnet.tuc.gr
Evagelos Vazeos	TSI	vagvaz@softnet.tuc.gr
Christof Fetzer	TUD	christof.fetzer@tu-dresden.de
André Martin	TUD	andre.martin@tu-dresden.de
Do Le Quoc	TUD	do@se.inf.tu-dresden.de
Jons-Tobias Wamhoff	TUD	jons@inf.tu-dresden.de
Pascal Felber	UniNE	Pascal.Felber@unine.ch
Raluca Halalai	UniNE	Raluca.Halalai@unine.ch
Marcelo Pasin	UniNE	Marcelo.Pasin@unine.ch
Etienne Rivière	UniNE	Etienne.Riviere@unine.ch
Valerio Schiavoni	UniNE	Valerio.Schiavoni@unine.ch
Anita Sobe	UniNE	Anita.Sobe@unine.ch
Pierre Sutra	UniNE	Pierre.Sutra@unine.ch

Document Approval

	Name	Email	Date
Approved by WP Leader	Jens Struckmeier	jens.struckmeier@cloudandheat.com	2014-04-29
Approved by GA Member 1	Etienne Riviere	Etienne.Riviere@unine.ch	2014-04-29
Approved by GA Member 2	Minos Garofalakis	minos@softnet.tuc.gr	2014-04-28

Contents

List of Figures

Executive Summary

The document presents efforts that were made throughout the last half a year in order to extend the representative application of adidas, deploy it on the LEADS platform running in the multi-cloud environment and validate the correctness of its execution, and the integration of the platform features exercised by the representative application.

Improving the meaningfulness of the dataset. Section 2 outlines the extensions that have been made to components forming the distributed application in order to enhance accuracy and increase business value. We have extended technological capabilities, information extraction precision, as well as proposed a new set of visualizations.

Multi-cloud setup. Section 3 gives details of the setup of the LEADS cluster in Cloud&Heat that we use to perform experiments. It consists of three primary micro-clouds.

Evaluation of the LEADS platform running the application. Section 4 presents the methods that we have used in order to test the current deployment of the LEADS platform in a micro-cloud setup.

Automation of LEADS cluster deployment. Section 5 gives details on how we use state-of-the-art configuration management system to setup the LEADS cluster in Cloud&Heat micro-clouds.

1. Introduction

Within Work Package 5 of the LEADS project, we implement a representative application using the capabilities and features of the LEADS platform, according to the requirements of the consortium's representative end-user, adidas. This application enables monitoring of mentions of products and assets of interest for the business purposes of adidas. It allows extraction of business intelligence from the massive amounts of public web data without having to set up a crawling, storage and processing solution in house. Yet, the application of mention monitoring requires specific computations to be performed on the data that is crawled, as soon as it is crawled (a near-real-time aspect). A distinctive feature of LEADS compared to other big data processing solutions is that it allows its clients to provide the system with such specific processing units in the form of plugins. This feature is of great importance to implement the distributed application.

Three types of components form the structure of a typical LEADS application, and the representative adidas application follows this pattern as well. First, we offer installation of custom plugins. A client can upload them on platform in order to extract data of her interest from content of pages. The extracted data may include information on a keyword level (e.g. sentiment), page level (e.g. page language), and site level (e.g. site category). Second, we offer query interfaces. A client can query extracted data with her custom set of SQL-like queries. Alternatively, she can upload more specific MapReduce jobs. Finally, the client can define specific visualizations that form the presentation layer for the results of the queries obtained from the system (and in addition to the visualizations available from the LEADS Web Graph Service, described in D5.6).

In this document, we describe which of functionalities of the distributed application we added, extended or updated in the past months. Those functionalities come as parts of the distributed application itself. Then, we present results of tests that we executed on top of the platform running in a multi-cloud environment.

2. Improving the meaningfulness of the dataset – extensions to components that enhance accuracy and increase business value

At month 24 we have presented the first working solution of an information extraction workflow on top of the LEADS platform. We have been extracting valuable content and keyword mentions out of HTML trees of the crawled E-commerce as well as news and blog pages. We have been extracting information about web sites. We treated it as a prototype. Based on the results of the prototype, we have defined additional functionalities that we wanted to integrate. These functionalities bring more business value to the final results. The process of creating the application continued to fuel the design decisions of several of the core components of the platform itself, e.g., for the definition of the plugins interface and support mechanisms among others.

2.1 Improvements in classification and information extraction

One of the business requirements of adidas is to have articles extracted out of blog and news sites. We have crawled these types of sites, however, it is not obvious to distinguish which pages of the sites actually contain an article and which do not. The issue that we have addressed was how to filter out pages that actually contain blog post or article out of sets of pages from blog and news domains. We have applied two simple methods that are good enough for current needs. In a first step, the metadata of web pages is checked for mentions of the date of publication (presence of this kind of metadata is a standard on article pages of many news sites). If the date is not found in the metadata, the URL's structure is checked against the date patterns (that in turn is a standard for article pages of blog sites). If a page contains one of these two, the process of article extraction is executed (as previ-

ously, using the open source–Apache 2.0–Boilerpipe[1]). Afterwards, the second filtering step is executed. The extracted content is verified against rules like: number of words / number of sentences / number of paragraphs. If these values are not large enough, the extracted content will not be stored.
Moreover, we have revised our code for information extraction from E-commerce sites. We have managed to have our code working properly for more web sites.

