Support Action
Big Data Europe – Empowering
Communities with Data Technologies
Project Number: 644564  Start Date of Project: 01/01/2015  Duration: 36 months

Deliverable 6.8 Pilot Evaluation and
Community-Specific Assessment III

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>December 31th, 2017</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>December 31th, 2017</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP6, Real-life Deployment &amp; User Evaluation</td>
</tr>
<tr>
<td>Task</td>
<td>T 6.3</td>
</tr>
<tr>
<td>Type</td>
<td>Report</td>
</tr>
<tr>
<td>Approval Status</td>
<td>Approved</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Number of Pages</td>
<td>121</td>
</tr>
<tr>
<td>Filename</td>
<td>D6.8_Pilot Evaluation and Community-Specific Assessment III</td>
</tr>
</tbody>
</table>

**Abstract:** Report summarizing the deployment of the second cycle of pilots, the obtained results, the evaluation of the results, and the acquired recommendations for improvements.

The information in this document reflects only the author’s views and the European Community is not liable for any use that may be made of the information contained therein.

The information in this document is provided “as is” without guarantee or warranty of any kind.
kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/her sole risk and liability.

History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>02/10/2017</td>
<td>Initial version</td>
<td>Ronald Siebes (VUA)</td>
</tr>
<tr>
<td>0.1</td>
<td>01/11/2017</td>
<td>First draft pilot questionnaires</td>
<td>Ronald Siebes + SC authors</td>
</tr>
<tr>
<td>0.2</td>
<td>11/12/2017</td>
<td>Final draft for review</td>
<td>Ronald Siebes + SC authors</td>
</tr>
<tr>
<td>0.3</td>
<td>14/12/2017</td>
<td>Reviewed draft</td>
<td>Alexandra Garatzogianni</td>
</tr>
<tr>
<td>1.0</td>
<td>30/12/2017</td>
<td>Final version</td>
<td>Ronald Siebes + SC authors</td>
</tr>
</tbody>
</table>

Author List

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Name</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open PHACTS Foundation</td>
<td>Kiera McNeice</td>
<td><a href="mailto:kiera@openphactsfoundation.org">kiera@openphactsfoundation.org</a></td>
</tr>
<tr>
<td>Institution</td>
<td>Name</td>
<td>Email</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>VU University Amsterdam</td>
<td>Ronald Siebes</td>
<td><a href="mailto:rm.siebes@few.vu.nl">rm.siebes@few.vu.nl</a></td>
</tr>
<tr>
<td>NCSR-D</td>
<td>S. Konstantopoulos, A. Charalambidis, I. Mouchakis, G. Stavrinos, A. Davvetas, I. Klampanos</td>
<td><a href="mailto:konstant@iit.demokritos.gr">konstant@iit.demokritos.gr</a></td>
</tr>
<tr>
<td>TenForce</td>
<td>Aad Versteden</td>
<td><a href="mailto:aad.versteden@tenforce.com">aad.versteden@tenforce.com</a></td>
</tr>
<tr>
<td>SWC</td>
<td>Victor Mireles, Jürgen Jakobitsch, Martin Kaltenboeck</td>
<td><a href="mailto:victor.mireles-chavez@semantic-web.com">victor.mireles-chavez@semantic-web.com</a>, <a href="mailto:juergen.jakobitsch@semantic-web.com">juergen.jakobitsch@semantic-web.com</a>, <a href="mailto:m.kaltenboeck@semantic-web.at">m.kaltenboeck@semantic-web.at</a></td>
</tr>
<tr>
<td>Fraunhofer IAIS</td>
<td>Alexandra Garatzogianni, Luigi Selmi</td>
<td><a href="mailto:Alexandra.Garatzogianni@iais.fraunhofer.de">Alexandra.Garatzogianni@iais.fraunhofer.de</a>, <a href="mailto:luigi.selmi@iais.fraunhofer.de">luigi.selmi@iais.fraunhofer.de</a></td>
</tr>
<tr>
<td>Agroknow</td>
<td>Maritina Stavrakaki</td>
<td><a href="mailto:pythk@agroknow.com">pythk@agroknow.com</a>, <a href="mailto:skonstan@agroknow.com">skonstan@agroknow.com</a>, <a href="mailto:maritina.stavrakaki@agroknow.com">maritina.stavrakaki@agroknow.com</a></td>
</tr>
<tr>
<td>SatCen</td>
<td>Michele Lazzarini</td>
<td><a href="mailto:michele.lazzarini@satcen.europa.eu">michele.lazzarini@satcen.europa.eu</a></td>
</tr>
</tbody>
</table>
Executive Summary

This document details the results and evaluation of the second round of pilots and provides the recommendations for the next piloting cycle. This document follows the methodology described in D6.4 and given the high variety in requirements and domains, the focus will be on a descriptive evaluation where the feedback of domain experts from within and outside the BDE consortium, and the feedback from the mid-term review meeting are leading the recommendations for the second cycle of pilots.

Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDI</td>
<td>The Big Data Integrator platform that is developed within Big Data Europe. The components that are made available to the pilots by BDI are listed here: <a href="https://github.com/big-data-europe/README/wiki/Components">https://github.com/big-data-europe/README/wiki/Components</a></td>
</tr>
<tr>
<td>BDI Instance</td>
<td>A specific deployment of BDI complemented by tools specifically supporting a given Big Data Europe pilot</td>
</tr>
<tr>
<td>BT</td>
<td>Bluetooth</td>
</tr>
<tr>
<td>ECMWF</td>
<td>European Centre for Medium range Weather Forecasting</td>
</tr>
<tr>
<td>ESGF</td>
<td>Earth System Grid Federation</td>
</tr>
<tr>
<td>FCD</td>
<td>Floating Car Data</td>
</tr>
<tr>
<td>LOD</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>SC1</td>
<td>Societal Challenge 1: Health, Demographic Change and Wellbeing</td>
</tr>
<tr>
<td>SC2</td>
<td>Societal Challenge 2: Food Security, Sustainable Agriculture and Forestry, Marine, Maritime and Inland Water Research and the Bioeconomy</td>
</tr>
<tr>
<td>SC3</td>
<td>Societal Challenge 3: Secure, Clean and Efficient Energy</td>
</tr>
<tr>
<td>SC4</td>
<td>Societal Challenge 4: Smart, Green and Integrated Transport</td>
</tr>
<tr>
<td>SC5</td>
<td>Societal Challenge 5: Climate Action, Environment, Resource Efficiency and Raw Materials</td>
</tr>
<tr>
<td>SC6</td>
<td>Societal Challenge 6: Europe in a changing world – Inclusive, innovative and reflective societies</td>
</tr>
<tr>
<td>SC7</td>
<td>Societal Challenge 7: Secure societies – Protecting freedom and security of Europe and its citizens</td>
</tr>
<tr>
<td>AK</td>
<td>Agroknow, Greece</td>
</tr>
<tr>
<td>CERTH</td>
<td>Centre for Research and Technology, Greece</td>
</tr>
<tr>
<td>CESSDA</td>
<td>Consortium of European Social Science Data Archives</td>
</tr>
<tr>
<td>CRES</td>
<td>Center for Renewable Energy Sources and Saving, Greece</td>
</tr>
<tr>
<td>FAO</td>
<td>Food and Agriculture Organization of the United Nations, Italy</td>
</tr>
<tr>
<td>FhG</td>
<td>Fraunhofer IAIS, Germany</td>
</tr>
<tr>
<td>InfAI</td>
<td>Institute for Applied Informatics, Germany</td>
</tr>
<tr>
<td>NCSR-D</td>
<td>National Center for Scientific Research “Demokritos”, Greece</td>
</tr>
<tr>
<td>OPF</td>
<td>Open PHACTS Foundation, UK</td>
</tr>
<tr>
<td>SWC</td>
<td>Semantic Web Company, Austria</td>
</tr>
<tr>
<td>UoA</td>
<td>National and Kapodistrian University of Athens</td>
</tr>
<tr>
<td>VU</td>
<td>Vrije Universiteit Amsterdam, the Netherlands</td>
</tr>
</tbody>
</table>
Table of contents

1. Introduction ........................................................................................................................................... 8
2. Rationale for the choice of evaluation methodologies ................................................................................. 12
3. Pilot evaluations ........................................................................................................................................ 14
   4.1 SC1: Life Sciences and Health ........................................................................................................... 15
   4.2 SC2: Food Security, Sustainable Agriculture and Forestry, Marine, Maritime and Inland
   Water Research and the Bioeconomy ........................................................................................................ 22
   BDE infrastructure questionnaire ............................................................................................................. 23
   File System ............................................................................................................................................. 23
   Resource Manager ................................................................................................................................. 25
   Scheduler ............................................................................................................................................... 27
   Coordination ........................................................................................................................................... 29
   Data Acquisition ...................................................................................................................................... 30
   Data Processing ...................................................................................................................................... 33
   4.3 SC3: Secure, Clean and Efficient Energy ............................................................................................ 38
   BDE infrastructure questionnaire ............................................................................................................. 39
   File System ............................................................................................................................................. 39
   Resource Manager ................................................................................................................................. 41
   Scheduler ............................................................................................................................................... 41
   Coordination ........................................................................................................................................... 43
   Data Acquisition ...................................................................................................................................... 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Processing</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>4.4 SC4: Smart, Green and Integrated Transport</td>
<td></td>
<td>49</td>
</tr>
<tr>
<td>BDE infrastructure questionnaire</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>File System</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Resource Manager</td>
<td></td>
<td>53</td>
</tr>
<tr>
<td>Scheduler</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>Coordination</td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>Data Acquisition and Message Passing</td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>Data Processing</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>4.5 SC5: Climate Action, Environment, Resource Efficiency and Raw Materials</td>
<td></td>
<td>65</td>
</tr>
<tr>
<td>BDE infrastructure questionnaire</td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>File System</td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>Resource Manager</td>
<td></td>
<td>71</td>
</tr>
<tr>
<td>Scheduler</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>Coordination</td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>Data Acquisition</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>Data Processing</td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>4.6 SC6: Europe in a changing world - inclusive, innovative and reflective societies</td>
<td></td>
<td>82</td>
</tr>
<tr>
<td>BDE infrastructure questionnaire</td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>File System</td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>Resource Manager</td>
<td></td>
<td>86</td>
</tr>
<tr>
<td>Scheduler</td>
<td></td>
<td>88</td>
</tr>
<tr>
<td>Coordination</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>Data Acquisition</td>
<td></td>
<td>91</td>
</tr>
</tbody>
</table>
1. Introduction

According to the planning in the Technical Annex WP6 – Real-life Deployment & User Evaluation – starts the end of year one and ends the third quarter of year three (cf. figure 1).
The pilot deployment will be done in three cycles, this is to have the possibility to improve, adjust and extend each pilot by the evaluation results of each previous cycle. Given the ambitious goal of the BDE project to provide an infrastructure that facilitates at least three versions of pilots for each of the seven Societal Challenges, the evaluation methodology for this second and third round of pilots build upon the results from Deliverable 6.3 "Pilot Evaluation & Community-specific Assessment" and the feedback from the mid-term reviews, the various webinars and workshops and other communications with the consortium and the BDE early adopters. The evaluation strategy.
and questionnaires for each of these Societal challenges are described in Deliverable D6.4. This document provides the answers on these questions.

![Diagram](image)

**Figure 2: pilot development and evaluation cycle**

During year one of the project, the first implementation of the BDE generic infrastructure was delivered in M12 and the final release is scheduled soon (M28). The final design and architectural decisions are based on the numerous feedback sessions and interviews with the project partners and the domain experts that are consulted by the
various domain partners and summarized in deliverable D6.3. The requirements and design specifications of the platform are described in deliverable D3.3, D3.5 and D5.2 which form the basis for the generic evaluation criteria. Basically, for each functional and non-functional requirement we now need to specify in which way we are going to evaluate if these requirements are met for the remaining two piloting phases in our project. It is important to realize that due to new insights, the choice of ‘core components’ that made up the BDE infrastructure was subject to major changes within the first two-year duration of the project. Since the thorough evaluation of the pilots and the platform, the final set of BDE core components that are currently being tested are carried by all the Societal Challenges and the community and we are confident that this leads to an integrated solution for every SC respectively.

The specification of the second cycle of pilots for each of the seven challenges is worked out in deliverable 5.4. For each of the seven challenges, a detailed pilot description is ready, implemented and evaluated. See figure 3 for the overall planning of
2. Rationale for the choice of evaluation methodologies

Deliverable 2.4.1 provides the results of an extensive requirements elicitation phase, which combined with the technical requirement analysis in deliverable 2.3 and the results of the interviews contain the functional and non-functional requirements for the BDE Platform. The goal of the
evaluation process is to investigate at which level these requirements are met during the various implementation phases of this project.

