Big Data to Enable Global Disruption of the Grapevine-powered Industries

D7.1 - Scalability and Robustness Experimental Methodology

DELIVERABLE NUMBER	D7.1
DELIVERABLE TITLE	Scalability and Robustness Experimental Methodology
RESPONSIBLE AUTHOR	Ida Mele (CNR)

Co-funded by the Horizon 2020
Framework Programme of the European Union

GRANT AGREEMENT N.	780751
PROJECT ACRONYM	BigDataGrapes
PROJECT FULL NAME	Big Data to Enable Global Disruption of the Grapevine-powered industries
STARTING DATE (DUR.)	01/01/2018 (36 months)
ENDING DATE	31/12/2020
PROJECT WEBSITE	http://www.bigdatagrapes.eu/
COORDINATOR	Pythagoras Karampiperis
ADDRESS	110 Pentelis Str., Marousi, GR15126, Greece
REPLY TO	pythk@agroknow.com
PHONE	+30 210 6897 905
EU PROJECT OFFICER	Mr. Riku Leppanen
WORKPACKAGE N. | TITLE	WP7 | Cross-sector Rigorous Experimental Testing
WORKPACKAGE LEADER	CNR
DELIVERABLE N. | TITLE	D7.1 | Scalability and Robustness Experimental Methodology
RESPONSIBLE AUTHOR	Ida Mele (CNR)
REPLY TO	ida.mele@isti.cnr.it
DOCUMENT URL	http://www.bigdatagrapes.eu/
DATE OF DELIVERY (CONTRACTUAL)	30 September 2018 (M9)
DATE OF DELIVERY (SUBMITTED)	28 September 2018 (M9)
VERSION | STATUS	1.0 | Final
NATURE	RE (REPORT)
DISSEMINATION LEVEL	PU (Public)
AUTHORS (PARTNER)	Ida Mele (CNR), Nicola Tonellotto (CNR), Franco Maria Nardini (CNR), Raffaele Perego (CNR), Vinicius Monteiro de Lira (CNR), Cristina Muntean (CNR)
CONTRIBUTORS	Pythagoras Karampiperis (Agroknow), Antonis Koukourikos (Agroknow), Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
REVIEWER	Stefan Scherer (GEOCLEDIAN)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Initial draft	02/09/2018	Ida Mele (CNR), Franco Maria Nardini (CNR)
0.2	Added contributions	11/09/2018	Ida Mele (CNR), Franco Maria Nardini (CNR)
0.3	Added contributions	12/09/2018	Franco Maria Nardini (CNR), Raffaele Perego (CNR), Cristina Muntean (CNR)
0.4	First draft	14/09/2018	Franco Maria Nardini (CNR), Raffaele Perego (CNR), Ida Mele (CNR)
0.5	Added contributions	18/09/2018	Milena Yankova (CNR)
0.6	Input from partners	18/09/2018	Pythagoras Karampiperis (Agroknow), Antonis Koukourikos (Agroknow), Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
0.9	Internal Review	21/09/2018	Stefan Scherer (GEOCLEDIAN)
1.0	Final edits after internal review	28/09/2018	Ida Mele (CNR), Nicola Tonellotto (CNR), Franco Maria Nardini (CNR), Raffaele Perego (CNR), Vinicius Monteiro de Lira (CNR), Cristina Muntean (CNR)

PARTICIPANTS		CONTACT
Agroknow IKE (Agroknow, Greece)		Pythagoras Karampiperis Email: pythk@agroknow.com
Ontotext AD (ONTOTEXT, Bulgaria)		Todor Primov Email: todor.primov@ontotext.com
Consiglio Nazionale Delle Richerche (CNR, Italy)		Raffaele Perego Email: raffaele.perego@isti.cnr.it
Katholieke Universiteit Leuven (KULeuven, Belgium)		Katrien Verbert Email: katrien.verbert@cs.kuleuven.be
Geocledian GmbH (GEOCLEDIAN Germany)		Stefan Scherer Email: stefan.scherer@geocledian.com
Institut National de la Recherché Agronomique (INRA, France)		Pascal Neveu Email: pascal.neveu@inra.fr
Agricultural University of Athens (AUA, Greece)		Katerina Biniari Email: kbiniari@aua.gr
Abaco SpA (ABACO, Italy)		Simone Parisi Email: s.parisi @abacogroup.eu
APIGAIA (APIGEA, Greece)		Eleni Foufa Email: foufa-e @apigea.com

ACRONYMS LIST

BDG	BigDataGrapes
DSPS	Distributed Stream Processing Systems
IoT	Internet of Things
LDBC	Linked Data Benchmark Council
MIPS	Million instructions per second
MFLOPS	Million floating-point operations per second
MUDD	Multi-dimensional data generator
OWL	Ontology Web Language
RDF	Resource Description Framework
SPARQL	Symantec Protocol and RDF Query Language
SNB	Social Network Benchmark
SPB	Semantic Publishing Benchmark
SDPS	Streaming Data Processing Systems
TPC	Transaction Processing Performance Council

EXECUTIVE SUMMARY

The deliverable D7.1, "Scalability and Robustness Experimental Methodology" consists in a report describing the methodology for assessing the performance of a big data system. In particular, the purpose of the task 7.1 is to develop and to implement a rigorous automated testing methodology for measuring and comparing the efficiency of the components in a big data system. The methodology takes into account the characteristics of the system and also the heterogeneity and distributed nature of big data.