2.2 Precise identification of keyword mentions and context of mentions

We use the Apache Lucene search engine in order to extract names of brands and products properly out of the content of the crawled web pages. Since these names are often complex, people make mistakes when mentioning them and very often use shorter names or various combinations of words. Therefore, there might be lots of differences between the original brand/product name and the way it is actually mentioned. Lucene comes with a couple of search capabilities that helps to deal with these mismatches. These capabilities include: fuzzy queries, span near queries, or Boolean "OR" queries with minimum number of matching terms set.
We have also put a focus on defining new ways in which context of mentions can be evaluated. The default one –sentiment – gives only an overview. We experiment with aspect-based sentiment analysis where a value is counted for each potential aspect that could have been mentioned together with product or brand name. To give an example, we want to be able to see which mentions are positive because of high comfort or which are negative because of low durability. We present the dimensions proposed in the prototype on Figure 1.

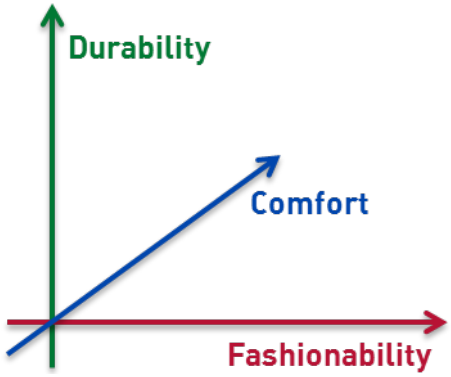

Figure 1: Scale for multidimensional sentiment of mentions

2.3 Integration

In previous periods, although components written in Python were already in use, they were not tightly integrated with the rest of the system but used through an ad-hoc mechanism. In the recent months, we implemented a more sustainable solution. We use the open-source queuing technology ZeroMQ[2] to link the core of the platform written in Java with the Python-based components of the application. Python processes are running as services on each node and awaiting requests from the Java components. Protocol Buffers[3] (also open source) are used to serialize messages.

[1] See: https://code.google.com/p/boilerpipe/
[2] See: http://zeromq.org/
[3] See: https://developers.google.com/protocol-buffers/

2.4 New visualizations to boost business value

In the past months, adidas worked on evaluating the available JavaScript libraries that offer clear visualizations. Some of the chosen visualizations for distribution of product mentions as well as product prices can be found on Figure 2, Figure 3, Figure 4 and Figure 5. It should be noticed that the visualizations themselves are not part of the LEADS distributed platform, but serve to present to business persons the results of the processing performed by the application when running on the platform.

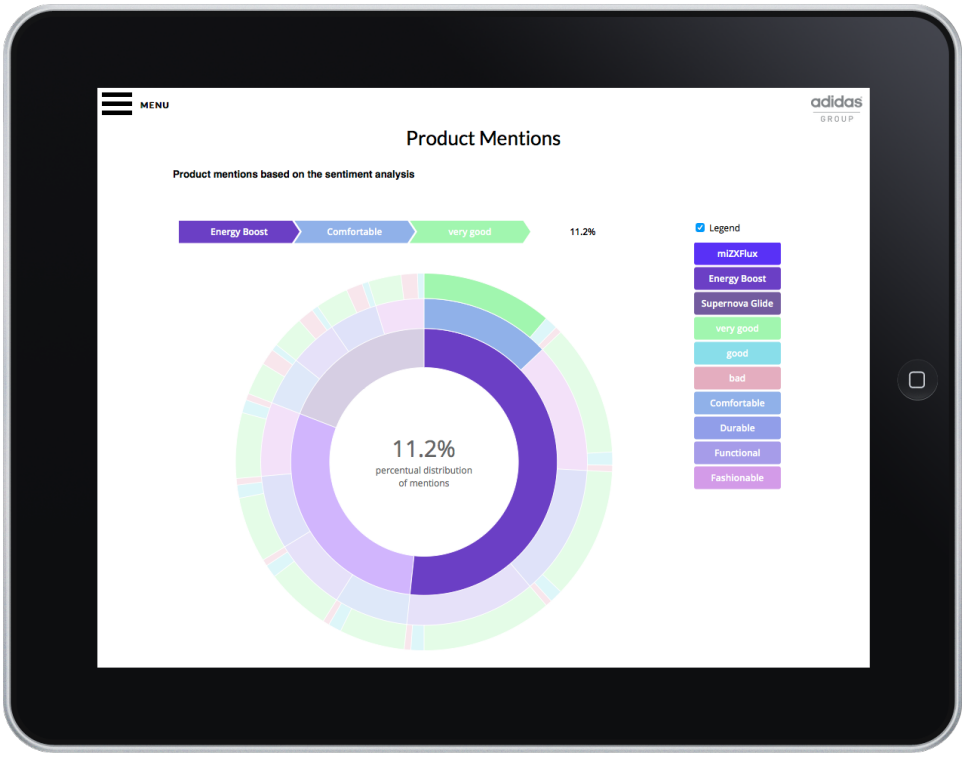

Figure 2: Distribution of product mentions – division by products (Energy Boost), attribute (Comfortable) and sentiment (very good)