For the functional and non-functional requirements of the generic infrastructure part the FURPS model\(^1\) is followed which classifies software quality attributes:

- **Functionality** - Capability (Size & Generality of Feature Set), Reusability (Compatibility, Interoperability, Portability), Security (Safety & Exploitability)
- **Usability (UX)** - Human Factors, Aesthetics, Consistency, Documentation, Responsiveness
- **Reliability** - Availability (Failure Frequency (Robustness/Durability/Resilience), Failure Extent & Time-Length (Recoverability/Survivability)), Predictability (Stability), Accuracy (Frequency/Severity of Error)
- **Performance** - Speed, Efficiency, Resource Consumption (power, ram, cache, etc.), Throughput, Capacity, Scalability
- **Supportability** (Serviceability, Maintainability, Sustainability, Repair Speed) - Testability, Flexibility (Modifiability, Configurability, Adaptability, Extensibility, Modularity), Installability, Localizability

The details of each of these requirements are different for each challenge and need to be addressed as such in our evaluation strategy. However, as mentioned before, the

\(^1\) "FURPS - Wikipedia, the free encyclopedia." 2011. 2 Nov. 2015 <https://en.wikipedia.org/wiki/FURPS>
generic BDE infrastructure also can be evaluated independent from these challenges according to the FURPS model. The system and societal challenge specific evaluation results are described next in the main chapter of this deliverable.

3. Pilot evaluations

The second cycle of pilots is specified in Deliverable 5.4. As mentioned before, each pilot will be evaluated on BDE generic and pilot specific requirements.

Deliverable 3.2 provides the details on requirements and the initial choice of software components for the generic BDE infrastructure. The functional and non-functional requirements were gathered from all the societal challenges categorized by the four Big Data challenges namely Volume, Velocity, Variety and Veracity of data. It has been found that the data requirements include all the four features of Big Data with particular focus on volume and velocity. The analysis of data value chain has revealed that each societal challenge has a different set of requirements resulting in a diverse set of tools and frameworks required for each step of handling in the data value chain.

Deliverable 5.4 outlines the choices of the components that form the BDE infrastructure with respect to the demands specified by the individual pilots. Hence, the evaluation plan for the second and third pilot cycle regarding the generic infrastructure will mainly be focused on gathering feedback on the selected tools, especially whether or not they fulfill the requirements from the individual pilots.
To achieve this goal, we will ask the leading technical persons and the leading domain experts for each selected pilot to fill out a questionnaire during the second evaluation period (M28). This questionnaire addresses the non-functional and functional requirements for each task that the BDE infrastructure should support.

4.1 SC1: Life Sciences and Health

The pilot is carried out by OPF and VU in the frame of SC1 Health, Demographic Change and Wellbeing.

The pilot demonstrates the workflow of reproducing the functionality of an existing data integration and processing system (the Open PHACTS Discovery Platform) on BDI. The third pilot extends the first and second pilots (cf. D5.2, Section 2) with the following:

- Further discussions with stakeholders and other Societal Challenges will identify how the existing Open PHACTS platform and datasets may potentially be used to answer queries in other domains. In particular applications in Societal Challenge 2 (food security and sustainable agriculture), for which the effects of chemistry (e.g. pesticides) on biology are probed in plants, could exploit the linked data services currently within the OPF platform. This will require discussion use case specifics with SC2 to understand their requirements, and ensure that the OPF data is applicable. Similarly we will explore whether SC2 data which could be linked to the OPF data platform is relevant for early biology research.

- No specific new datasets are targeted for integration in the third pilot. However if datasets to be made available through other pilots have clear potential links to Open PHACTS datasets, these will be considered for integration into the platform to offer researchers the ability to pose more complex queries across a wider range of data.

- The third pilot will aim to expand on first pilot by refreshing the datasets integrated into
the pilot. Homogenising and integrating the new data available for these datasets, and developing ways to update datasets by integrating new data on an ongoing basis, will enable new use cases where researchers require fully current datasets for their queries.

Key evaluation questions:

1. Did you manage to store all the RDF data and answer curated queries in a reasonable time?

We positively tested the storage on various machines. These minimum system requirements will have to be met in order to get the storage functional:

- 250 GB SSD disk space
- 32 GB RAM memory
- 8 CPU cores
- Recent x64 Linux distribution (e.g. Ubuntu 14.04 LTS, Centos 7)
- Docker 1.7.1 or later
- Good internet connection when loading external data (only needed at setup stage). This takes around 6 hours with 8Mb/s connection.
- Docker Compose 1.5.2 or later
- Tested on: Centos 6.7 (with kernel 3.18.21-17.el6 - yum install centos-release-xen ; yum update)
- Tested on: Ubuntu 14.04 LTS

We also positively tested the API as documented in the SWAGGER file from which the web interface is rendered.
2. Were you able to fill the Puella SPARQL templates with the HTTP-GET parameters to execute RDF queries on the 4Store DB?

Yes, Demokritos demonstrated a successful execution where the Virtuoso RDF component is exchanged with 4Store. Part of this effort is published at the BLINK workshop².

3. How many of the 21 Open PHACTS research questions³ were you able to answer?

With the Virtuoso RDF Docker component we can answer 18 of the 21 questions. The reason for not being able to answer 3 is because they depend on the patent data set which is not available as an open dataset, which is one of the requirements in the BDE project.

² Antonis Troumpoukis, Angelos Charalambidis, Giannis Mouchakis, Stasinos Konstantopoulos, Ronald Siebes, Victor de Boer, Stian Soiland-Reyes and Daniela Digles. Developing a Benchmark Suite for Semantic Web Data from Existing Workflows. Workshop on Benchmarking Linked Data (BLINK), ISWC conference, October 18, 2016, Kobe, Japan

4. Were you able to identify possible applications of the linked data services currently within the OPF platform to other societal challenges?

SC2 was identified as a potential opportunity to apply the Open PHACTS pilot to new datasets, and link data across the two societal challenges. Pursuing this idea proved to be beyond the scope and timeframe of the BDE project, but the Open PHACTS Foundation will continue to look for ways to expand the platform to connect to new kinds of data.

5. Were you able to implement automated source data refreshes within the BDI framework?

Implementing automated data refreshes in a platform that links data from multiple sources, with multiple release schedules and systems, is a complex problem. The Open PHACTS Foundation's work on automating data refreshes in the original platform ended up taking many months longer than expected, and the next data refresh is now scheduled for January 2018, after the end of the BDE project. It was therefore not possible to attempt to transfer the approaches still under development by the Open PHACTS Foundation to the BDI tech stack during the project. However the Open PHACTS Foundation will continue to work towards automating data refreshes, if at all possible in a way that can be Dockerised and compatible with the BDI infrastructure.
Other evaluation questions based on the requirements specified in D5.4:

Table 1: Evaluation questions for the first SC1 pilot

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
</table>
| R1 | The solution should be packaged in a way that makes it possible to combine the Open PHACTS Docker and the BDE Docker to achieve a custom integrated solution. | Is SWAGGER suitable for specifying a REST API on a dynamic distributed environment like the BDE infrastructure? 
Yes, the Docker approach has a very intuitive networking approach which makes it easy to deploy the RESTful API in a secure way and to set-up a gateway via a public IP address. The Web GUI rendering of the SWAGGER script is done via a custom PHP script on the Puelia RDF web server. |
| R2 | RDF data storage. | What are the experiences with the Docker version of the open-source Virtuoso software with respect to this pilot? 
In general the software is performing very well: it is stable and able to load all the data in a reasonable amount of time (around 4 hours) on a high-end consumer PC. One issue that needs improvement is the way the |
Docker stack deals with temporal node failures and recovery from the cache files. Now, if for any reason the Virtuoso process halts, the whole data loading procedure (which takes more than four hours) has to be repeated and can be prevented by using the automatic persistent caching mechanism provided by Virtuoso.

Which other RDF stores did you use (e.g. 4Store or Sansa)?

We have not yet used any other RDF stores.

<table>
<thead>
<tr>
<th>R3</th>
<th>Datasets are aligned and linked at data ingestion time, and the transformed data is stored.</th>
<th>How does the BDE infrastructure communicate with the external Instance Mapping Service provider?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The IMS became an independent docker module and is initiated via the docker compose script. This allows to customize various parameters that deal with the communication between the various other components like the Puelia RDF abstraction layer, the Virtuoso store, the IMS etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>What is the difference in delay between the current OPS system and the pilot version?</td>
</tr>
</tbody>
</table>
When running the whole OPS docker on the same LAN the delay is in the order of milliseconds which is much faster than the current OPS system.

<table>
<thead>
<tr>
<th>R4</th>
<th>Queries are expanded or otherwise processed and the processed query is applied to the data.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Were there any changes in the SPARQL templates needed due to the transition to another RDF store (it’s known that some providers include extra ‘shortcuts’ and functionality next to the SPARQL standard)?</td>
</tr>
<tr>
<td></td>
<td>At this moment there were no changes needed because the first cycle of this pilot uses the same components as the current OPS system. The only difference is that the OPS system uses the commercial version of Virtuoso and the OPS-docker stack uses the open-source version of Virtuoso, however there is no difference in the language constructs between both versions. In the next cycle we are going to experiment further with other RDF stores like 4Store. Most likely this will result in making adaptations in the query constructs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R6</th>
<th>Data and query security and privacy requirements.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Are there currently vulnerabilities in the BDE infrastructure that might reveal any sort of</td>
</tr>
</tbody>
</table>
communication to a 3rd party (e.g. queries and results, or ip addresses)?

The current OPS-docker has no additional ACL functionality besides the built-in options from the Docker stack. The Docker stack provides sufficient mechanisms to 'shield' components from outside access by maintaining a virtual local network stack and bridging them internally. When desired the docker configuration file provides the possibility to proxy incoming and outgoing traffic to any of the components, like the OpenPHACTS explorer (part of the OPS-docker).

4.2 SC2: Food Security, Sustainable Agriculture and Forestry, Marine, Maritime and Inland Water Research and the Bioeconomy.

The pilot is carried out by AK, FAO, and SWC in the frame of SC2 Food Security, Sustainable Agriculture and Forestry, Marine, Maritime and Inland Water Research and the Bioeconomy.

The third pilot cycle aims to lowering the SC2 community boundaries, providing a demonstration system which communicates the BDE message in a user-friendly, community oriented mean. To this end, the third pilot extends the first and second cycle (cf. D5.2, Section 2) by providing an engaging, intuitive graphical web interface addressing key data-oriented questions relevant
to the Viticulture Research Community.
The functionalities (and data) remain the same with the previous pilot version (cycle 2). As a consequence, the infrastructure questionnaire which follows is the same with that of the previous version.

BDE infrastructure questionnaire

File System

*The platform requires a distributed file system which provide storage, fault tolerance, scalability, reliability, and availability to the multitude of SC partners.*

This has resulted in selection of Apache Hadoop Distributed File system, HDFS.

**Evaluation questions:**

- How much data did you store?
  - 1.8M RDF Triples
- How much data did you process?
- 200 Mb Tabular Data (Genetic profiles and Ampelographic Descriptions stored as CSV files)
- Up to 53 years weather data (~1.2 TB) retrieved from 38 weather stations

- Please specify which storage mechanism(s)/tool(s) you used (e.g. HDFS, a structured database like Postgres, an RDF store like Virtuoso, a NoSQL store like Redis)
  - An RDF Store (GraphDB)
  - HDFS

- For every mechanism you mentioned, please answer the following questions:
  - Is the store being managed via the BDE infrastructure?
    - No
  - If no, why not?
    - VITIS, the Viticulture Research and Practice Gateway, that was extended using the BDE technologies, requires a persistent datastore.
  - If yes:
    - How much effort/time did it take to setup and understand the tool/mechanism?
- We have invested around 1 pm altogether in setup and development, although this is an ongoing process to stay up to date.

- Was the file system able to store and retrieve the files that you need for running the pilot?
  - Yes. The usage of Apache Flume allows for multiple file systems by default. The pilot was tested using HDFS and the default linux file system given on the test servers.

- Did you experience any problems related to fault tolerance, scalability, reliability and availability?
  - No. We could easily determine that the designed architecture is scalable. The tools that form the core of SC2's architecture didn't expose any reliability problems within the scope of the pilot.

- Was the upload time satisfactory?
  - Upload had been done using "scp" (secure copy) into the linux file system of the test servers. Insertion of files into HDFS from a local file system went without trouble.

Resource Manager
The platform should be able to provide resource management capabilities and support schedulers for high utilization and throughput.