In this first version of the deliverable, we present the general concepts related to big data and its properties (i.e., volume, velocity, variety, and veracity). We analyze the state-of-the-art for big data benchmarking considering different challenges which range from preserving the 4V properties of big data to streaming and scalability issues.

Besides reviewing the literature, we present the steps that can be followed for providing a rigorous testing of the BDG system. We believe that it would be better to follow a layered design where the user interfaces are at the top in order to provide easy access to the benchmarking for the user. Below the interfaces, there are the functional and execution layers. The former allows to capture the data and test generators as well as the metrics; the latter represents the basic operations for configuring the system, converting the data, and analyzing the results.

Another contribution of this deliverable is providing some guidelines that can be helpful in the process of rigorous testing a big data system. We believe that a good approach would be following a standardized benchmarking methodology which is divided into different stages going from the selection of the application domain to the execution of the tests. Since the BDG system is not finalized yet, these guidelines are very general and will be refined and concretized once the system will be developed.

Since in the BDG project the semantic infrastructure is represented by graph databases, we also describe the main limitations of the current benchmarking in the context of relational databases and semantic repositories. Also, we provide some valid solutions for our project, for example, the benchmarks proposed by the Linked Data Benchmark Council (LDBC) which ensure linearity, reliability, repeatability, and easy to measure of the metrics. Additionally, LDBC is open for submissions of novel industry benchmarks which may represent specificity of data distribution in big data applications, and this makes it particularly suitable for the BDG project.

Another contribution of this deliverable is a first proposal on the metrics to use for assessing the performance of the BDG system. Such metrics are chosen based on the datasets employed in the use cases of the BDG project. As for the guidelines, also the metrics are prone to changes since the use cases could be refined during the development of the project and the corresponding datasets and metrics would change accordingly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Big data is a large umbrella under which we group any voluminous amount of structured, semi-structured, and unstructured data that has the potential to be mined along with the whole set of scalable approaches, tools, and methods used for processing and analyzing them. Big data represents an important research area due to the availability of large datasets that can be generated from various sources, e.g., Internet of Things (IoT), social media, and multimedia applications.

Big data systems allow to capture, store, manage, analyze, and visualize huge amount of heterogeneous data by exploiting highly scalable and robust tools and methods. Due to the complexity of the data managed and the scalability requirements, the standard techniques and methodologies traditionally used for measuring and assessing computational performance of software components suffer certain limitations while dealing with big data systems. To evaluate and compare the performance of big data systems, it is thus important to develop benchmarks and methodologies explicitly tailored for them.

In big data benchmarking, data is modeled with four dimensions: Volume, Velocity, Variety, and Veracity, which are also known as the "4V" properties of big data (Han et al.,2015). They are briefly described in the following of this section.

1. *Volume* represents the amount of data such as Terabyte (TB) or Petabyte (PB). Nowadays, data are generated faster than ever and the current speed is expected to increase exponentially over the next decade according to International Data Corporation (IDC). This means that big data generators must generate different volumes of data as input. The data volume has different meanings which depend on the workload. For example, in text processing (e.g., sort), the volume is represented by the amount of textual data. In graph analysis (e.g., social network), the volume is represented by the number of vertices in the graph. In image processing, the volume is represented by the number of pictures/second or the number of pixels/ images.
2. *Velocity* reflects the speed of generating, updating, or processing data which are also known as *data generation rate*, *data updating frequency*, and *processing speed.* The last one is particularly important in real-time applications.
3. *Variety* denotes the range of data types. Data can be structured (e.g., tables), unstructured (e.g., text, images, audio and video files), and semi-structured (e.g., web logs, .xls or .csv files). Big data benchmarking requires data generators that can support the whole spectrum of data types.
4. *Veracity* determines whether the data used in benchmarking reflects the characteristics of raw data. The best way to ensure veracity is to use real datasets. In case these datasets are not available, the alternative would be generating the synthetic datasets either randomly or from some statistic distributions. Since this could affect the veracity, another more recommendable approach would consist in synthetic data generation based on some examples from real datasets. Such synthetically generated datasets would be the input of workloads and guarantee the reality and credibility of the benchmarking results (Han et al.,2015).

In big data system benchmarking, it is of valuable importance to keep in mind these properties in order to make sure that they are preserved by the datasets used for the assessment of the system performance. If this is not respected the system may be tested over a too trivial dataset and the performance could be overestimated or conversely, the dataset may be too challenging and the system performance could result poor. This is especially true when real datasets are not available or they need to be sampled, thus the data generation/sampling process has to ensure the 4V properties of big data. In Section 2, we will discuss the data generation requirements based on a layered design where the functional layer is devoted to preserve all the 4V properties of big data.

Various big data tools such as Hadoop, Storm, Spark are often employed in industry and research communities since they allow huge dataset to be distributed and processed in parallel. Big data applications use big data analytics techniques to efficiently analyze large datasets. However, choosing the suitable big data tools is difficult due to the challenges in processing and applying big data.