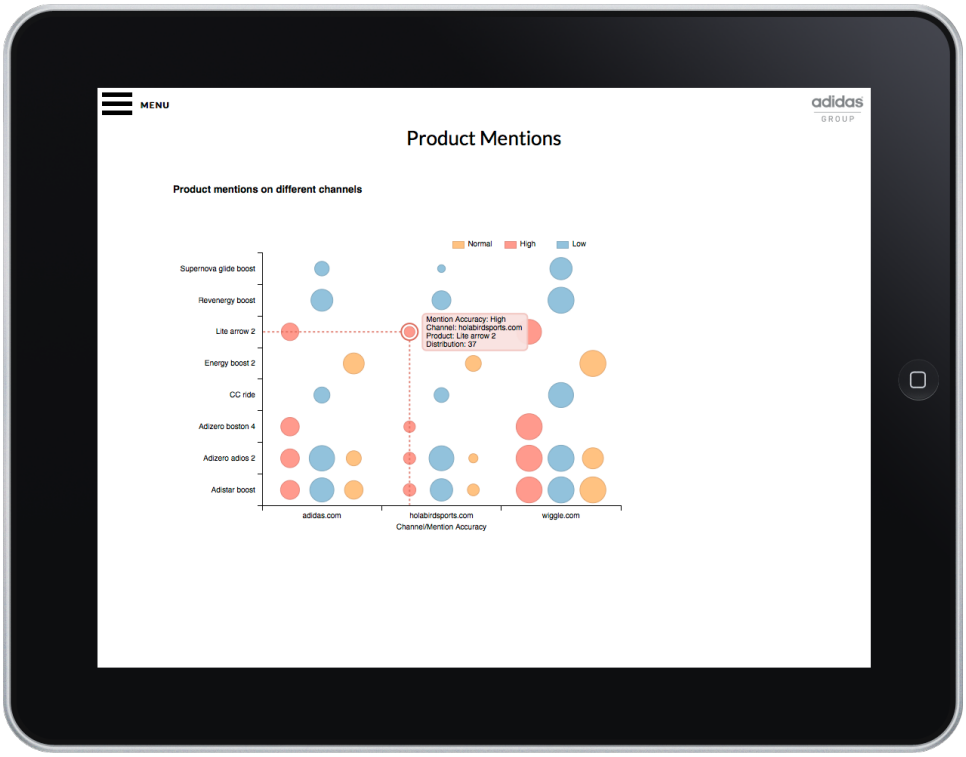

Figure 3: Distribution of product mentions on channels taking accuracy of mentions into account (low/medium/high)

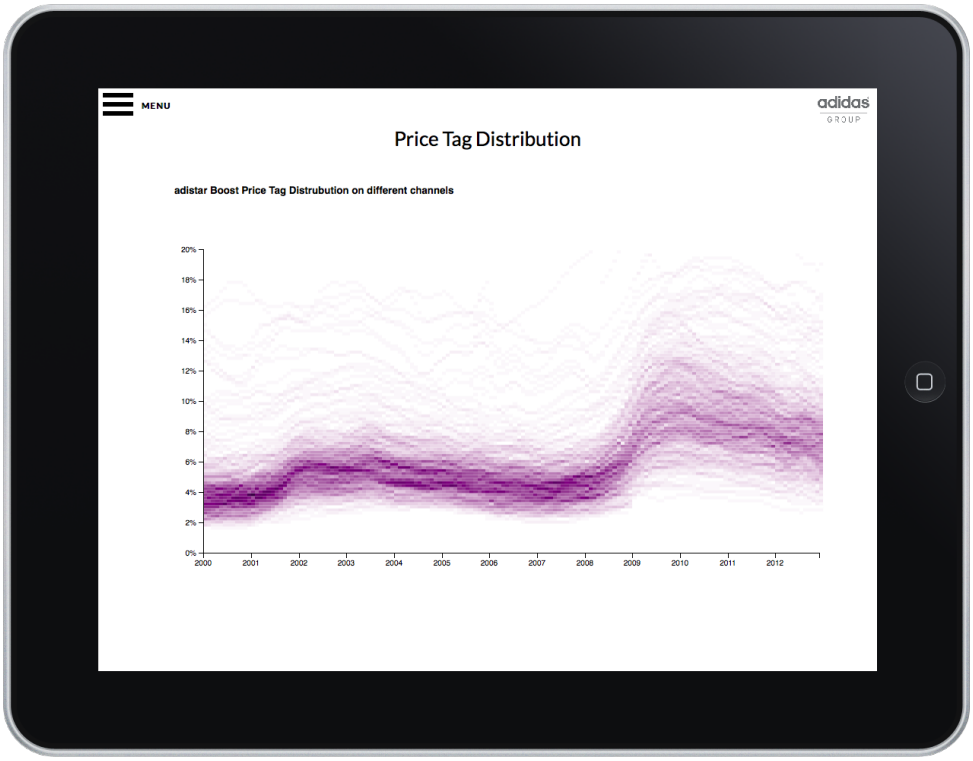

Figure 4: Distribution of prices of category of adidas running shoes called 'adidas Boost' in a set of Ecommerce sites