This set of properties is delivered by Docker Swarm which offers optimal resource management for distributed applications.

Evaluation questions:

- How many applications do you run in parallel during your pilot?
  - 8: Poolparty, CSV->RDF converter, SWC Unified Views, AK AKSTEM, AK VITIS, Apache Flume, Apache Kafka, Graph DB Triple Store

- Did you need an automatic resource manager to delegate the resources for distributed applications?
  - Yes

- If yes, did you use Docker Swarm?
  - If no, what did you choose, and why not Docker Swarm?
  - If yes
    - Is it easy to set up for your pilot requirements?
      - Yes.
- Which type of interface did you use to interact with Docker Swarm (e.g. the HTTP REST API, a Web interface, the command line shell).
  - **Command line**
  - Storing and retrieving files?
  - How did you describe the ‘tasks’ and was the ‘matchmaking’ by Docker Swarm finding the appropriate resources satisfactory?
    - **N/A**

- Any other remarks?

## Scheduler

*The scheduler needs to schedule the distributed tasks and offer resources to increase the throughput of overall system.*

Two schedulers Marathon and Chronos have been selected for task scheduling in the framework for the first release of the platform. Since Mesos is not part of the BDE platform anymore, also both schedulers are being deprecated. Instead now the INIT_DEAMON and Docker Healthchecks are being used.
Evaluation questions:

Many system administrators heard of Cron(jobs) that at certain time intervals or other conditions manage starting and stopping processes.

- Since Marathon and Chronos are being deprecated, which schedulers are now being considered and even integrated?
  - We tested the whole pilot on docker swarm and Marathon to our full satisfaction.

- Is there need for schedulers? If yes, where and when?
  - No

- What are your experiences with INIT_DEAMON?
  - Was not really required by our pilot, since the components took care of this within the scope of the application as well as the startup sequence within docker swarm was enough. For example Apache Flume created Apache Kafka topics and wouldn't startup if there was an error. On the other hand it was made sure within the docker swarm snippet that Apache Kafka started before Apache Flume.

- What are your experiences with Docker Healthchecks?
  - N/A
Coordination

The platform requires an efficient system for managed state, distributed coordination, consensus and lock management in the distributed platform.

ZooKeeper will be used as a decentralized tolerant coordination framework.

Evaluation questions:
The most important feature of ZooKeeper is to make sure the processes keep running and communicating. It responds to node crashes by for example delegating the job to another node.

- Does your pilot contain a set of continuously running applications that interact with each other?
  - Yes
- If yes, did you choose for a coordination framework to keep the processes stable?
  - Yes
- If yes, which framework did you choose, and was it doing what you expected?
**docker swarm and Marathon**

- Did you experience any problems like physical node failures or memory leaks that resulted in the coordinator to interfere?
  - No
- If you chose something else than Zookeeper, please explain why.
- If you chose for Zookeeper, what were your experiences?
  - **Zookeeper was running stable and is required by most components that form part of SC2’s architecture.**

### Data Acquisition

*Owing to the wide range of input data properties, a set of tools is needed to support the process of gathering, filtering and cleaning data before it is put in a data warehouse or any other storage solution on which data processing can be carried out during data acquisition*

The set of frameworks including Apache Flume and Apache Kafka have been chosen with an ambition that it would cater for the all the four properties of Big Data.

*Apache Flume: A framework to populate HDFS with streaming event data.*
Apache Kafka: A framework to transfer reliable messaging between applications.

Evaluation questions:
Most likely some of the data you use in your first pilot you already had available in one form or another. The ‘data acquisition’ process is the bridge between the BDE storage mechanism and the data.

- Is your data ‘special’ in order that it needs some tool to transform or process the data so that it can be stored in the respective data-store (for example streaming data, XML, data that needs to be transformed to JSON, data that needs to be migrated between DBS, etc)?
  - CSV data that has to be transformed to RDF triples
- If yes, which tools did you use for this pilot?
  - Custom made parsers and converters, interacting with Poolparty Semantic Suite.
- What are your experiences with Apache Kafka?
  - Works stable and as expected. Interaction with Apache Spark works without much trouble. The only thing that is to be mentioned here is versioning. This could be improved especially if Apache Kafka is used in connection with Apache Spark. This however is outside the scope of the BDE project.
What are your experiences with Apache Flume?

- Works stable and as expected. We've extended Apache Flume's HDFS source to include parent directory headers due to the nature of our pilot. Files need to be stored in structured subdirectories and shouldn't be stored in one single large directory. This extension worked without issues.

Are any of these tools part of the generic BDE infrastructure?

- Both Apache Flume and Apache Kafka.
- If no, would you recommend it to add it?
  - If yes,
    - was it difficult to learn and setup the tool?
      - In general those tools and their application need to be understood. It takes some time but not difficult. Training for users of the platform is necessary.
    - What is the weakest link in the pipeline (e.g. the store, the network, the tooling), and was it still performing well enough to be satisfactory for this pilot?
      - We didn't have experience with especially weak points. It is understood that all components need to work seamlessly in a productive environment.
Data Processing

Data Processing Frameworks: The platform requires different frameworks for various SC instances. Each framework has a different set of strengths and is applicable for a specific set of properties of the underlying data.

A multitude of tools are available for the type of processing to be performed on the underlying data, this includes, but is not limited to MapReduce for batch processing, Spark GraphX for iterative processing, Apache Spark and Apache Flink for data stream and real time processing.

Evaluation questions:

- Which tools do you use to transform your (raw) data in order to, for example, perform analysis, filter noise, make it suitable for sorting and querying etc?
  - Custom made parsers and converters, interacting with Poolparty Semantic Suite and AK VITIS.

- Which of these tools are being supported by the generic BDE infrastructure? And did you use them?
  - Above parsers and converters are wrapped inside Apache Spark Jobs so they are supported by the BDE infrastructure.
- Did the tools you use, and which are supported by the generic infrastructure, do what you expected? Were they easy to use? Please give a summary about your experiences for each data transformation tool.

  We created a general Big Data Pipeline for data processing that can be used for many use cases with minor adaptations. HDFS/Linux FileSystem worked fine with Apache Flume, Apache Kafka and Apache Spark.

Evaluation questions based on the requirements specified in D5.4:

Table 3: Requirements of the Second SC2 Pilot

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>In case of failures during data processing, users should be able to recover data produced earlier in the workflow along with their lineage and associated metadata</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Were you able to periodically store intermediate results and their lineage metadata? How?</td>
</tr>
<tr>
<td></td>
<td>Through the use of Apache Flume the pipeline is continuously accepting new input. Through the use of Apache Kafka those chunks of data of any kind are distributed in a failsafe manner, meaning that the system knows which chunks of data have already been processed successfully. Apache Kafka stores which messages have successfully been processed by the data processing unit (e.g. a spark job). As an example, if the HDFS system is full, Apache Kafka can continue where it left off when the HDFS system gets more space</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R2</th>
<th>Extracting images and their captions from scientific publications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Although the overall architecture is data model and format agnostic by design (data is moved between components as streams of bytes), it</td>
</tr>
<tr>
<td>R3</td>
<td>Extracting thematic annotations from text in scientific publications</td>
</tr>
<tr>
<td>R4</td>
<td>Extracting researcher affiliations from the scientific publications</td>
</tr>
<tr>
<td>R5</td>
<td>Variety identification</td>
</tr>
<tr>
<td>R6</td>
<td>Phenolic modeling</td>
</tr>
</tbody>
</table>

exposes a specific point of control where variety of data formats can easily be tackled. For our use case we created a plugin system using Java’s Service Provider Interfaces (SPI), so as to enable framework extension and component replaceability. The different data types provided on harvesting is unified such that the rest of the pipeline can work on one data model and format. Parallelization is guaranteed by the data processing unit in use like Apache Spark or Apache Flink. Apache Kafka will additionally make sure that data messages are distributed evenly amongst all consumers, e.g. a Spark job, running on multiple nodes. All the extraction / identification and modeling tasks have been implemented as Spark jobs.
### R7 Expose data and metadata in JSON through a Web API

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Was the Data ingestion module implemented to write JSON documents on a HDFS?</strong></td>
<td><strong>Did you create a SPARQL endpoint returning the required results in JSON?</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Instead of 4store, we used the GraphDB triplestore. The reason for this choice was based on the fact that Garlik terminated the support and maintenance of the 4store project (<a href="https://github.com/4store/4store">https://github.com/4store/4store</a>) from 2015.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>GraphDB provides the SPARQL endpoint for accessing all the data and responds with results in JSON.</strong></td>
</tr>
</tbody>
</table>

### SC2 Summary

The key evaluation questions of SC2 Pilot Cycle 2 that were presented in D6.3, remained the same for the SC2 Pilot Cycle 3.
In summary, SC2 Pilot Cycle 2 and 3 demonstrated the ability of BDE proposed technologies to extend and scale up the processing workflows to handle a variety of data types (beyond bibliographic data) relevant to Viticulture. To this end, the Flume/Kafka pipeline was extended to handle other than bibliographic data (e.g. sensor/weather data) and inclusion of use case scenarios which combine/link more heterogeneous data sources. The BDE proposed technologies that were evaluated successfully during the SC2 Pilot Cycle 1, proved to be equally capable of handling other data types also in the case of SC2 Pilot Cycle 2. Therefore, the choice of the technologies used for the purposes of the SC2 Pilot proved to be successful. Especially for cycle 3, an engaging, intuitive graphical web interface was created to lower the SC2 community boundaries.

4.3 SC3: Secure, Clean and Efficient Energy

The pilot is carried out by CRES in the frame of SC3 Secure, Clean and Efficient Energy.

The second pilot cycle extends the first pilot by adding additional data analysis on raw data regarding Acoustic Emissions (AE) sensors and aggregated data such as parametrics from continuous monitoring systems (CMS). The pilot demonstrates the following workflow: a developer in the field of wind energy enhances condition monitoring for each unit in a wind farm by pooling together data from multiple units from the same farm (to consider the cluster operation in total) and third party data (to perform correlated assessment). The custom analysis modules created by the developer use both raw data that are transferred offline to the processing cluster and condensed data streamed online at the same time order that the event occurs.

The following datasets are involved:
- Raw sensor and SCADA data from a given wind farm
- Raw sensor data from Acoustic Emissions module from a given wind farm.
- Online stream data comprised of parametrics and statistics extracted from the raw
SCADA data. All data is in custom binary. Files contain a metadata header and in tabulated form the signal data (signal in columns, time sequence in rows). All data is annotated by location, time, and system id.

The following processing is carried out:

- Periodic execution of parametrized models to return operational statistics, warnings including correlation analysis of data across units
- Periodic execution of operational statistics
- Periodic execution of model parametrization
- Periodic specific acoustic emissions DSP

The following outputs are made available for visualization or further processing:

- Operational statistics
- Model parameters

BDE infrastructure questionnaire

File System

The platform requires a distributed file system which provide storage, fault tolerance, scalability, reliability, and availability to the multitude of SC partners.

This has resulted in selection of Apache Hadoop Distributed File system, HDFS.
Evaluation questions:

- How much data did you store?
  
  The data acquisition is continuing. Currently 20TB are under processing for second cycle.

- How much data did you process?
  
  All data are processed. After periodic optimization of processing parameters the complete raw database is revisited.

- Please specify which storage mechanism(s)/tool(s) you used (e.g. HDFS, a structured database like Postgres, an RDF store like Virtuoso, a NoSQL store like Redis)
  
  HDFS for raw data and Postgres for analytics results.
Resource Manager

*The platform should be able to provide resource management capabilities and support schedulers for high utilization and throughput.*

This set of properties is delivered by Docker Swarm which offers optimal resource management for distributed applications.

**Evaluation questions:**

- How many applications do you run in parallel during your pilot?
  The main application is the raw data processing and periodically the analytics module is operating.

- Did you need an automatic resource manager to delegate the resources for distributed applications?
  No.

Scheduler
The scheduler needs to schedule the distributed tasks and offer resources to increase the throughput of overall system.

Two schedulers Marathon and Chronos have been selected for task scheduling in the framework for the first release of the platform. Since Mesos is not part of the BDE platform anymore, also both schedulers are being deprecated. Instead now the INIT_DEAMON and Docker Healthchecks are being used.

Evaluation questions:

Many system administrators heard of Cron(jobs) that at certain time intervals or other conditions manage starting and stopping processes.

- Since Marathon and Chronos are not being deprecated, which schedulers are now being considered and even integrated?
- Is there need for schedulers? If yes, where and when?
  No.
- What are your experiences with INIT_DEAMON?
- What are your experiences with Docker Healthchecks?
Coordination

The platform requires an efficient system for managed state, distributed coordination, consensus and lock management in the distributed platform.