The document is structured as follows. We describe the high-level process for benchmarking big data systems in Section 2. This process consists of several steps, and it goes from planning the benchmarking to the final analysis and evaluation of the results. Then, we enlarge upon the process of benchmarking, providing a description of the layered design. Often the development of a benchmarking process is made of three layers: user interface, function, and execution layers. The function layer is the core of the designing, and we analyze it in more details. In particular, we explain the several steps that are needed to generate the datasets that would be used in the evaluation of the system performances. Such datasets must preserve the properties of big data, especially when they are created synthetically or sampled from the real data. Besides data, at this layer the tests must be decided, and they depend on the metrics chosen for the evaluation. Section 2 concludes with the process for generating tests, the five-stage benchmarking methodology, and the requirements for benchmarking the BDG project. The guidelines are general on purpose since the BDG system is not in the final stage and can change during the project.

In Section 3, we review the state-of-the-art in terms of methodologies and tools. Previous research works on benchmarking suites for big data analytics and databases are described. We organize our literature review in different topics based on scalability, cloud storage, preserving the properties of big data, streaming, and security.

In Section 4, we present the limitations of the most popular relational benchmarks and semantic repositories (e.g., UniProt, DBpedia) when applied to the BDG project. We also propose to use the benchmarks by the Linked Data Benchmark Council (LDBC) which seem to be particularly suitable for big data projects as they ensure the linearity, reliability, repeatability, and easy to measure for the performance metrics.

In Section 5, we describe the use cases of the BDG project, and we also propose some metrics that would be used in the benchmarking process for rigorous testing the efficiency of the BDG system. To highlight the metrics that we propose for the assessment of the system performance, we show in a table the use cases grouped by datasets and for each dataset we highlight the operations, formats and the corresponding metrics that can be employed during the assessments. This is only a tentative set of metrics since the use case definitions will be revised and consequently datasets and metrics could change.

Lastly, Section 6 concludes the deliverable.

2 BIG DATA BENCHMARKING

Big data benchmarks are developed to evaluate and compare the performance of big data systems and architectures. Figure 1 shows the benchmarking process for a big data system. It basically consists of five steps: (1) the *planning* step, at which the benchmarking object, the application domain, and the evaluation metrics are determined; (2) the *data generation* step when data is created; (3) the *test generation* step at which the possible tests are determined followed by (4) the *execution* step when the benchmark is conducted and finally (5) the *evaluation and analysis* step which takes care of analyzing and reporting the results.

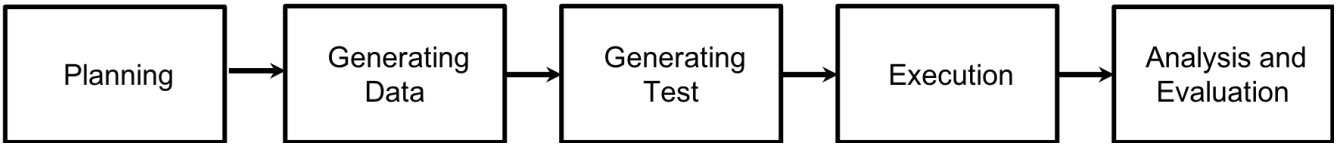

Figure 1: Big Data Benchmarking Process

Successful and efficient benchmarking can provide realistic and accurate measuring of big data systems and thereby it promotes the development of a big data technology. Also, it helps the system owners to make decisions for planning big data systems. For example, benchmarking results can help the system owners to identify the performance bottlenecks in big data systems. The analysis of such bottlenecks can then be leveraged for optimizing the system configuration and the resource allocation.

2.1 LAYERED DESIGN OF BIG DATA BENCHMARKS

Figure 2 shows the design layers of a big data benchmark. There are three layers:

1. *User Interface Layer.* It provides interfaces which help the system owners to specify their benchmarking requirements (e.g., data, workloads, metrics and the preferred data volume and velocity).
2. *Function Layer.* It has three components: *data generators, test generators,* and *metrics*. Briefly, *data generators* are designed to produce datasets covering different data types and application domains, ensuring the 4V properties of big data in these datasets. The *test generators* automatically generate tests with comprehensive workloads for big data systems. The *metrics* can be tailored for the user needs or the architecture.
3. *Execution Layer.* It offers functions for supporting the execution of the benchmark tests over the different software stacks. Specifically, the system configuration tools can be used for running a generated test in a specified software stack. The data format conversion tools are used to transform a generated dataset into a format suitable for the test. Finally, the *result analyzer* and the *result reporter* analyze and display the evaluation results.