Figure 5: Distribution of products of each category per date per site (channel) together with an average price

3. Multi-micro-cloud LEADS cluster

Figure 6 presents the cluster overview. We have three primary micro-clouds: hamm5, hamm6, and dresden2. In each deployment, we have an on-demand Hadoop YARN cluster (in current testing phase, we run it permanently). It consists of three nodes. We always install the LEADS WP1 crawler Unicrawler on the primary YARN node. Unicrawler uses YARN to distribute crawling tasks per deployment.

Every micro-cloud runs at least one query engine instance (with Infinispan and Ensemble). The query engine nodes can be dynamically added with our central configuration based on Saltstack. It lets us extend our infrastructure with nodes from other micro-clouds and perform experiments with different configuration.

We run the query engine nodes in dresden2 and hamm6 in order to perform the experiments presented in Section 4. We are going to extend the tests with additional deployments hamm5 and (not shown in Figure 6) muenster2 for the large-scale platform evaluation planned for M36.

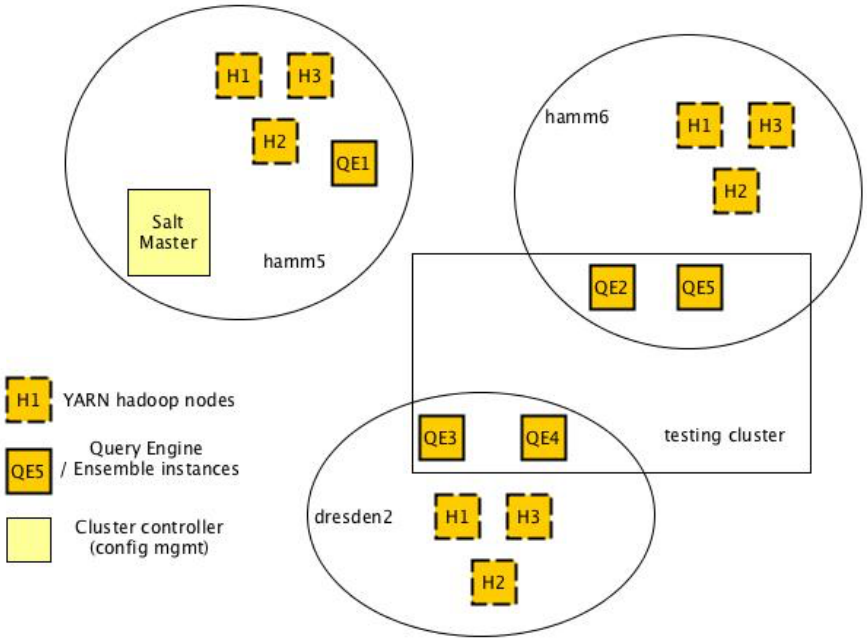

Figure 6: Multi-cloud LEADS setup

An essential requirement for the LEADS cluster setup was to deliver a robust way for collecting metrics. For this purpose, we deployed PCP.io on all nodes[4]. The most important performance indicators for our experiments are: the CPU utilization, the memory consumption, and the network usage. PCP.io stores metric values in a very concise format (so we can collect them for long time) and --- in case of any issues --- we have almost thousand metrics collected additionally per default. The additional metrics help us to investigate deeper any issue or interesting observation. Figure 7 depicts one of PCP frontends – vector[5]. We store all PCP measurements in object store to share them among project partners.

[4] See: https://github.com/skarab7/leads_query- engine/blob/develop/salt/salt_master/srv_salt/top.sls#L22
[5] See: https://github.com/Netflix/vector