ZooKeeper will be used as a decentralized tolerant coordination framework.

Evaluation questions:
The most important feature of ZooKeeper is to make sure the processes keep running and communicating. It responds to node crashes by for example delegating the job to another node.

- Does your pilot contain a set of continuously running applications that interact with each other?
  No.
- If yes, did you choose for a coordination framework to keep the processes stable?
  No.
- If yes,
  o  which framework did you choose, and was it doing what you expected?
Did you experience any problems like physical node failures or memory leaks that resulted in the coordinator to interfere?

If you chose something else than Zookeeper, please explain why.

If you chose for Zookeeper, what were your experiences?

Data Acquisition

Owing to the wide range of input data properties, a set of tools is needed to support the process of gathering, filtering and cleaning data before it is put in a data warehouse or any other storage solution on which data processing can be carried out during data acquisition.

The set of frameworks including Apache Flume and Apache Kafka have been chosen with an ambition that it would cater for the all the four properties of Big Data.

*Apache Flume:* A framework to populate HDFS with streaming event data.

*Apache Kafka:* A framework to transfer reliable messaging between applications.

Evaluation questions:
Most likely some of the data you use in your first pilot you already had available in one form or another. The ‘data acquisition’ process is the bridge between the BDE storage mechanism and the data.

- Is your data ‘special’ in order that it needs some tool to transform or process the data so that it can be stored in the respective data-store (for example streaming data, XML, data that needs to be transformed to JSON, data that needs to be migrated between DBS, etc)?

**Structured binary data with documented open format.**

- If yes, which tools did you use for this pilot?

  **Custom executables for transforming data to be handled by SPARK.**

- What are your experiences with Apache Kafka?
- What are your experiences with Apache Flume?
- Are any of these tools part of the generic BDE infrastructure?
  - If no, would you recommend it to add it?
  - If yes,
    - was it difficult to learn and setup the tool?
    - What is the weakest link in the pipeline (e.g. the store, the network, the tooling), and was it still performing well enough to be satisfactory for this pilot?
Data Processing

Data Processing Frameworks: The platform requires different frameworks for various SC instances. Each framework has a different set of strengths and is applicable for a specific set of properties of the underlying data.

A multitude of tools are available for the type of processing to be performed on the underlying data, this includes, but is not limited to MapReduce for batch processing, Spark GraphX for iterative processing, Apache Spark and Apache Flink for data stream and real time processing.

Evaluation questions:

- Which tools do you use to transform your (raw) data in order to, for example, perform analysis, filter noise, make it suitable for sorting and querying etc?

  For raw data processing custom executables containing standardised analysis procedures (such as IEC-61400) are used.
- Which of these tools are being supported by the generic BDE infrastructure? And did you use them?

Algorithms and procedures can be reprogrammed in the tools available in BDI, yet the use of independent executables offer the use of proprietary tools (if needed) and shorten the development time.

- Did the tools you use, and which are supported by the generic infrastructure, do what you expected? Were they easy to use? Please give a summary about your experiences for each data transformation tool.

Evaluation questions based on the requirements specified in D5.4:

**Table 4: Requirements of the Second SC3 Pilot**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>What are your experiences with the data connector that provides for receiving OPC streams from an OPC-compatible server?</td>
</tr>
</tbody>
</table>
source tools for OPC client was not attained. OPC operation remains a demonstration only aspect of the pilot supporting SCADA data processing. The main content of second cycle regarded the Acoustic Emission monitoring.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>The application should be able to recover from short outages by collecting the data transmitted during the outage from the data sources</td>
</tr>
<tr>
<td></td>
<td>What are your experiences with the OPC data connector that is developed to retrieve the missing data collected at the intermediate level from the distributed data historian systems? All data are stored in the intermediate level. Missed synoptic data are retrieved along with the raw data with the foreseen procedure (due to high volume of pilot case data this is periodic).</td>
</tr>
<tr>
<td>R3</td>
<td>Near-realtime execution of parameterized models to return operational statistics, including correlation analysis of data across units</td>
</tr>
<tr>
<td></td>
<td>Was the analysis software able to write its results back into a specified format and data model? Was it appropriate input for further analysis? Data processing and analysis results are delivered in tabulated ASCII and database format. The use of custom executables presented high flexibility in accessing the binary data file format.</td>
</tr>
<tr>
<td>R4</td>
<td>The GUI supports database querying and data visualization</td>
</tr>
<tr>
<td></td>
<td>Was the GUI able to access files in the format and</td>
</tr>
</tbody>
</table>
4.4 SC4: Smart, Green and Integrated Transport

The pilot is carried out by FhG and CERTH in the frame of SC4 Smart, Green and Integrated Transport.

The pilot demonstrates how to implement the workflow for the ingestion, processing and storage of near real-time and historical floating car data (FCD) in a distributed setting. The pilot demonstrates the following workflows:

- The monitoring of the current traffic conditions in the city of Thessaloniki using the near real-time floating car data available for 1200 vehicles.
- The forecasting of future traffic conditions based on a model trained from the historical and the near real-time FCD data.

The data from each vehicle contains the vehicle's identifier, valid for only 24 hours, speed, orientation, latitude and longitude. In order to be used as a proxy for the traffic flow in each road segment the location of the vehicle must be matched with the road segment in which the vehicle is being driven. The near real-time Floating Car Data (FCD) stream is generated by the taxi fleet and is provided via web services. The historical data is provided via FTP and contains 40 GB of data. The road network is extracted from the OpenStreetMap database. The map matching can be performed in parallel and is based on R scripts and stored procedures that make queries on
a geographical database using topological rules. The matched records are aggregated in time windows in order to compute the average flow (number of vehicles) and speed within each time window. The result of the aggregation are records containing the identifier of a road segment, the traffic flow (number of vehicles in the time window), the average speed and the timestamp. The result data is stored in a distributed database.

The 1st pilot showcased the implementation of a traffic monitoring service using the BDE components and the FCD web service. The objective of the second pilot was on one hand to implement a traffic forecasting service and on the other hand to deploy the pilot in a distributed setting using Docker swarm. The forecasting service is based on a machine learning algorithm. The historical data is used to train a model for each road segment that can learn the pattern of the traffic flow. The algorithm has been developed in R and integrated within a Flink application. In developing the code for the pilot we had to mediate between two opposing constraints. On one hand the knowledge of the domain experts that use R as their main programming language and on the other hand the need to integrate the R scripts in a distributed Java application. The solution was to deploy on each node a component with PostGis and R server with all the scripts. Another issue was the difficulty of integrating and deploying many services in a Docker swarm. The technology is still quite new and subject to many changes. In order to test the deployment of docker containers and run them using a docker compose file we have developed a simplified version of the pilot that uses a subset of the containers and HDFS for the storage of the FCD data and the result of the processing.

BDE infrastructure questionnaire

File System
The platform requires a distributed file system which provide storage, fault tolerance, scalability, reliability, and availability to the multitude of SC partners.

This has resulted in selection of Apache Hadoop Distributed File system, HDFS.

Evaluation questions:

- How much data did you store?
  The most relevant data is the historical FCD data, about 40 GB

- How much data did you process?
  We processed 250 MB of data for testing

- Please specify which storage mechanism(s)/tool(s) you used (e.g. HDFS, a structured database like Postgres, an RDF store like Virtuoso, a NoSQL store like Redis)
  We have developed two pipelines as mentioned above, one for testing the deployment with docker swarm and one full pipeline. The first one uses HDFS for storage and the second one uses Elasticsearch and PostGis

- For every mechanism you mentioned, please answer the following questions:
Is the store being managed via the BDE infrastructure?

- Yes the storage is managed using the BDE components, both for HDFS and for Elasticsearch

If no, why not?

If yes:

- How much effort/time did it take to setup and understand the tool/mechanism?
  
  The components used in the pilot architecture have dependencies and the setup of a pilot even in a simple configuration need a good knowledge of docker and of the Big Data framework used. It is not a simple task to set up a pipeline. The BDE infrastructure helps in maintaining and deploying new versions of the components thanks to the loose coupling between them.

- Was the file system able to store and retrieve the files that you need for running the pilot?
  
  We have used Hadoop HDFS successfully with the Hue browser. The data files could be moved from a local file system into Hadoop from the browser.

- Did you experience any problems related to fault tolerance, scalability, reliability and availability?
● We are testing the performances of the pilot based on docker swarm for fault tolerance
  ▪ Was the upload time satisfactory?
  ▪ The upload time is satisfactory but we have still to measure the time needed for the different steps of the pilot’s pipeline
  ▪ Any other points you like to mention related to the Data Store?
    ● We have used Elasticsearch as main data storage system as it is fast and we did not plan to make joins between the data records. Elasticsearch provides also visualization tools to represent geographical data and time series data.

Resource Manager

The platform should be able to provide resource management capabilities and support schedulers for high utilization and throughput.

This set of properties is delivered by Docker Swarm which offers optimal resource management for distributed applications.
Evaluation questions:

- How many applications do you run in parallel during your pilot?
  
  o We run one distributed application using Apache Flink to map match the FCD records and to compute the aggregations. We have tested the application with a parallelization factor 2, that is, two instances in parallel. Each instance is implemented as a docker container that contains the jar file to be executed and is connected to a third container, the Flink master.

- Did you need an automatic resource manager to delegate the resources for distributed applications?
  
  o The resources for computation, message passing and storage are all provided as docker containers managed using the BDE infrastructure and docker swarm

- If yes, did you use Docker Swarm?
  
  ▪ We are testing the pilot in a docker swarm. We had to face many issues related to the complexity of deploying an architecture in a distributed setting and on the quick change of the technology (Docker)
  
  o If no, what did you choose, and why not Docker Swarm?
O If yes
  ▪ Is it easy to set up for your pilot requirements?
    ● As mentioned before is not easy to set up the pilot because it relies on many components that must be configured to work together in a distributed setting. Once the pilot is set up it is easy to replace a component because they are loosely coupled and the configuration of each component is mostly kept inside the container.
  ▪ Which type of interface did you use to interact with Docker Swarm (e.g. the HTTP REST API, a Web interface, the command line shell).
    ● We have been using the command line shell
  O storing and retrieving files?
    ▪ How did you describe the ‘tasks’ and was the ‘matchmaking’ by Docker Swarm finding the appropriate resources satisfactory?
    - Any other remarks?

Scheduler
The scheduler needs to schedule the distributed tasks and offer resources to increase the throughput of overall system.

Two schedulers Marathon and Chronos have been selected for task scheduling in the framework for the first release of the platform. Since Mesos is not part of the BDE platform anymore, also both schedulers are being deprecated. Instead now the INIT_DEAMON is being used.

Evaluation questions:
Many system administrators heard of Cron(jobs) that at certain time intervals or other conditions manage starting and stopping processes.

- Since Marathon and Chronos are not being deprecated, which schedulers are now being considered and even integrated?
- Is there need for schedulers? If yes, where and when?
- What are your experiences with INIT_DEAMON?
- What are your experiences with Docker Healthchecks?

The pilot uses the BDE Init Daemon for the orchestration of its services. The BDE init daemon provides also an UI to define the dependencies among the different pilot’s components and another UI to monitor the status of the components that can be started automatically and start those that need a manual intervention (e.g.
copying data sets from the local file system to the distributed file system used by the pilot). Docker healthchecks have not been used.

Coordination

*The platform requires an efficient system for managed state, distributed coordination, consensus and lock management in the distributed platform.*

ZooKeeper is used as a decentralized tolerant coordination framework for the Apache Kafka servers that are used to manage the communication channels (topics) between the data producers and the data consumers.

Evaluation questions:

The most important feature of ZooKeeper is to make sure the processes keep running and communicating. It responds to node crashes by for example delegating the job to another node.

- Does your pilot contain a set of continuously running applications that interact with each other?
The pilot is based on many distributed components for data ingestion, processing and storage. The components can act as data sources, data producers and consumers and data sink. The communication between these components is handled by the distributed framework itself (e.g. Apache Flink, Elasticsearch) or an external framework such as Zookeeper for Apache Kafka.

- If yes, did you choose for a coordination framework to keep the processes stable?
  
  - The pilot uses Apache Kafka topics to connect a producer, that ingests and transforms the data in a suitable format, and a consumer that computes a function on the data. Each topic can be distributed on different containers and is managed by a broker within a container. The Kafka brokers use Zookeeper for coordination.

- If yes,
  
  - which framework did you choose, and was it doing what you expected?
    
    - Our pilot is based on Apache Kafka that depends on Apache Zookeeper for the coordination of the servers (brokers)

  - Did you experience any problems like physical node failures or memory leaks that resulted in the coordinator to interfere?
    