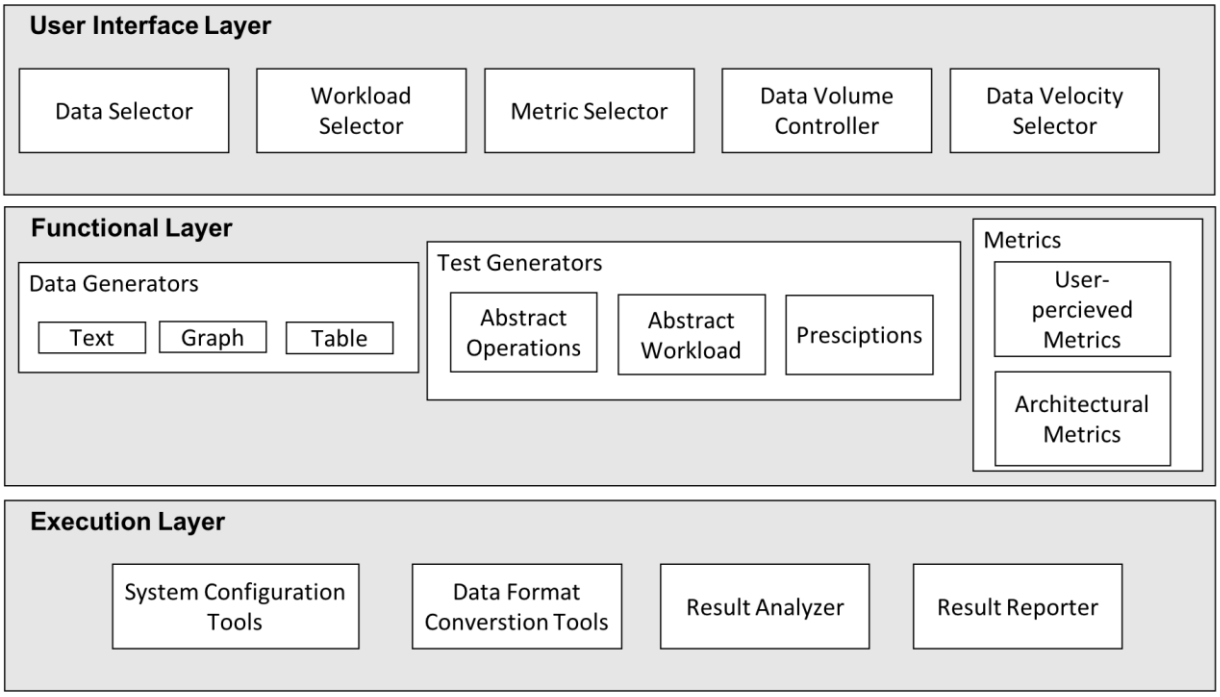

Figure 2: Design Layers of a Big Data Benchmark

Functional Layer. In this paragraph, we describe the different components of the *functional layer* which represents the core of the layered design of a benchmark. The *functional layer* has 3 important components:

1. *Data Generators.* They aim to generate datasets preserving the 4V properties of big data. The data can be represented by textual content, graphs, or tables. Figure 3 shows the process for generating datasets which can be summarized with 4 steps. The first step is for selecting the dataset so that the *variety* of big data is ensured. At this step, real datasets are selected to cover representative application domains and different data sources and types. Sometimes, the generator tools are employed to directly make synthetic datasets, which means that the datasets are independent of real data. At the second step, there is the data processing. Each data generator employs a data model to capture and preserve the important characteristics in one or multiple datasets of a specific data type. At this step, different models have to be developed to capture the characteristics of data and preserving the data *veracity*. Moreover, different data types (e.g., tables, text, streams, and graphs) have to be considered. At the data processing step, the data sampling is also important as it allows to scale down the dataset sizes. At the third step, there is the data generation so that *volume and velocity* can be controlled according to the user requirements for supporting different data generation rates (e.g., parallel and distributed execution over multiple machines). At the fourth step, after the dataset is generated, the format conversion tools transform it into a format that can be used as the input data of a specific workload.
2. *Test Generators.* They generate automatic benchmarking tests. The aim of these generators is abstracting the workload of the current big data systems to create a set of *operations* and *workload patterns* that can be used in big data processing. As reported in Figure 2, test generators consist of three parts:
    a. *Abstract operations.* They represent the abstracted processing actions performed on the datasets (i.e., operators). In the phase of generating tests, the operations can be divided into three categories: (i) element operation, (ii) single-set operation, and (iii) double-set operation, according to the number of datasets processed by them.
    b. *Abstract workload.* These are abstracted patterns which are designed to combine operations to form complex processing tasks. In the test generator, three workload patterns can be

abstracted: (i) a single-operation pattern which contains one single operation; (ii) a multi-operation pattern which is for multiple operations; and (iii) an iterative-operation pattern which is for operations that are executed iteratively. The difference between a multi-operation and an iterative-operation pattern is that the former contains a finite number of operations, while the latter only provides stopping conditions, therefore the exact number of operations can be known only at run time.

c. *Prescriptions.* They include the information needed to produce a benchmarking test, such as datasets, a set of operations and the workload patterns, a method to generate workload, and the evaluation metrics.

3. The *metrics* (single or multiple) can be divided into two types: user-perceived and architecture metrics[1]. User-perceived metrics represent observable metrics that are easy to understand for the users (e.g., duration of a test, request latency, and total throughput). While the user-perceived metrics are used to compare performances of workloads of the same category, the architecture metrics are designed to compare workloads from different categories and they should also take energy consumption and cost efficiency into account. Some examples of architecture metrics are million instructions per second (MIPS) and million floating-point operations per second (MFLOPS).