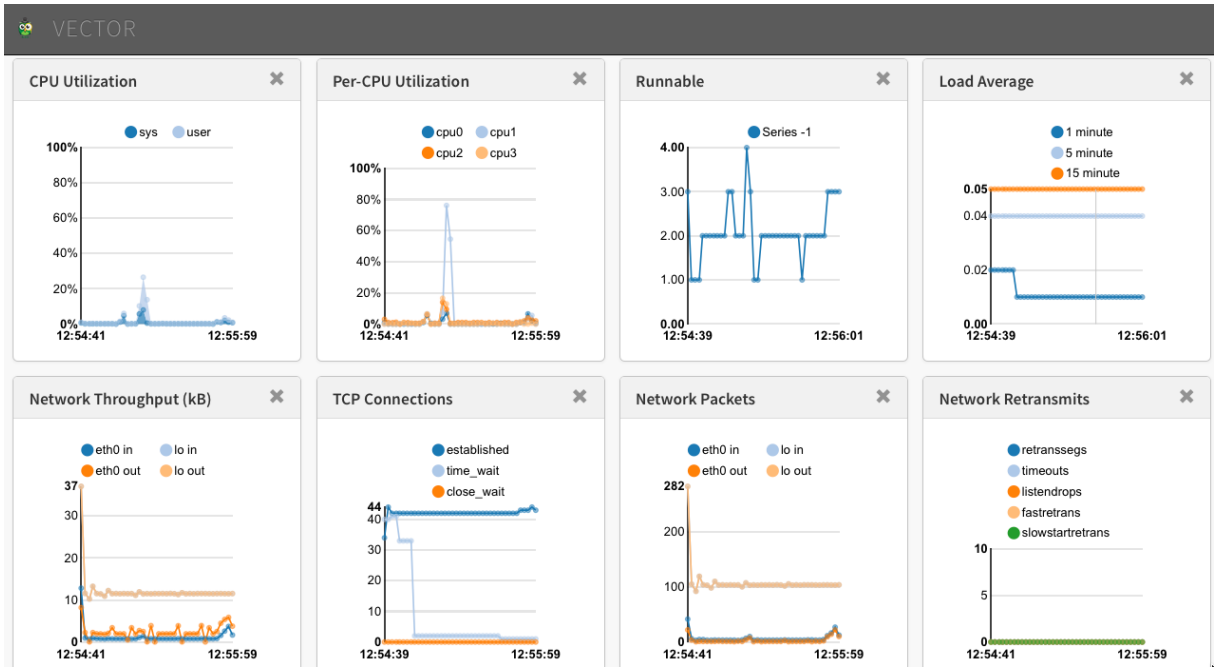

Figure 7: PCP frontend Netflix vector

Additionally, we have access to the infrastructure monitoring on host node (see Figure 8). We plan to correlate the infrastructure monitoring data and virtual environment in the next months.

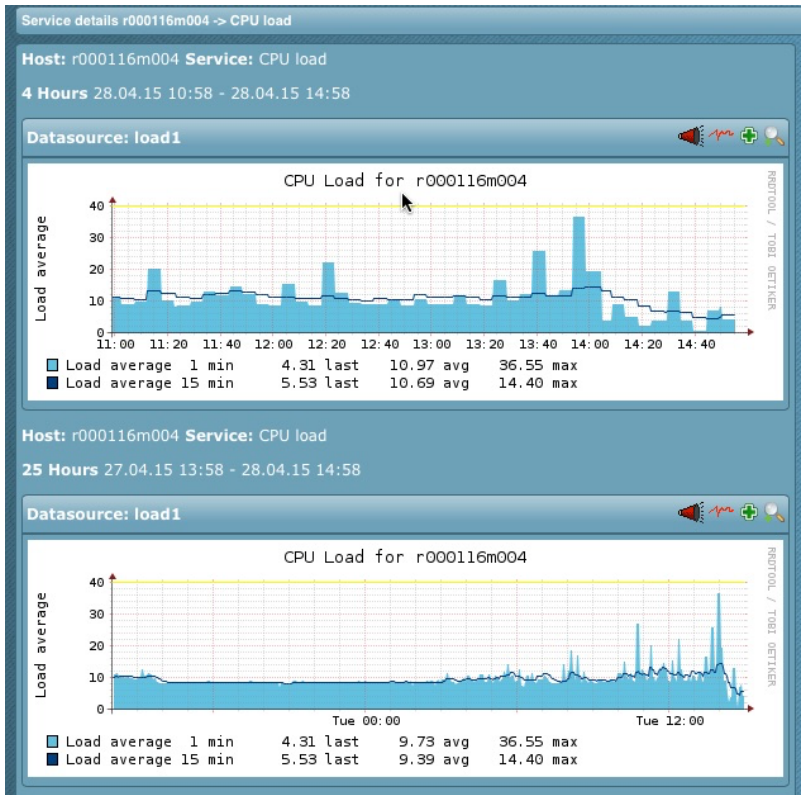

Figure 8: Screenshot of Cloud&Heat internal monitoring

To analyse the optimization techniques that minimize the cross-deployment traffic, we capture network traffic with Tcpflow. Figure 9 shows the standard Tcpflow report on traffic recorded.