    - We did not experience any issues in using Apache Zookeeper and Apache Kafka

  - If you chose something else than Zookeeper, please explain why.
If you chose for Zookeeper, what were your experiences?

- Zookeeper is easy to run and configure. It is also well integrated with Apache Kafka

Data Acquisition and Message Passing

The pilot consumes near real-time and historical FCD data. The near real-time data, provided via a web service, is ingested by an Apache Flink job that fetches the data from the web service and sends it to an Apache Kafka topic in a binary format for further processing. The historical data is made available via FTP and can be stored in a local file system or on HDFS.

Evaluation questions:
Most likely some of the data you use in your first pilot you already had available in one form or another. The ‘data acquisition’ process is the bridge between the BDE storage mechanism and the data.

- Is your data ‘special’ in order that it needs some tool to transform or process the data so that it can be stored in the respective data-store (for example streaming data, XML, data that needs to be transformed to JSON, data that needs to be migrated between DBS, etc)?
The data is ‘special’ in the sense that it is a time series, each record contains a timestamp of the time in which the record has been produced. Since the pilot must be able to handle different sources in parallel, records that have been produced at the same time can arrive late because of network latency. Apache Flink can manage late arrivals in computing the data in time windows within a specific threshold. Since the data is received by a Flink job from a Kafka topic its structure must be known in advance in order for Flink to know which is the timestamp and process the records correctly at event time.

- If yes, which tools did you use for this pilot?
  - The pilot uses Apache Kafka to set up pipelines for data transformation and computation tasks

- What are your experiences with Apache Kafka?
  - Apache Kafka is a scalable and fast message passing framework. It provides connectors to other systems such as databases or processing frameworks.

- Are any of these tools part of the generic BDE infrastructure?
  - Yes Apache Kafka is supported by the BDE infrastructure and the related BDE container is used in the pilot
  - If no, would you recommend it to add it?
  - If yes,
was it difficult to learn and setup the tool?

- It’s quite easy to setup Apache Kafka and sending messages to a topic. Its usage for best performances in a distributed setting with consumers working in parallel requires a good understanding of the technology

- What is the weakest link in the pipeline (e.g. the store, the network, the tooling), and was it still performing well enough to be satisfactory for this pilot?

- The pipeline is quite robust once it is setup and used properly

Data Processing

Data Processing Frameworks: The platform requires different frameworks for various SC instances. Each framework has a different set of strengths and is applicable for a specific set of properties of the underlying data.

A multitude of tools are available for the type of processing to be performed on the underlying data, this includes, but is not limited to MapReduce for batch processing,
Spark GraphX for iterative processing, Apache Spark and Apache Flink for data stream and real time processing.

Evaluation questions:

- Which tools do you use to transform your (raw) data in order to, for example, perform analysis, filter noise, make it suitable for sorting and querying etc?
  - The pipeline is based on Apache Flink for the computing aggregations on tabular data and on R scripts and SQL procedures for map matching

- Which of these tools are being supported by the generic BDE infrastructure? And did you use them?
  - Apache Flink is supported by the BDE generic infrastructure while PostGis and the R server have been created specifically for the pilot

- Did the tools you use, and which are supported by the generic infrastructure, do what you expected? Were they easy to use? Please give a summary about your experiences for each data transformation tool.
  - Apache Flink works fine with the data streams. Its capacities are well beyond what we need for our pilot but the aim of the pilot was to be extended to connect to more data sources with higher throughput.
Evaluation questions based on the requirements specified in D5.4:

Table 4: Requirements of the Second SC4 Pilot

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>The pilot will integrate the R scripts used for the map matching of the FCD records with the Apache Flink application. Please describe the reason for the usage of the R scripts and how the integration with the Java code used to implement the Apache Flink application has been made. The R scripts have been used to implement the map matching algorithm. The algorithm has been implemented by domain experts at CERTH that have little knowledge of Java or distributed computing.</td>
</tr>
<tr>
<td>R2</td>
<td>The pilot will use Postgis to store the Thessaloniki road network used in the map matching algorithm. Please describe how the application flow will work in a distributed setting. Postgis is being used because it supports topological queries that are used by the R script for the map matching algorithm. A docker container with Postgis, the road network data and the R server with the scripts is deployed in all the nodes that are used by the pilot. The Flink application (worker) sends the</td>
</tr>
<tr>
<td>R3</td>
<td>The pilot will acquire, transform, and aggregate the near real-time FCD data. The result will be stored in a distributed database (Elasticsearch).</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>FCD data to the R script through a Java to R interface and receives the map matched records</td>
</tr>
<tr>
<td></td>
<td>What kind of transformations are applied to the data? Please describe the algorithm that uses the FCD data to determine the current traffic condition.</td>
</tr>
<tr>
<td></td>
<td>The original data is provided as JSON records. These records are transformed into a binary model that is known by producers and consumers of the data. The structure of the binary format (Avro) is easy to modify and consumes less space than the original format. A Flink distributed application (consumer) reads the data as soon it is sent to its topic. The application then sends the data to the R server and Postgis for the map matching and computes the aggregations (traffic flow and average speed).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R4</th>
<th>The pilot will acquire, transform and aggregate the historical FCD data. The result will be stored in a distributed database (Elasticsearch).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What kind of transformations are applied to the data? Where are the source historical data sets stored and where is the result of the processing saved?</td>
</tr>
<tr>
<td></td>
<td>The historical FCD data is used to train a machine learning algorithm for the prediction of the traffic flow. The FCD historical records</td>
</tr>
</tbody>
</table>
must be map matched to the near real-time data in order to be used as example for the machine learning algorithm. The map-matched records are saved in HDFS or in Elasticsearch.

| R5 | The processed data, flow and average speed, will be visualized using Kibana. | What kind of visualization will be available? | The data can be visualized on a map where the location of the vehicles or the flow in each road segment can be shown. |
| R6 | The pilot can be started in two configurations using a docker-compose file: single node (for development and testing) and cluster (production) | Were you able to run all the pilot components in one single node for development and testing purposes? | We have been able to run the pilot in a single node using a docker-compose file. We have still to fix some problems in order to be able to run the pilot in a swarm in a stable and reproducible way. |

Table 7: Requirements of the Second SC4 Pilot

4.5 SC5: Climate Action, Environment, Resource Efficiency and Raw Materials

The pilot is carried out by NCSR-D in the frame of SC5 Climate Action, Environment, Resource
Efficiency and Raw Materials.

The pilot demonstrates the following workflow: A (potentially hazardous) substance is released in the atmosphere that results in increased readings in one or more monitoring stations. The user accesses a user interface provided by the pilot to define the locations of the monitoring stations as well as a timeseries of the measured values (e.g. gamma dose rate). The platform initiates

- a weather matching algorithm, that is a search for similarity of the current weather and the pre-computed weather patterns, as well as
- a dispersion matching algorithm, that is a search for similarity of the current substance dispersion patterns with the precomputed ones.

The weather patterns have been extracted in a pre-processing step by clustering weather conditions recorded in the past, while the substance dispersion patterns have been precomputed by simulating different scenarios of substance release and weather conditions. The pre-computed patterns are stored in the BDE infrastructure and retrieved upon request.

The following datasets are involved:

- NetCDF files from the European Centre for Medium range Weather Forecasting (ECMWF⁴)
- GRIB files from National Oceanic and Atmospheric Administration (NOAA⁵)

The following processing will be carried out:

- The weather clustering algorithm that creates clusters of similar weather conditions implemented using the BDI platform (see Section 6.3).
- The WRF downscaling that takes as input a low resolution weather and creates a high resolution weather.

⁴ [http://apps.ecmwf.int/datasets](http://apps.ecmwf.int/datasets)
The NOAA HYSPLIT\textsuperscript{6} atmospheric dispersion model computes dispersion patterns given predominant weather conditions. The following outputs are made available for visualization or further processing:

- The dispersions produced by HYSPLIT.
- The weather clusters produced by the weather clustering algorithm.

**BDE infrastructure questionnaire**

**File System**

*The platform requires a distributed file system which provide storage, fault tolerance, scalability, reliability, and availability to the multitude of SC partners.*

This has resulted in selection of the Apache Hadoop Distributed File system, HDFS. This allows for elasticity in the need for new storage nodes, while it is fast enough for the purposes of the pilot.

**Evaluation questions:**

\footnote{http://ready.arl.noaa.gov/HYSPLIT.php}
- **How much data did you store?**

  2 years worth of weather files (netCDF format) divided in 3 days periods (19 GB). We provide only 2 years worth of data for testing purposes that could be later scaled to represent current weather.

  Cluster descriptor dispersions in netCDF form for 2 types of configurations (2 GB), each configuration has 15 descriptors for 20 stations.

  **Cells & EU populated locations points**: 290246 triples 65M
  
  **EU populated location metadata**: 765387 triples 84M
  
  **EU Hospital metadata**: 22M 33488 rows
  
  **Cells & hospital points**: 284489 triples 64M

- **How much data did you process?**

  13 years worth of weather reanalysis data in the GRIB2\(^7\) format (2.5 TB)

  - Openstreetmap dataset from where we extracted hospitals metadata: 20G in pbf compressed format

---

\(^7\) [https://rda.ucar.edu/docs/formats/grib2/grib2doc/](https://rda.ucar.edu/docs/formats/grib2/grib2doc/)
- Geonames dataset from where we extracted geonames metadata: 15G

- Please specify which storage mechanism(s)/tool(s) you used (e.g. HDFS, a structured database like Postgres, an RDF store like Virtuoso, a NoSQL store like Redis)

  Apache Hadoop: Manageable via the BDE infrastructure, which made it effortless to setup and get familiar with, storing and retrieving of vital files was successfully achieved, upload time and response time were satisfactory.

  PostgreSQL: Manageable via the BDE infrastructure, which made it effortless to setup and get familiar with, querying and retrieving of catalogues was successful and quick, the upload time was satisfactory even for some cases of BLOB files. We also used the PostGIS extension as a backend for Strabon.

  Cassandra: We used Cassandra to store metadata about hospitals located in Europe. The metadata where access with their IDs so we needed a fast Key-Value store to retrieve them fast.

  Virtuoso: To store semantic metadata for populated places located in Europe extracted from the Geonames dataset. Mpla mpla.

  Stabon: To store geographical information about the dispersion cloud and geographic information about populated places or hospitals in Europe. With
Stabon we could determine which points of interest are included in the dispersion cloud.

Semagrow: SPARQL endpoint over Virtuoso, Cassandra and Strabon.

- For every mechanism you mentioned, please answer the following questions:
  - Is the store being managed via the BDE infrastructure?
  
  The stores above, as well as the rest of the components employed in the pilot, have been deployed and used via the BDI platform deployed at NCSRD.
  
  - If no, why not?
  - If yes:
    
    • How much effort/time did it take to setup and understand the tool/machinery?
    • Was the file system able to store and retrieve the files that you need for running the pilot?
    • Did you experience any problems related to fault tolerance, scalability, reliability and availability?
    • Was the upload time satisfactory?
    • Any other points you like to mention related to the Data Store?
Resource Manager

_The platform should be able to provide resource management capabilities and support schedulers for high utilization and throughput._

This set of properties is delivered by Docker Swarm which offers optimal resource management for distributed applications.

**Evaluation questions:**

- How many applications do you run in parallel during your pilot?

  **The SC5 Pilot has 6 applications running in parallel in order to function properly:** Apache Hadoop, Postgres/PostGIS, Sextant, Cassandra, Semagrow and Virtuoso.

- Did you need an automatic resource manager to delegate the resources for distributed applications?

  **There is a need for managing resources and for that reason we used Docker Swarm. The setup for our requirements was easy, interaction was achieved**
through command line shell.

- If yes, did you use Docker Swarm?
  - If no, what did you choose, and why not Docker Swarm?
  - If yes
    - Is it easy to set up for your pilot requirements?
    - Which type of interface did you use to interact with Docker Swarm (e.g. the HTTP REST API, a Web interface, the command line shell).
    - Storing and retrieving files?
      - How did you describe the ‘tasks’ and was the ‘matchmaking’ by Docker Swarm finding the appropriate resources satisfactory?
- Any other remarks?

Scheduler

*The scheduler needs to schedule the distributed tasks and offer resources to increase the throughput of overall system.*

Two schedulers Marathon and Chronos have been selected for task scheduling in the framework for the first release of the platform. Since Mesos is not part of the BDE
platform anymore, also both schedulers are being deprecated. Instead now the
INIT_DAEMON and Docker Healthchecks are being used.

Evaluation questions:
Many system administrators heard of Cron(jobs) that at certain time intervals or other
conditions manage starting and stopping processes.