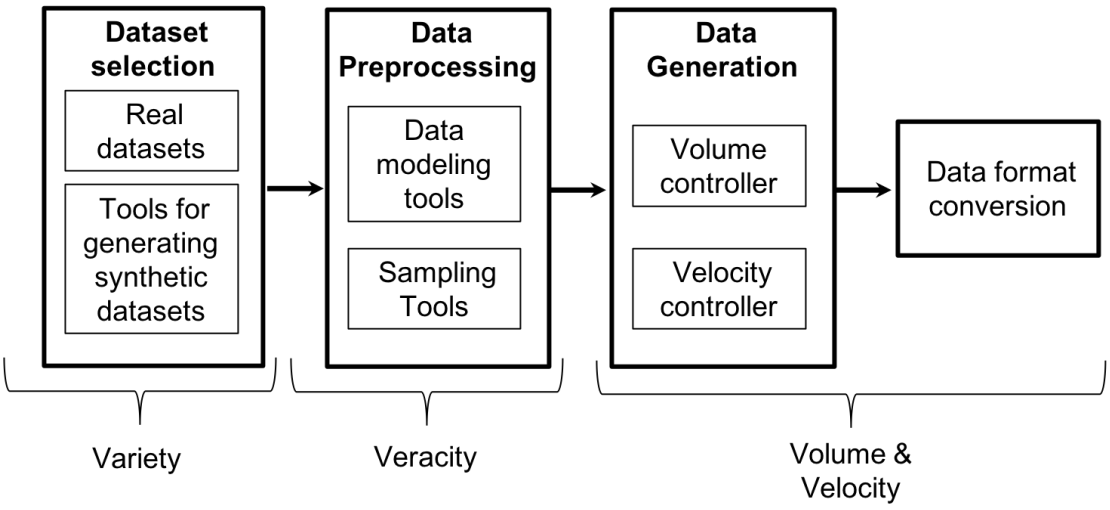

Figure 3: The big data generation process

2.2 GUIDELINES

As we have seen in the previous section, big data benchmarks are developed to evaluate and compare the performance of big data systems and architectures. Hence, benchmarking must provide realistic and accurate measuring of big data systems in order to address two objectives:

1. Promoting the development of big data technology, i.e., developing new architectures (processors, memory systems, and network systems), proposing innovative theories, algorithms and techniques to manage big data and to extract their value and hidden knowledge.
2. Assisting the system owners to make decisions for deciding the system features, for tuning system configurations, validating deployment strategies, and conducting other efforts to improve systems performance. For example, benchmarking results can identify the performance bottlenecks in big data systems, thus guiding the optimization of the system configuration and the resource allocation.

[1] http://prof.ict.ac.cn/BigDataBench/

We now provide some general rules for testing a big data system for the BDG project.

2.2.1 The benchmark test generation process.

In Figure 4, we report the steps for generating a benchmark test. As we can see, at step 1, the dataset is selected. It can be a real dataset or a dataset generated by an automatic tool. At step 2, a set of abstracted operations that can be performed over the data must be decided. This is followed by step 3 at which the set of operations are combined based on the workload patterns. After that, at step 4, a prescription is generated. Lastly, at step 5, a test for a specific system and software stack is created based on the prescription and the system configuration tools. Following these steps, we can create automatically the workloads for the different use cases.

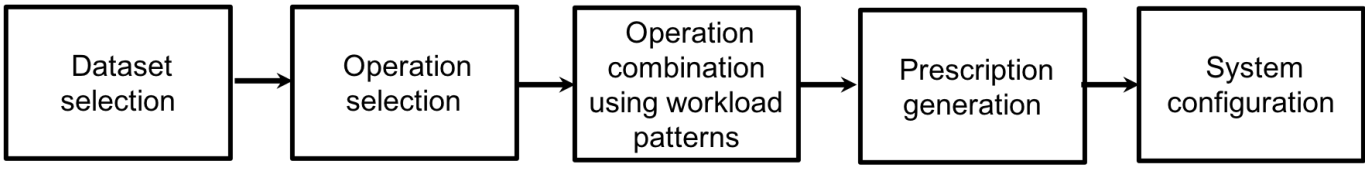

Figure 4: The benchmark test generation process

2.2.2 The preparation of a big data system benchmark.

Figure 5 shows a typical benchmarking methodology for big data systems which consists of five stages. At the stage 1, the application domain is selected followed by stage 2 for choosing the applications that are more suitable for the domain, e.g., the data models, data operations, workload patterns, and metrics. Stage 3 consists in implementing the data generation tools which will produce datasets with the 4V properties. Plus, at this stage the workloads are implemented to support application-specific benchmarking tests. At stage 4, the target system is determined, and the input data is prepared for testing the system. Such operations depend on a *prescription* which includes all the information needed to produce a benchmarking test, including input data, workloads, a method to generate test, and the evaluation metrics. Finally, the benchmark test is carried out at stage 5 and the results can be analyzed and evaluated.