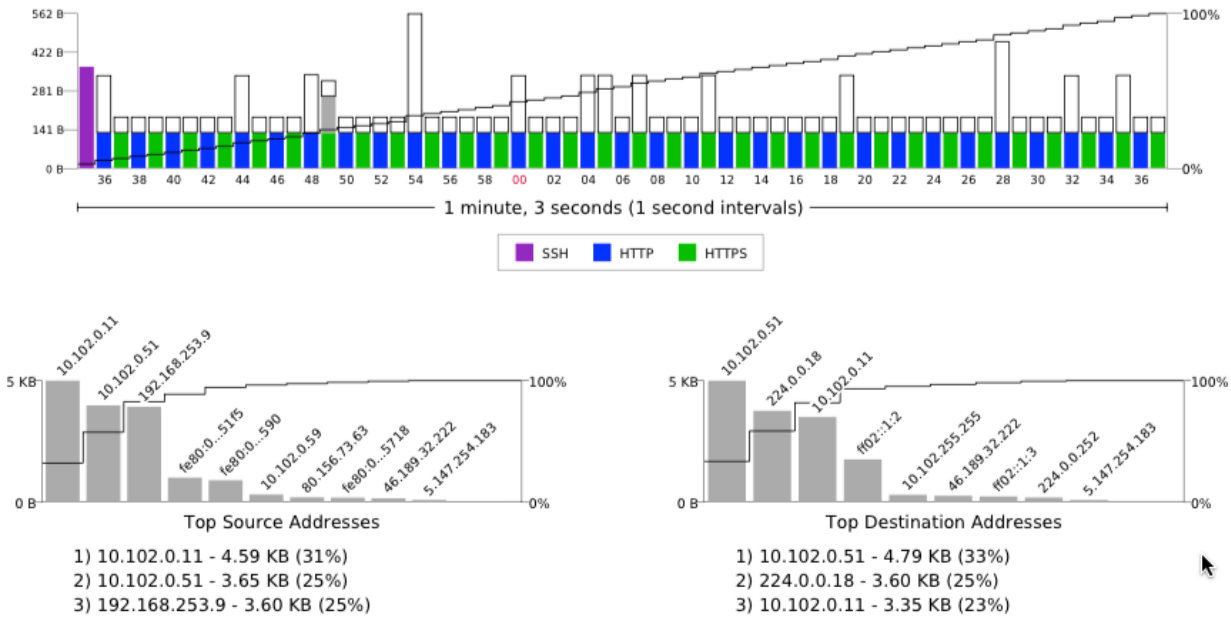

Figure 9: Screenshot of tcpflow report from Query Engine 3 in dresden2.

4. Evaluation of the LEADS platform running the application

As discussed, the representative application was constructed and deployed over the LEADS platform, and tested with real web-crawled data sets. We now present indicative results (size of extracted content, network transfer volume and indicative execution time) focusing on the two key aspects of the application:

a) The information extraction plugins, which are executed over the streams of crawled web pages (Section 4.1);
b) The query engine, used through its interface for SQL queries (Section 4.2).

Both plugins and SQL queries are crucial for adidas and for other potential users of the LEADS platform. On the one hand, plugins support performing arbitrary, user-specific processing and information extraction from the crawled web pages over the multi-user LEADS platform, thereby enabling sharing the cost of maintaining the crawl across all LEADS users. adidas uses plugins to extract useful information from the crawled web pages, such as sentiment of each web page, products, prices and keywords mentioned in the E-commerce web pages, etc. On the other hand, SQL queries provide the ideal language for a command-line interface and an Application Programming Interface, since the language is known and well-used. adidas uses SQL queries both through API (in the representative application) and through the command-line interface, to quickly collect the required data.

4.1 Information extraction on streams of crawled web pages

In order to test the correctness and performance of the information extraction functionality in the representative application, we have separately considered the two deployed information extraction plugins. The first plugin is called for all crawled web pages. It extracts the sentiment of each web page. The second plugin implements the following workflow:

1. Extraction of web page's domain;
2. Extraction of web page's text content;

3. Extraction of content's language;
4. Determination of an algorithm to extract valuable part of the content;
5. Extraction of valuable content;
6. Search for keywords inside of the valuable content and determination of relevance and sentiment of mentions.

The results produced by both plugins are stored over (distributed) ensemble caches in a relational format. Notice that both plugins perform computationally-expensive NLP tasks, as well as SQL queries that need to process a high data volume. As such, the ability of the LEADS platform to scale across micro-clouds is pivotal for the scalability of the plugins.

Both plugins were tested by initiating a live web crawl starting from two blog/news sites that have business interest for adidas as the crawling seeds. These sites are *runblogger.com* featuring running reviews and news as well as *footwearnews.com* featuring news on shoes. The resulting data set after one full crawl over a period contained a total of 3486 web pages of interest among all crawled pages. The extracted information was saved in 17.5 thousand records, in a relational table.

The two plugins were tested on both a single-cloud and a multi-cloud configuration. In the single-cloud configuration the plugins were deployed over 2 virtual machines (VMs) in micro-cloud hamm6. The multi-cloud configuration included two micro-clouds (hamm6 and dresden2), each hosting 2 VMs. All VMs were configured with 4 virtual processors and 8 GBs memory. In the following, we will be focusing on the multi-cloud deployment, which is more meaningful in terms of scalability.

Table 1 presents indicative processing rates for the two plugins, under the two different system configurations.