- Since Marathon and Chronos are being deprecated, which schedulers are now
  being considered and even integrated?

  **The SC5 pilot uses INIT_DAEMON for scheduling purposes.**

- Is there need for schedulers? If yes, where and when?

  **Yes, in order for the SC5 pilot to function as well as to offer an overall
  better user experience to the domain experts we need to ensure the
  availability of the services. For that reason, we are using schedulers on vital
  processes.**

- What are your experiences with INIT_DAEMON?

  **INIT_DAEMON made scheduling of vital processes easy and it was a
  perfect fit in our case. INIT_DAEMON ensured availability with minimum**
effort to setup and to get familiar with.

- What are your experiences with Docker Healthchecks?

Coordination

*The platform requires an efficient system for managed state, distributed coordination, consensus and lock management in the distributed platform.*

ZooKeeper will be used as a decentralized tolerant coordination framework.

Evaluation questions:

The most important feature of ZooKeeper is to make sure the processes keep running and communicating. It responds to node crashes by for example delegating the job to another node.

- Does your pilot contain a set of continuously running applications that interact with each other?

  Yes, in order to keep the applications communication stable, we must ensure
the availability of their services. We used YARN as a resource manager for Apache Hadoop (to ensure scheduling and coordination) and INIT_DAEMON tools for PostgresSQL (initctl) and Sextant (upstart). YARN and initctl for PostgresSQL are available through the BDI.

- If yes, did you choose for a coordination framework to keep the processes stable?
  - If yes,
    o which framework did you choose, and was it doing what you expected?
    o Did you experience any problems like physical node failures or memory leaks that resulted in the coordinator to interfere?
    o If you chose something else than Zookeeper, please explain why.
    o If you chose for Zookeeper, what were your experiences?

Data Acquisition

Owing to the wide range of input data properties, a set of tools is needed to support the process of gathering, filtering and cleaning data before it is put in a data warehouse or any other storage solution on which data processing can be carried out during data acquisition.
The set of frameworks including Apache Flume and Apache Kafka have been chosen with an ambition that it would cater for the all the four properties of Big Data.

*Apache Flume*: A framework to populate HDFS with streaming event data.

*Apache Kafka*: A framework to transfer reliable messaging between applications.

**Evaluation questions:**

Most likely some of the data you use in your first pilot you already had available in one form or another. The ‘data acquisition’ process is the bridge between the BDE storage mechanism and the data.

- Is your data ‘special’ in order that it needs some tool to transform or process the data so that it can be stored in the respective data-store (for example streaming data, XML, data that needs to be transformed to JSON, data that needs to be migrated between DBS, etc)?

The SC5 Pilot makes use of weather data files that are usually in the netCDF/GRIB format. Although they are ‘special’, meaning that there is a need to perform transformations and processing in order to extract more valuable information, they do not fall under well known and general format such as XML/JSON and therefore we did not use any of Apache Kafka/Flume. The Geonames dataset is in RDF format, the Openstreetmap
dataset is in PBF format.

- If yes, which tools did you use for this pilot?
- What are your experiences with Apache Kafka?
- What are your experiences with Apache Flume?
- Are any of these tools part of the generic BDE infrastructure?
  - If no, would you recommend it to add it?
  - If yes,
    - was it difficult to learn and setup the tool?
    - What is the weakest link in the pipeline (e.g. the store, the network, the tooling), and was it still performing well enough to be satisfactory for this pilot?

Data Processing

Data Processing Frameworks: The platform requires different frameworks for various SC instances. Each framework has a different set of strengths and is applicable for a specific set of properties of the underlying data.
A multitude of tools are available for the type of processing to be performed on the underlying data, this includes, but is not limited to MapReduce for batch processing, Spark GraphX for iterative processing, Apache Spark and Apache Flink for data stream and real time processing.

**Evaluation questions:**

- Which tools do you use to transform your (raw) data in order to, for example, perform analysis, filter noise, make it suitable for sorting and querying etc?

  Due to the structure of weather files as well as the input guides of the dispersion models (HYSPLIT, DIPCOT) the tools for transforming the raw data into more suitable versions was the WRF model along with its preprocessor WPS. Docker versions of these tools are available but are not supported by the BDE infrastructure. GeoNames and OpenStreetMap data were pre-processed via scripts.

- Which of these tools are being supported by the generic BDE infrastructure? And did you use them?
- Did the tools you use, and which are supported by the generic infrastructure, do what you expected? Were they easy to use? Please give a summary about your experiences for each data transformation tool.

Evaluation questions based on the requirements specified in D5.4:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
</table>
| R1 Provide a means of downloading current/evaluation weather from ECMWF or alternative services | Please describe the functionality provided by the Data connector/interface developed for this pilot. What are your experiences?  
ECMWF’s data access service is primarily an interactive one. An API is also provided, however it was not practical to use for the purposes of this pilot. This did not constitute a problem, as the pilot is based on significant pre-processing, which the raw data are also
| R2 | ECMWF and NOAA datasets are compatible with the WRF and DIPCOT naming conventions | Was the preprocessing WPS normalization step able to perform the necessary transformations and variable renamings needed to ensure compatibility?  
   The ECMWF dataset (as well as its NOAA derivatives) are compatible with the WRF model by design. This is due to their significance in the weather and climate modelling community as well as due to WRF's popularity. The atmospheric dispersion model HYSPLIT is designed to work on input data adhering to the WRF schema. |
| R3 | Retrieve NetCDF files from HDFS as input to the weather clustering algorithm | How did you retrieve the NetCDF files?  
   For retrieving NetCDF files, both for clustering as well as for visualisation (e.g. atmospheric dispersions), we use appropriate catalogues stored in Postgres (also provided by the BDI). Querying these files is based on date information (pattern matching), name of estimated source and clustering configuration (visualisation). |
| R4 | Dispersion matching will filter on dispersion values | Was the relational database used for this pilot, able to provide indexes on dispersion values for |
Due to the relatively large number of dispersion cells that needed to be matched, this approach in practice appeared to be inefficient and was abandoned during the implementation of the pilot. At the same time, as matching dispersions to readings is based on vector-based measurements (e.g. the cosine similarity), this choice did not lead to loss in efficiency.

Was the weather and dispersion matching producing output compatible with Sextant's input or was it needed to modify Sextant in order support the new input? Please elaborate.

As sextant is able to visualise geospatial information in various formats, the data produced were immediately usable. We modified Sextant only in order to support the interaction requirements of the pilot, i.e. to add specialised input controls, etc.
4.6 SC6: Europe in a changing world - inclusive, innovative and reflective societies.

The pilot is carried out by NCSR-D, and SWC in the frame of SC6 Europe in a changing world - inclusive, innovative and reflective societies.

The pilot demonstrates the following workflow. Municipality economic data (i.e., budget and budget execution data) are ingested at a regular basis (daily, weekly, monthly and so on) from a series of locations in a variety of structures and formats are homogenized so that they can be compared, analyzed and visualized in a comprehensible way. On top of this harvested and normalised data financial ratios (https://en.wikipedia.org/wiki/Financial_ratio) are calculated that help users to better understand the data as well as to easier compare the financial situation of the municipalities (by comparing financial ratios). The data (the financial ratios) is exposed to users via a visualisation dashboard that exposes search/discovery, aggregation, analysis, correlation, and visualization functionalities over structured data. The results of the data analysis is being stored in the BDE/BDI infrastructure to avoid carrying out the same processing multiple times.

The second cycle of the pilot has extended the first pilot by incorporating different formats by developing a modular parsing library as well as by updating the Pilot to the latest Big Data Integrator (BDI) instance / components and by making the Pilot more stable and performant.

The following datasets are involved:

- Budget execution data of Municipality of Athens.
- Budget execution data of Municipality of Thessaloniki
- Budget execution data of Municipality of Kalamaria

The current datasets involved are exposed either as an API or as CSV, XML files.
Datasets are described by DCAT-AP\textsuperscript{8} metadata and - where possible - by the FIBO\textsuperscript{9} and FIGI\textsuperscript{10} ontologies, AND consistently by the accounting scheme of the data (means mapping between the accounting systems / schemes of the municipalities). Statistical data is being described in the RDF DataCube\textsuperscript{11} vocabulary.

The following processing is carried out:

- Data ingestion and homogenization, data mapping
- Aggregation, analysis, correlation over scientific data

The following outputs are made available for visualization or further processing:

- Structured information extracted from budget datasets exposed as a SPARQL endpoint (link see below)
- Metadata for dataset searching and discovery
- Aggregation and analysis
- Financial ratio calculation including explanation per ratio (and formula for ratio) in the form of chart visualisations, that can be exported to be used further

**Relevant / important links for the SC6 Pilot:**

BDE SC6 Pilot - analytics dashboard of financial ratios of municipalities (publicly available):
https://bde.poolparty.biz/GraphSearchSC6/

SPARQL endpoint for the Pilot data (publicly available): https://bde-virtuoso.poolparty.biz/sparql

\textsuperscript{8} Cf. https://joinup.ec.europa.eu/asset/dcat_application_profile/description
\textsuperscript{9} Cf. http://www.omg.org/spec/EDMC-FIBO/FND/1.0/Beta1/index.htm
\textsuperscript{11} Cf. https://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/
BDE infrastructure questionnaire

File System

*The platform requires a distributed file system which provide storage, fault tolerance, scalability, reliability, and availability to the multitude of SC partners.*

This has resulted in selection of Apache Hadoop Distributed File system, HDFS.

**Evaluation questions:**

- How much data did you store?
  - ~30.000.000 RDF Triples
- How much data did you process?
  - 247 Mb of CSV data
- Please specify which storage mechanism(s)/tool(s) you used (e.g. HDFS, a structured database like Postgres, an RDF store like Virtuoso, a NoSQL store like Redis)
- An RDF Store (Virtuoso)
- HDFS (for debugging)

For every mechanism you mentioned, please answer the following questions:

- Is the store being managed via the BDE infrastructure?
  - No (BUT we are using BDE/BDI components at a SWC infrastructure - see given reason below)

- If no, why not?
  - This Pilot is developing a analytics dashboard (based on PoolParty GraphSearch), displaying calculated financial ratios, that requires a persistent datastore. Thereby we are using an additional infrastructure at SWC, making use of BDE / BDI components for continuous development (as this infrastructure needs to be available 24/7). In a production deployment the store would be managed by the BDE Infrastructure.

- If yes:
  - How much effort/time did it take to setup and understand the tool/mechanism?
    - We have invested around 1 pm altogether in setup, although this is an ongoing process to stay up to date.
  - Was the file system able to store and retrieve the files that you need for running the pilot?
Yes. The usage of Apache Flume allows for multiple file systems by default. The pilot was tested using HDFS and the default Linux file-system given on the test servers.

- Did you experience any problems related to fault tolerance, scalability, reliability and availability?
  - No. We could easily determine that the designed architecture is scalable. The tools that form the core of SC6's architecture didn't expose any reliability problems within the scope of the pilot.

- Was the upload time satisfactory?
  - Upload had been done using "scp" (secure copy) into the Linux file system of the test servers. Insertion of files into HDFS from a local file system went without troubles.

- Any other points you like to mention related to the Data Store?
  - No

Resource Manager

*The platform should be able to provide resource management capabilities and support schedulers for high utilization and throughput.*
This set of properties is delivered by Docker Swarm which offers optimal resource management for distributed applications.

**Evaluation questions:**

- How many applications do you run in parallel during your pilot?
  - The total number is 8: Poolparty, GraphSearch, CSV->RDF converter, Financial Ratios Calculator, Apache Flume, Apache Kafka, Apache Spark, Virtuoso Triple Store

- Did you need an automatic resource manager to delegate the resources for distributed applications?
  - Yes

- If yes, did you use Docker Swarm?
  - If no, what did you choose, and why not Docker Swarm?
    - Yes, the Pilot is using Docker Swarm. As well as Apache Mesos with Marathon
  - If yes
    - Is it easy to set up for your pilot requirements?
Yes. Older versions of the BDE Infrastructure also include installation of Mesos/Marathon

- Which type of interface did you use to interact with Docker Swarm (e.g. the HTTP REST API, a Web interface, the command line shell).
  - Command line
    - Storing and retrieving files?
      - How did you describe the ‘tasks’ and was the ‘matchmaking’ by Docker Swarm finding the appropriate resources satisfactory?
        - Yes it was.

- Any other remarks?
  - No

Scheduler

The scheduler needs to schedule the distributed tasks and offer resources to increase the throughput of overall system.
Two schedulers Marathon and Chronos have been selected for task scheduling in the framework for the release of the platform. Since Mesos is not part of the BDE platform anymore, also both schedulers are being deprecated. Instead now the INIT_DEAMON and Docker Health Checks are being used.