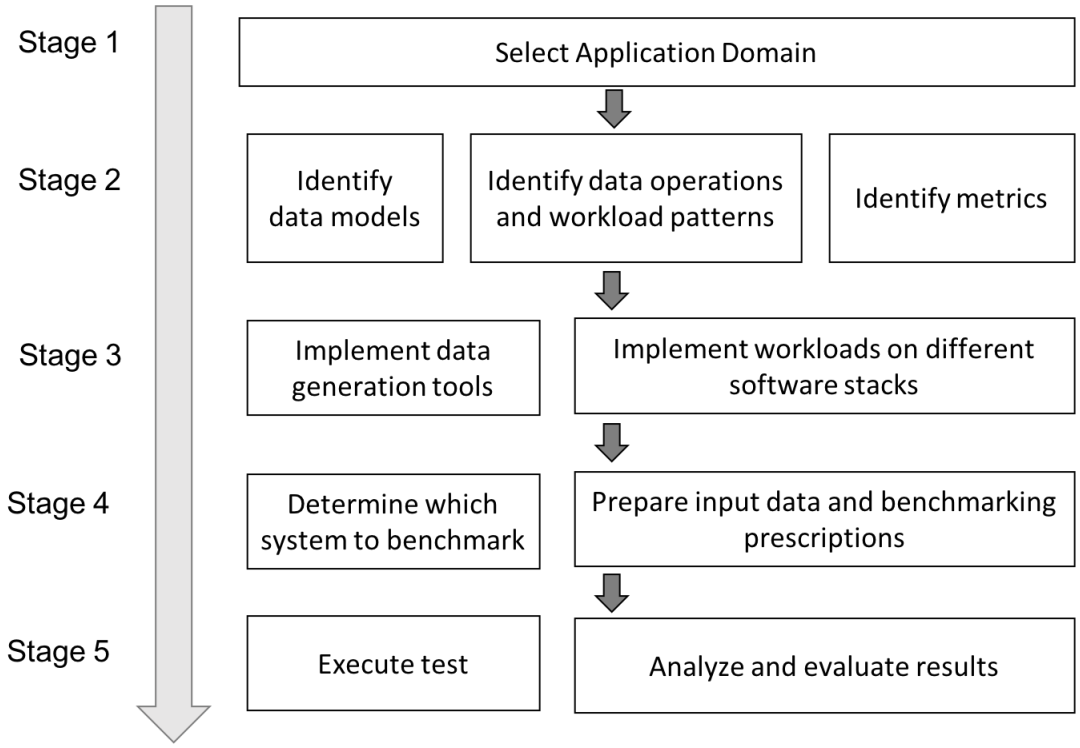

Figure 5: The five-stage benchmarking methodology for a big data system

2.2.3 Benchmarking Requirements for BDG.

When using real data as input of a big data system the data veracity is automatically guaranteed. However, very often it is difficult to use real data since the volume and velocity of this type of data cannot be used due to the different requirements. Hence, in some scenarios we may want to apply synthetic data which preserves the 4V properties of big data. As we have seen, the data should cover different data sources and types to ensure *variety*. This synthetic data can be used as workload of some tasks, such as *Sort* and *WordCount*, in micro benchmarks, and as input for database basic operations, such as *Read*, *Write*, and *Scan*.

To preserve the data veracity there must be different models which can capture the characteristic of real data of different types such as table, text, stream, and graph data. Then, volume and velocity are controlled to fulfill user requirements (as also shown in Figure 3). For example, the data generation can be parallel and distributed over multiple machines for supporting different data generation rates. At the final step, after a dataset is generated, the format conversion tools transform this dataset into an appropriate format which can be used as the input of a workload running on a specific system.

3 STATE-OF-THE-ART

3.1 METHODOLOGIES

Several commercial and open source products have been released for big data storage and processing. Hence, the users can choose which system best suits their needs, and big data system developers have to decide how to evaluate their systems with regard to the big data processing needs. System benchmarking is the classic way of meeting the above requests.

Big data benchmark effort consists in implementing different workloads to evaluate some specific types of systems or architectures.

Regarding big data analysis, most of the benchmarks target the popular Hadoop platform (REN et al., 2012) Pavlo et al. (2009) presented a micro benchmark for big data analytics for comparing Hadoop-based analytics to a row-based Relational DataBase Management System (RDBMS) and a column-based RDBMS one. Also, Gridmix (2013) and HiBench (2010) are benchmarking suites for Hadoop MapReduce and Hive. While GridMix includes micro-benchmarks for only text data, Hibench covers more data types and software stacks. It has four categories of workloads and the inputs of such workloads can be datasets of fixed size or scalable and synthetic datasets. Finally, LinkBench (Armstrong et al., 2013) is based on traces from "social graph" databases (e.g., Facebook data). The authors characterized the data and query workload in many dimensions providing a realistic and challenging test for persistent storage of social and web service data.

3.1.1 Scalability

In order to deal with scalability issues, Ming et al. (2014) proposed a Big Data Generator Suite (BDGS) for generating synthetic data representing big-data workloads. Using real datasets as seed, BDGS generates synthetic data by scaling the seed data and making sure that the characteristics of raw data is preserved. This approach allows to have fully scalability since each data generator can produce synthetic datasets, and its tools for the data format conversion can transform the format of the datasets. The users can specify their preferred data size, and the limit can only be bounded by the storage size and by the BDGS parallelism in terms of the nodes and its running time.