Table 1 Indicative plugins results

Plugin	Processing time per page (average)	Processing rate (average)	Total extracted information
1	22737 msec	633.3 pages/hour	17.5k records
2	230843 msec	62.3 pages/hour	7k records

4.2 Performance and validity of the LEADS query engine running the application SQL queries

The second set of experiments was focused on evaluating the query engine performance. For this, we used a larger data set, which was crawled for the M24 prototype. The data set contained 28 thousand web pages. Our experiment involved executing the following representative SQL queries, which are essential for the functionality described in deliverable D5.4:

```
Q1: SELECT K.uri, K.ts, K.keywords, K.sentiment AS sentiment FROM keywords K
       JOIN page_core C ON C.uri = K.uri
       WHERE  C.ts = K.ts
              AND K.partid like 'article_content:000'
              AND C.lang like 'en'
              AND K.ts>=0 AND K.ts<=1416504380252
              AND keywords IN ('adidas');
```

Intuitively, query Q1 requests all articles that are written in English, contain keyword 'adidas' and were crawled before time stamp '1416504380252' (expressed in Unix timestamp, i.e., as milliseconds

since January 1st, 1970). The query contains a join between two tables, i.e., table keywords that contains all keywords detected in each page, and table page_core which stores page-related information, e.g., the language for each web-page.

```
Q2: SELECT uri, ts FROM keywords
       WHERE  ts>=0 AND ts<=1416501600106
              AND partid = 'ecom_prod_name:000';
```

Query Q2 requests all pages of E-commerce sites that contain a product offer, as adidas plugin extracts a product name and price from this type of pages. We are interested only in versions of the pages site that were crawled with timestamp before '1416501600106'.

Table 2 Query answering

Query	#records	Execution time	Transfer volume
Q1	21	50 sec.	141 Mb
Q2	751	9.8 sec.	7.4 Mb

Table 2 summarises the number of returned records, execution time and transfer volume for the two queries when these are executed over the multi-cloud configuration described in the previous section.

4.3. Performance monitoring

Additionally to measuring the performance and the time for executing the queries, we investigate the performance metrics for the platform as a whole when running the application. We were interested in the node load generated by executing the queries, memory consumption, and sample packet exchange between tested nodes. The package exchange patterns also allow assessing whether the load is distributed properly.

Below, we present two examples from this work stream. Figure 10 depicts the CPU load (reported by the Linux kernel) distribution at each of the four nodes in the two micro-clouds, while running the SQL queries reported in Section 3. We observe that there is an almost even workload distribution across the nodes. Notice that this is a required property for scalability and high parallelization to avoid bottlenecks, and it is achieved due to a combination of the load balancing scheme (uniformly distributing of the records across all nodes/micro-clouds) and the distributed implementation of the SQL operators (each node processes the data contained in the local portion of the Ensemble). Similar pattern we may also see on Figure 11 that presents the disk writes during question executions.