Evaluation questions:

Many system administrators heard of Cron(jobs) that at certain time intervals or other conditions manage starting and stopping processes.

- Since Marathon and Chronos are not being deprecated, which schedulers are now being considered and even integrated?
  - We tested the whole pilot on Docker Swarm and Mesos/Marathon to our full satisfaction.

- Is there need for schedulers? If yes, where and when?
  - No

- What are your experiences with INIT_DEAMON?
  - This was not really required by our Pilot, since the components took care of this within the scope of the application as well as the startup sequence within docker swarm was enough. For example Apache Flume created Apache Kafka topics and wouldn't startup if there was an error. On the other hand it was made sure within the docker swarm snippet that Apache Kafka started before Apache Flume.
What are your experiences with Docker Health-checks?
   ○ N/A

Coordination

*The platform requires an efficient system for managed state, distributed coordination, consensus and lock management in the distributed platform.*

ZooKeeper will be used as a decentralized tolerant coordination framework.

**Evaluation questions:**

The most important feature of ZooKeeper is to make sure the processes keep running and communicating. It responds to node crashes by for example delegating the job to another node.

- Does your pilot contain a set of continuously running applications that interact with each other?
  ○ Yes
If yes, did you choose for a coordination framework to keep the processes stable?

- Yes

- If yes, which framework did you choose, and was it doing what you expected?
  - Docker Swarm and (before) Mesos/Marathon
  - No

- Did you experience any problems like physical node failures or memory leaks that resulted in the coordinator to interfere?
  - No

- If you chose something else than Zookeeper, please explain why.
  - N/A

- If you chose for Zookeeper, what were your experiences?
  - Zookeeper was running stable and is required by most components that form part of SC6's architecture.

Data Acquisition

Owing to the wide range of input data properties, a set of tools is needed to support the process of gathering, filtering and cleaning data before it is put in a data warehouse or
any other storage solution on which data processing can be carried out during data acquisition

The set of frameworks including Apache Flume and Apache Kafka have been chosen with an ambition that it would cater for all the four properties of Big Data.

   * **Apache Flume**: A framework to populate HDFS with streaming event data.
   * **Apache Kafka**: A framework to transfer reliable messaging between applications.

**Evaluation questions:**
Most likely some of the data you use in your first pilot you already had available in one form or another. The ‘data acquisition’ process is the bridge between the BDE storage mechanism and the data.

- Is your data ‘special’ in order that it needs some tool to transform or process the data so that it can be stored in the respective data-store (for example streaming data, XML, data that needs to be transformed to JSON, data that needs to be migrated between DBS, etc)?
  - CSV data that has to be transformed to RDF triples; but also schema mapping is required to map the incoming data to one schema
- If yes, which tools did you use for this pilot?
- Custom made parsers and converters, interacting with Poolparty Semantic Suite.

- What are your experiences with Apache Kafka?

  - Works stable and as expected. Interaction with Apache Spark works without much troubles. The only thing that is to be mentioned here is versioning. This could be improved especially if Apache Kafka is used in connection with Apache Spark. This however is outside the scope of the BDE project.

- What are your experiences with Apache Flume?

  - Works stable and as expected. We've extended Apache Flume's HDFS source to include parent directory headers due to the nature of our pilot. Files need to be stored in structured subdirectories and shouldn't be stored in one single large directory. This extension worked without issues.

- Are any of these tools part of the generic BDE infrastructure?

  - Both Apache Flume and Apache Kafka.
  - If no, would you recommend it to add it?
  - If yes,
    - was it difficult to learn and setup the tool?
      - In general those tools and their application need to be understood. It takes some time but not difficult. Training for users of the platform is necessary.
What is the weakest link in the pipeline (e.g. the store, the network, the tooling), and was it still performing well enough to be satisfactory for this pilot?

- We didn't have experience with especially weak points. It is understood that all components need to work seamlessly in a productive environment.

Data Processing

*Data Processing Frameworks: The platform requires different frameworks for various SC instances. Each framework has a different set of strengths and is applicable for a specific set of properties of the underlying data.*

A multitude of tools are available for the type of processing to be performed on the underlying data, this includes, but is not limited to MapReduce for batch processing, Spark GraphX for iterative processing, Apache Spark and Apache Flink for data stream and real time processing.

**Evaluation questions:**
- Which tools do you use to transform your (raw) data in order to, for example, perform analysis, filter noise, make it suitable for sorting and querying etc?
  - Custom made parsers and converters, interacting with Poolparty Semantic Suite.

- Which of these tools are being supported by the generic BDE infrastructure? And did you use them?
  - Above mentioned parsers and converters are wrapped inside Apache Spark Jobs so they are supported by the BDE infrastructure.

- Did the tools you use, and which are supported by the generic infrastructure, do what you expected? Were they easy to use? Please give a summary about your experiences for each data transformation tool.
  - We created a general Big Data Pipeline for data processing that can be used for many use cases with minor adaptations. HDFS/Linux file-system worked fine with Apache Flume, Apache Kafka and Apache Spark.
Evaluation questions based on the requirements specified in D5.4:

Table 6: Requirements of the Second SC6 Pilot

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>In case of failures during data processing, users should be able to recover data produced earlier in the workflow along with their lineage and associated metadata</td>
</tr>
<tr>
<td></td>
<td>Please describe the workflow and the technology used for the processing modules that periodically store intermediate results. Do these modules at start-up When starting up, processing modules check at the metadata registry if intermediate results are available? If yes, how?</td>
</tr>
<tr>
<td><strong>ANSWER</strong></td>
<td>Data is being continually ingested through Apache Flume. Apache Kafka takes this data and feeds it to the Spark job. When the Spark job fails for any reason, Kafka does not receive a delivery notification and thus the data remain in waiting until the Spark job is further available. Furthermore, Spark provides mechanisms to recover from failure of processing nodes.</td>
</tr>
</tbody>
</table>
**R2**

Transform budget data into a homogenized format using various parsers.

Which sources are being parsed?

Currently three sources are being used: financial execution (including budget data) data from Thessaloniki (CSV), Kalamaria (CSV) and Athens (RDF/XML).

The data is being parsed with home-built parsers. The parsers have shown no issue with the data available so far. It must be mentioned, however, that the parsing involves matching strings in the data to those in a fixed vocabulary, so stability depends on this vocabulary being followed. Since all parsers conform with a single interface, usability is very good. Extensibility requires some work in order to easily configure the parsers for different files.

Which software provides the SPARQL endpoint?

Currently, these sources are being used: Protégé, R2RML, XSLT and TBoxvocabulary (CSV) and the TBoxphishing (CSV) description (including budget data) from Athens.

**R3**

Expose data and metadata through a SPARQL endpoint.

- Please elaborate on the data and metadata that is being stored at this moment (e.g. number of triples, which schema(s) used and which queries executed).
<table>
<thead>
<tr>
<th>R4</th>
<th>Intuitive, easy-to-use, interface for searching and selecting relevant data sources. The use of the user interface should be documented so that users can ease into using it with as little effort as possible.</th>
</tr>
</thead>
</table>

**ANSWER**

The SPARQL endpoint is provided by an instance of Virtuoso Universal Server (a triple store). There are two types of data being stored. The first is incoming the budget-execution data that is converted to RDF data (~30 Mio RDF triples at the moment of creation of this deliverable). The second is the financial ratios derived from this data (~10k RDF triples so far). The queries performed on these graphs are those used by PoolParty GraphSearch (the user interface) for data visualization.

<table>
<thead>
<tr>
<th>R4</th>
<th>Please describe the GraphSearch UI. What is the feedback of the users:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Which questions can be answered via the interface?</td>
</tr>
<tr>
<td></td>
<td>- Was it easy to use?</td>
</tr>
<tr>
<td></td>
<td>- What can be improved?</td>
</tr>
<tr>
<td></td>
<td>- Was the average and maximum response time acceptable?</td>
</tr>
</tbody>
</table>

**ANSWER**

The PoolParty GraphSearch UI allows the user to search the set of financial ratios that have been computed (via faceted search mechanism). The user can choose several such ratios and see
how the different municipalities compare across the chosen ratios. This is shown in the form of bar charts, which can be further downloaded in different image formats (PNG, Jpeg, etc) to be used further (e.g. in presentations or reports).

Typical questions that can be answered with graph search are: “Which municipality has a larger revenue to income ratio?” or “For this municipality, what makes up a greater part of income, regular Income or subsidies?”

The interface is intuitive and conforming to modern UX principles. The response time is not noticeable.

Currently, the filtering of results does not allow for complex search queries to be performed.
There is no output to csv files, but there is an API in the form of a SPARQL endpoint that allows to query the whole data in place.

### 4.7 SC7: Secure societies – Protecting freedom and security of Europe and its citizens

The pilot is carried out by SatCen, UoA, and NCSR-D in the frame of SC7 Secure societies – Protecting freedom and security of Europe and its citizens.
The third and the last phase of the pilot aimed to extend the pilot with new data (from Remote Sensing and Social Sensing sources) and to demonstrate the fusion and integration of satellite data and news. The final goal was to increase the information available for the final users with respect to a specific event and to extend the sources for the cross-validation.

Through a user-friendly interface (Sextant), the user can analyze satellite images to detect areas with changes (Change Detection workflow) on land cover or land use and integrate the output extracting information from social media and news items (Event Detection workflow). In line with the project recommendations, the Change Detection workflow is based on the use of open data, such as the satellite images provided by ESA Sentinel 1 and Sentinel 2 (added in the third phase).

The pilot demonstrates the following workflows:

1. **Event detection workflow**: The Event Detection workflow uses publicly available content from Social Media (e.g. Twitter, and in the third phase Flickr) and news agencies (e.g. Reuters). News sites and social media are monitored, filtered, analyzed and augmented with location and named entity metadata (e.g. well-known people). The Event Detector clusters news and twitter content into events, and associated with user-uploaded Flickr images. Using this information, the end-user is notified about the geographical area representing the subject of the respective news, and can visualize at the same time the information on the event, and a map layer with landcover changes detected by the other workflow (if activated). The keyword-based Twitter API is employed to retrieve tweets based on predefined keywords. To further support the keyword-based search, a full-text indexing engine is included.

2. **Change detection workflow**: The user navigates on a map to the geographical location of interest. The user selects a relevant Area of Interest over which to inspect changes occurring between two specific dates. The images are searched for in the ESA Sentinels Scientific Hub and the earliest and latest images around the selected temporal interval are selected and downloaded. The processing starts automatically once the download is complete. The user can inspect the processing status via a progress bar and is notified upon completion. Finally, the user can display,
as layers on the base map, the two images and the change detection result. The information about
the images acquisition date and time can be viewed via the Layers panel.

The following outputs are made available for visualization or further processing:

- Relevant news related to specific keywords together with the corresponding Area of Interest;
- Detected area with changes.

BDE infrastructure questionnaire

File System

The platform requires a distributed file system which provide storage, fault tolerance,
scalability, reliability, and availability to the multitude of SC partners.

This has resulted in selection of Apache Hadoop Distributed File system, HDFS.

Evaluation questions:

- How much data did you store?
  - Storing data in HDFS is a dynamic procedure. Every time the user activates the change detection service approximately 2 GB are stored in HDFS. The HDFS multiplier factor is 3, so a total of 6 GB are stored in HDFS. During the testing phase and providing access to other BDE members to test the pilot, the change detection service was triggered more or less 50 times, so 300 GB of data were stored in it.
The Event Detector module stores as much data is available from its sources (RSS feeds and twitter monitored accounts) at half-hour intervals, roughly a few hundred items per iteration. For each item, the retrievable metadata (locations, entities, etc) depends on its textual content of each article or social media post.

- How much data did you process?
  - Every time the user requests a change-detection approximately 2 GB of images (2 images) are downloaded and processed.
  - The Event Detector module stores a few KB of data per item crawled.

- Please specify which storage mechanism(s)/tool(s) you used (e.g. HDFS, a structured database like Postgres, an RDF store like Virtuoso, a NoSQL store like Redis)
  - PostGres is used for storing user-accounts for the entry-point of the pilot (Sextant) and all their information included.
  - HDFS is used for storing the satellite images that are processed during change-detection.
  - PostGres with PostGIS added is used as a backend for Strabon.
  - Strabon (a spatiotemporal RDF storage) is used for storing information regarding events and changes.
  - Cassandra is used for storing news articles, social media, location WKT geometries, events and all entity metadata.
For every mechanism you mentioned, please answer the following questions:

- Is the store being managed via the BDE infrastructure?
  - Yes

- If yes:
  - How much effort/time did it take to setup and understand the tool/mechanism?
    - The SC7 pilot is a very large application. We needed about 2 months to run it for the 1st time. During the first month we did all the basic configuration and during the second we did fine tuning to some of the apps to achieve good communication among them.
  - Was the file system able to store and retrieve the files that you need for running the pilot?
    - Yes
  - Did you experience any problems related to fault tolerance, scalability, reliability and availability?
    - The tools that form the SC7 architecture did not show any reliability problems within the scope of the pilot.
  - Was the upload time satisfactory?
    - Yes
Any other points you like to mention related to the Data Store?