BigDataBench[2] (Wang et al, 2014) applied a similar idea to the scenario of internet services. It is based on a wide class of application domains (search engine, e-commence, and social network). The effectiveness of this framework was tested over six real life datasets that cover three representative data types (structured, semi-structured, and unstructured data), and three data sources (text, graph, and table). Overall, the framework covers several micro benchmarks, three dominant internet services (i.e., Search Engine, Social Network, and E-commerce), and two fast emerging big data domains (multimedia and bioinformatics).

3.1.2 Cloud Storage

Much work focused on benchmarking the cloud and datacenter infrastructure (Ferdman et al., 2010; Cooper et al., 2010). The Yahoo! Cloud Serving Benchmark (YCSB) (Cooper et al., 2010) was designed to provide tools for comparing different cloud data serving systems. This framework is extensible as it supports easy definition of new workloads and makes easy to benchmark new systems. Furthermore, it was used for comparing two non-relational databases (Cassandra and HBase) against one geographically distributed database (Yahoo's PNUTS) and a traditional relational database (MySQL). The CloudSuite benchmark (Ferdman et al., 2011) was implemented to test cloud service architectures using the performance counters of modern servers. The

[2] http://prof.ict.ac.cn/BigDataBench/

authors discovered that the modern processor micro-architecture is inefficient for running study scale-out workloads. Such inefficiency is due to the mismatch between the workload needs and modern processors, particularly in the organization of instruction and data memory systems and the processor core micro-architecture.

3.1.3 Big data properties

All the existing benchmarking techniques may have some limitations with respect to the 4V properties of big data. Concerning the *volume*, benchmarks like HiBench (2010) and LinkBench (Armstrong et al., 2013) are only partially scalable. Instead, BDGS (Ming et al. 2014) represents a fully scalable big data generator suite. Also, *velocity* is oftentimes underestimated, for example, LinkBench (Armstrong et al., 2013), CloudSuite (Ferdman et al., 2011) and BigDataBench (Wang et al., 2014) try to capture dynamic adjustment of data generation speed using parallel strategies. For the *variety*, most of the benchmarks support only one type of data (e.g., Hibench is suitable for unstructured data) and cannot handle multiple data types. Same limitation is encountered for the *veracity* since GridMix (2013), HiBench, and YCSB (Cooper et al., 2010) have a generation process of synthetic data which is independent of real raw data. In particular, HiBench randomly generates synthetic data using the programs in the Hadoop distribution or other statistic distributions. On the other hand, TPC-DS (Ghazal et al., 2013) and BigDataBench (Wang et al., 2014) try to preserve the characteristics of real data. In particular, the Transaction Processing Performance Council (TPC) proposes a series of benchmarks to test the performance of DBMSs in the decision support systems. One of them is the TPC-DS (Ghazal et al., 2013) which implements a multi-dimensional data generator (MUDD).

Han et al. (2015) reviewed all the benchmarking techniques and also proposed a comprehensive workload for big data systems by taking into consideration the bid data issues, namely, generating data which preserves the 4V properties (i.e., volume, velocity, variety and veracity) and generating tests with comprehensive workloads for big data systems.

Similarly, Zhu et al. (2014) tried to generate comprehensive big data workloads called *BigOP*. It is an end-to-end system benchmarking framework featuring the abstraction of representative operations, workload patterns, and prescribed tests. Its abstraction model guides the development of the benchmark and also enables automatic generation of tests with comprehensive workloads. The framework was tested on three big data processing systems, i.e., Hadoop, Spark and MySQL Cluster involving relational data, text and graph data, as well as all operations and workload patterns.

Another example of end-to-end big data benchmark which models different types of data but for the specific business model of product retailers is BigBench (Ghazal et al., 2013). It covers a data model and synthetic data generator that addresses the variety, velocity, and volume aspects of big data systems containing structured, semi-structured, and unstructured data. The structured part of the BigBench data model is adopted from the TPC-DS benchmark, which is enriched with semi-structured and unstructured data components. The semi-structured part captures user clicks on the retailer's website. The unstructured data captures product reviews submitted online. The data generator designed for BigBench provides scalable volumes of raw data based on a scale factor. The BigBench workload is designed around a set of queries against a data model. From a business prospective, the queries cover the different categories of big data analytics. From a technical prospective, the queries are designed to span three different dimensions based on data sources, query processing types, and analytic techniques. The feasibility of BigBench was shown by implementing it on the Teradata Aster Database.

Dimitrov et al. (2013) studied how the conventional optimizations, such as caching, prediction, and prefetching, can be applied to big data workload. They analyzed spatial and temporal reference patterns to bring out several insights related to memory and platform usages (e.g., memory footprints, read-write ratios, bandwidths and latencies). Indeed, trends may intersect by characterizing the memory access patterns of various Hadoop and noSQL big data workloads. Using memory DIMM traces collected using special hardware, the authors analyzed

the spatial and temporal reference patterns to bring out several insights related to memory and platform usages, such as memory footprints, read-write ratios, bandwidths, latencies.