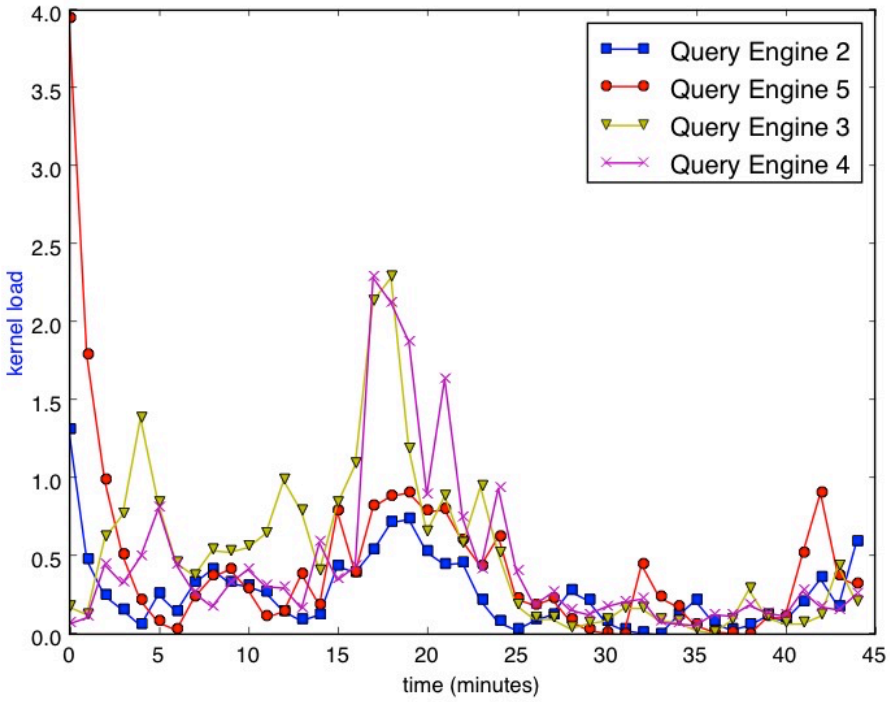

Figure 10: Node load running Query Engine Instances

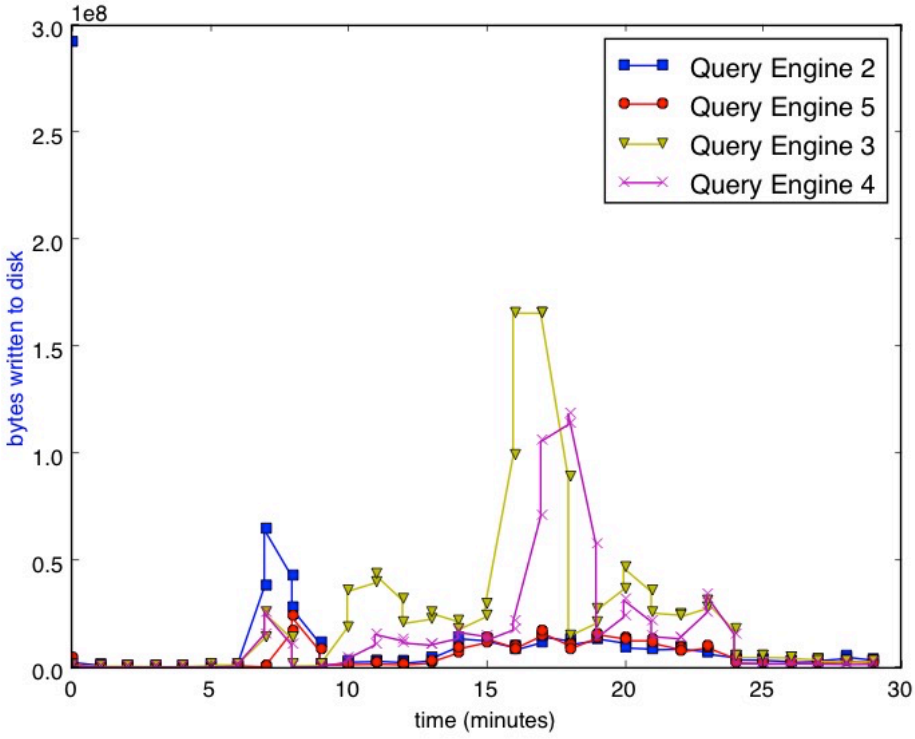

Figure 11: Write operations on the disk

5. Automation of LEADS cluster deployment

To democratize DaaS technology, we want not only to demonstrate its applicability in real world scenarios but also show how to use state-of-the-art tools to automatize its deployment in a multi-micro-cloud environment. Therefore, we build the cluster with one of the leading configuration management platforms --- SaltStack[6].

Saltstack provides a declarative language (YAML-based) to define infrastructure and provide tools to automatize the virtual machines creation and virtual machine provisioning. The YAML-based configuration files play a role of living documents and help us discuss its architecture internally. It is especially important for a distributed and diverse team like consortium of LEADS. In future, we may use the declarations as examples in the dissemination process. The configuration platform helps us also to have an agile (add and remove notes) and deterministic infrastructure (its definition is versioned in git).

The Salt stack-based provision is a part of a larger effort to integrate all the LEADS components. The integration is a base for full platform evaluation in T5.3 (WP5) due for M36.

Listing 1 shows node definitions for query engines. One can easily see their location, the size of instance (large), and operation system. We use this definition along with cloud (define, e.g., sizes of virtual machines available) and providers (define location of micro-clouds) to create five virtual machines in three micro-clouds.

```
ubuntu_large_hamm5:
    - leads-qe1

ubuntu_large_hamm6:
    - leads-qe2
    - leads-qe5

ubuntu_large_dresden2:
    - leads-qe3
    - leads-qe4
```

Listing 1: Node definition for micro clouds

After creating virtual machines, we use Saltstack to install necessary software components and configuration. Listing 2 depicts what should be installed on each of virtual machines. The names of steps (states in Saltstack vocabulary) are self-explanatory.

```
base:
  'leads-qe[12345]':
    - leads.packages
    - leads.adidas_plugin_deps
    - leads.java
    - leads.setup_script
  'leads-yarn-[123]':
    - leads.java
    - leads.yarn
  'leads-yarn-1':
    - leads.unicrawl
  'leads-yarn-hamm6-*':
    - leads.java
    - leads.yarn
  'leads-yarn-hamm6-1':
```

[6]See: http://saltstack.com/

```
    - leads.unicrawl
  'leads-yarn-dresden2-*':
    - leads.java
    - leads.yarn
  'leads-yarn-dresden2-1':
    - leads.unicrawl
  '*':
    - monitoring.pcp
  'leads-saltmaster':
    - monitoring.vector
```

Listing 2: Declaration of what should be installed on each of the nodes

We distribute the configuration for our multi-micro-cloud setup with a Saltstack master (installed in hamm5 as shown in **Error! Reference source not found.**). In this way, we have a central point to manage the LEADS cluster. The SaltStack master is provision with Saltstack, so we can recreate it in case of any issues.

We did not want to install (manage) anything then components for LEADS platform, therefore, instead of software repository, we use object storage provided by Cloud & Heat micro-clouds to distribute packages.

Configuration of cluster and individual machines can be found in a GitHub repository[7].

6. Conclusion

This document presented the effort of consortium in order to validate correctness and evaluate the integrated LEADS platform using the representative application in a multi-cloud environment. Based on our lessons-learned from M24 prototype, we have managed to extend application components, so that they bring greater business value. We have managed to integrate application with the rest of the platform and distribute it over multiple micro-clouds. We have managed to run both - processing plugins and queries - in multi-micro-cloud environment. We have equipped nodes of micro-clouds with tools that enable automatic cluster deployment and performance monitoring. They will help us when providing final optimizations for the final evaluation for M36.

[7] See: https://github.com/skarab7/leads_query-engine