- No

Resource Manager

The platform should be able to provide resource management capabilities and support schedulers for high utilization and throughput.

This set of properties is delivered by Docker Swarm which offers optimal resource management for distributed applications.

Evaluation questions:

- How many applications do you run in parallel during your pilot?

- Did you need an automatic resource manager to delegate the resources for distributed applications?
  - Yes

- If yes, did you use Docker Swarm?
- **Yes, we used Docker Swarm for the SC7-Pilot**
  - If no, what did you choose, and why not Docker Swarm?
  - If yes
    - Is it easy to set up for your pilot requirements?
      - **Yes**
    - Which type of interface did you use to interact with Docker Swarm (e.g. the HTTP REST API, a Web interface, the command line shell) and to storing and retrieving?
      - **We used the command line shell.**
    - How did you describe the ‘tasks’ and was the ‘matchmaking’ by Docker Swarm finding the appropriate resources satisfactory?
      - **Yes, the Docker Swarm was able to find the appropriate resources in every case.**
  - Any other remarks?
    - **N/A**

Scheduler
The scheduler needs to schedule the distributed tasks and offer resources to increase the throughput of overall system.

Two schedulers Marathon and Chronos have been selected for task scheduling in the framework for the first release of the platform. Since Mesos is not part of the BDE platform anymore, also both schedulers are being deprecated. Instead now the INIT_DEAMON and Docker Healthchecks are being used.

**Evaluation questions:**

Many system administrators heard of Cron(jobs) that at certain time intervals or other conditions manage starting and stopping processes.

- Since Marathon and Chronos are not being deprecated, which schedulers are now being considered and even integrated?
  
  - **The cron utility.**

- Is there need for schedulers? If yes, where and when?

  - **The Event Detector container used the cron utility to run its components in a periodic manner.**

- What are your experiences with INIT_DEAMON?

  - **It was not really required by our pilot. Most of the components are web services that run run in Tomcat and they take care of the workflow themselves.**
The Event Detector container supported INIT_DAEMON scheduling, but did not require it.

- What are your experiences with Docker Healthchecks?
  - They are straightforward and easy to use. They add some delay in starting the docker-containers, but they work fine. They are embedded in all the components of the SC7 Pilot.

Coordination

*The platform requires an efficient system for managed state, distributed coordination, consensus and lock management in the distributed platform.*

ZooKeeper was used as a decentralized tolerant coordination framework.

**Evaluation questions:**

The most important feature of ZooKeeper is to make sure the processes keep running and communicating. It responds to node crashes by for example delegating the job to another node.

- Does your pilot contain a set of continuously running applications that interact with each other?
Yes
- If yes, did you choose for a coordination framework to keep the processes stable?
  - Yes
- If yes, which framework did you choose, and was it doing what you expected?
  - Docker Swarm
  - Did you experience any problems like physical node failures or memory leaks that resulted in the coordinator to interfere?
    - Yes
  - If you chose something else than Zookeeper, please explain why.
    - N/A
  - If you chose for Zookeeper, what were your experiences?
    - N/A

Data Acquisition

Owing to the wide range of input data properties, a set of tools is needed to support the process of gathering, filtering and cleaning data before it is put in a data warehouse or
any other storage solution on which data processing can be carried out during data acquisition

The set of frameworks including Apache Flume and Apache Kafka have been chosen with an ambition that it would cater for the all the four properties of Big Data.

Apache Flume: A framework to populate HDFS with streaming event data.

Apache Kafka: A framework to transfer reliable messaging between applications.

Evaluation questions:

Most likely some of the data you use in your first pilot you already had available in one form or another. The ‘data acquisition’ process is the bridge between the BDE storage mechanism and the data.

- Is your data ‘special’ in order that it needs some tool to transform or process the data so that it can be stored in the respective data-store (for example streaming data, XML, data that needs to be transformed to JSON, data that needs to be migrated between DBS, etc)?

  o News articles were parsed using regex-based filtering, XML and http parsers and Web crawlers. All content was stored in Cassandra tables.

  o Location geometries are encoded in the WKT format.
Events resulting from news articles, social media messages and metadata are transformed to RDF triples to be stored in Strabon.

Areas with changes found by the Change Detector are stored as RDF in Strabon with the help of GeoTriples. (also are transformed in JSON for visualization through Sextant)

- If yes, which tools did you use for this pilot?
  - Geotriples, JSOUP, JTS, Twitter API, modified components of the NOMAD platform.

- What are your experiences with Apache Kafka?
  - None

- What are your experiences with Apache Flume?
  - None

- Are any of these tools part of the generic BDE infrastructure?
  - If no, would you recommend it to add it?
    - N/A
  - If yes,
    - Was it difficult to learn and setup the tool?
      - No
What is the weakest link in the pipeline (e.g. the store, the network, the tooling), and was it still performing well enough to be satisfactory for this pilot?

- The slowness of the network for downloading satellite images.

Data Processing

Data Processing Frameworks: The platform requires different frameworks for various SC instances. Each framework has a different set of strengths and is applicable for a specific set of properties of the underlying data.

A multitude of tools are available for the type of processing to be performed on the underlying data, this includes, but is not limited to MapReduce for batch processing, Spark GraphX for iterative processing, Apache Spark and Apache Flink for data stream and real time processing.

Evaluation questions:
- Which tools do you use to transform your (raw) data in order to, for example, perform analysis, filter noise, make it suitable for sorting and querying etc?
  - We process satellite images with Change-Detector, news with social-media with Web Crawlers and convert results to RDF triples with Geotriples.
  - The Event Detector uses modified components and structures of the NewSum algorithm.
  - The Event Detector uses Apache SPARK to execute in a distributed environment, for better scalability.

- Which of these tools are being supported by the generic BDE infrastructure? And did you use them?
  - Except Web-Crawlers, all the others are components of the SC7 Pilot, so they are being supported by the generic BDE infrastructure.

- Did the tools you use, and which are supported by the generic infrastructure, do what you expected? Were they easy to use? Please give a summary about your experiences for each data transformation tool.
  - Yes they are supported and did what we expected. They are easy to use and the documentation is sufficient. Change-Detector successfully
detects possible changes by processing 2 satellite images. Geo-triples successfully converts data into RDF triples, it is easy to use and fast.

Evaluation questions based on the requirements specified in D5.4:

Table 7: Requirements of the Second SC7 Pilot

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Evaluation questions</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>R1</th>
<th>Monitor keyword-based text services (Twitter). Text is retrieved and stored together with provenance and any metadata provided by the service (notably, location).</th>
</tr>
</thead>
</table>
|    | In which way did you adapt the NOMAD data connectors to access the keyword search API of Twitter and Reuters?  
  - A different type of crawling was supported and the interfaces were updated to be better integrated to the BDE logic. |
|    | Is the code released as Open Source?  
  - Yes, the code is available on GitHub repositories in the BDE group. |
|    | In which format did you store the results in Cassandra?  
  - Textual representation for text, long numbers for dates, and WKT format for geometries were used. |
|    | What metadata and provenance did you manage to extract and store?  
  - A full set of meta-data provided by Twitter (author, creation date, location if available, etc.) was kept  
    For RSS information from the feed, such as last update, title, URL, etc was kept. |
|    | How is Cassandra performing in the context of your pilot?  
  - It appears to perform sufficiently well, supporting the storage as expected. |
| R2 | Regularly execute event detection using Spark over the most recent text batch. | Please elaborate on the event detection module. For example, is the code public?  
- The code is open-source and available in the Github repositories of the BDE group.  

Did you describe the algorithms somewhere?  
- The event detection algorithm is based on the work described here:  
  - An appropriate technical report describing the evolved, distributed version of the work implemented for BDE is under preparation for publication. A draft version can be found [here](#) for ease of reference.  

Can you elaborate on the computational performance of your module?  
- The performance of the main algorithm parts and report the findings in the aforementioned draft technical report was studied. |
Did you experience any difficulties with Spark in terms of stability or complexity of usage?

- Not really. However, the smart distribution of represented documents (n-gram graphs) across partitions has significant impact to the final efficiency of the distributed clustering algorithms.
<table>
<thead>
<tr>
<th>R3</th>
<th>Improve the speed of the change detection workflow.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>How did you optimize the scalability of the operators developed in Apache Spark for the change detection workflow?</td>
</tr>
<tr>
<td></td>
<td>● The scalability was optimized by configuring the number of partitions for the RDDs. Different tests were performed before finding the most efficient.</td>
</tr>
<tr>
<td></td>
<td>What is the gained performance?</td>
</tr>
<tr>
<td></td>
<td>● Approximately 15% speed improvement comparing the best with the worst.</td>
</tr>
<tr>
<td>R4</td>
<td>Extend change detection workflow to improve accuracy.</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Did you manage to adapt the fundamental SNAP operators (e.g. Subset and Terrain Correction) for Sentinel 1 to Apache Spark?</td>
</tr>
<tr>
<td></td>
<td>- Subset operator is fully adapted, but the more demanding Terrain Correction operator is implemented in C++.</td>
</tr>
<tr>
<td></td>
<td>Please elaborate on which operators you managed to adapt and how you achieved it.</td>
</tr>
<tr>
<td></td>
<td>What is the performance in terms of accuracy, computing resources and response time?</td>
</tr>
<tr>
<td></td>
<td>- Adding the Subset Operator totally improved the speed of the change-detection workflow. Processing is done in much smaller images and we achieve to be more precise in processing the exact area of user’s interest, while performing it in much less time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R5</th>
<th>Areas of Interest are automatically defined by event detection.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Please provide some insight on the workings of the GIS shape parametrization as part of the event detection module.</td>
</tr>
</tbody>
</table>
|          | - The process is as follows: location names from the text were identified; GADM dataset information was used, with maps locations to geometries; the union of these mapped
<table>
<thead>
<tr>
<th>R6</th>
<th>End-user interface is based on Sextant.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In which ways did you extend the Sextant functionalities?</td>
</tr>
<tr>
<td></td>
<td>● The following functionalities were added:</td>
</tr>
<tr>
<td></td>
<td>● new Change Detection Feature;</td>
</tr>
<tr>
<td></td>
<td>● new Event Detection Feature;</td>
</tr>
<tr>
<td></td>
<td>● new Twitter feed Feature;</td>
</tr>
<tr>
<td></td>
<td>● Authentication/Authorization Feature;</td>
</tr>
<tr>
<td></td>
<td>● History Feature for Classified and Admin users.</td>
</tr>
<tr>
<td></td>
<td>● Integration with Pool Party for entity extraction;</td>
</tr>
<tr>
<td></td>
<td>● Flickr images based on geolocation tag provided for each detected event.</td>
</tr>
</tbody>
</table>

Did it improve the user experience?
- All features were implemented from scratch to meet the user requirements as they were described by SatCen. We were in close communication at each step to design an application that offered all the new functionality that was required in a way that improves the user experience.

How did you evaluate the user experiences?
- The evaluation was conducted with users from SatCen for each separate feature. For each new
functionality we went through several evaluation cycles with feedback from the users, in order to finalize the design of the UI and the back-end functions to meet the user requirements and improve the user experience.
R7 Users must be authenticated and authorized to access the pilot data

In which way did you extend the Sextant system in order to support authentication and authorization?

- An Authentication/Authorization feature was added by designing a module in the interface using Javascript and a PostgreSQL database. Admins in the system can be defined: they have a special menu in the Sextant interface that shows them any new sign-ups, and they can accept or decline the requests.

Can this module easily be reused by other pilots?

- Not at this moment.

4. Conclusion

This report provides the evaluation for the second cycle of pilots for the seven Societal Challenges, the generic BDE platform and recommendations for the next cycle. For all the seven pilots and the BDE platform this report outlined the evaluation results. The main goal of this document is to have the practical and to-the-point open set of questions from D6.7 answered and how the requirements from D5.5 are met. Together with the feedback from the external community gathered via workshops, hangout etc this document not only provides the input needed for the development of the final round of pilots, but also provides valuable insights on determining which Big Data technology fits best to fulfill the specific requirements unique to the pilots from the seven challenges.