3.1.4 Streaming

Chintapalli et al. (2016) focused on real-time streaming benchmarks. Instead of testing speed-of-light event processing, they created a full data pipeline to closely mimic the real-world production scenarios and compared the performance of three data engines: Flink[3], Storm[4] and Spark Streaming[5,6]

Karimov et al. (2018) presented a framework for benchmarking Streaming Data Processing Systems (SDPS). Differently from other works, their benchmark was specifically tailored for small structured messages. The evaluation consisted on measuring the performance for three open-source SDPS (i.e., Apache Storm, Spark, and Flink) with regard to the throughput and latency of windowed operations (e.g., windowed aggregations and joins). Such operations are important for applications like video games, which require fast processing of large-scale online data feeds from different sources and the windowed aggregations/joins are used for monitoring user feeds. The typical use cases are tracking the in-application-purchases and monitoring the advertisements. The workload adopted for this type of benchmark is derived from an online video game application called Rovio and monitors the user's actions in a given game to ensure that the services work as expected.

In "Benchmarking Distributed Stream Processing Platforms for IoT Applications", Shukla and Simmhan (2016) developed a benchmark suite and performance metrics to evaluate Distributed Stream Processing Systems (DSPS) for streaming IoT applications. Their benchmark included 13 common IoT tasks which were classified across different functional categories in order to form micro-benchmarks. They also proposed two applications: statistical summarization and predictive analytics that leverage the dataflow compositional features of DSPS. The benchmark was validated using Apache Storm DSPS over datasets consisting of observations from smart cities and transportation. The contribution of their work is threefold: (1) they proposed a rigorous classification of the different characteristics of streaming applications and of their data sources; (2) they described performance metrics of DSPS that are necessary to meet the latency and scalability needs of streaming IoT applications; (3) they proposed an IoT Benchmark for DSPS based on representative micro-benchmark tasks. Overall, they aimed at evaluating DSPS in terms of performance and scalability and offer a baseline for big data researchers and developers to uniformly compare DSPS platforms for different IoT domains.

In a follow-up work (Shukla et al., 2017), the authors classified common characteristics of streaming applications and their composition semantics and data sources. They also proposed categories of tasks that are frequently used by IoT applications and the key features of input data streams from sensors. The proposed real-time IoT benchmark is called RIoTBench and it is based on 27 representative microbenchmark tasks.

3.1.5 Security

Cermak et al. (2016) in "A performance benchmark for NetFlow data analysis on distributed stream processing systems" presented a novel performance benchmark which is based on common security analysis algorithms of NetFlow data to determine the suitability of distributed stream processing systems. More in details, the authors contributed to give specifications of the challenges and requirements for distributed stream processing

[3] http://storm.apache.org/

[4] http://flink.apache.org/

[5] http://spark.apache.org/

[6] https://tsicilian.wordpress.com/2015/02/16/streaming-big-data-storm-spark-and-samza/

systems for effective flow analysis. Also, they proposed a benchmark for measuring the performance of a distributed stream processing system based on flow data processing. Lastly, they compared different distributed stream processing systems to quantify their suitability for the flow analysis. Although the benchmark can be used for any distributed stream processing systems, the authors took into account three popular systems: Samza, Storm, and Spark.

3.2 TOOLS

Nagios is an open source tool for network monitoring[7]. It can be used for ensuring the proper functioning of systems, applications, services, and business processes of any IT infrastructures. More in detail, Nagios checks the network for a timely detection of problems that could be caused by data overload, crashed servers, or network connections. Also, it is able to monitor availability, uptime, and response time of every node on the network and it can deliver the results in a variety of visual representations and reports.

In case of a failure, Nagios alerts the technical staff promptly, allowing them to fix the problem or to take proper countermeasures before the outages may start affecting business processes and the services for end-users or customers.

Nagios consists of 3 main components:

1. *Nagios XI.* It allows to monitor the health of the entire network (e.g., reducing downtime, detecting network incidents). It can be installed on different platforms such as Microsoft, VMware, and Linux. Moreover, there are plenty of third-party add-ons for monitoring all in-house and external applications, services, and systems.
2. *Nagios Log Server*. It can be used for making sense of the data, identifying trends as well as finding security threats. It makes simpler the process of searching log data. It has no data limits and allows to get all the log data in one location ensuring high availability.
3. *Nagios Fusion*. It offers high visibility and scalability to the network. It helps to solve problems that come with multiple networks and geographical separation. It also allows to visualize multiple Nagios XI and Core servers in one location with the purpose of simplifying the network management.

[7] . https://www.nagios.com/

4 EVALUATION OF THE BDG SEMANTIC COMPONENTS

In the context of relational databases there exist some well-established benchmarks, namely the well-known **TPC benchmarks**[8]. Although relational databases are not considered in this project, relational benchmarks introduced several best practice techniques that could serve as inspiration for benchmarking.

One particular limitation of the most influential and commonly used TPC benchmarks is that it uniformly distributed and uncorrelated data produced by generators of data and tests. While this simplifies the setup, it also ignores very challenging big data problems. Another influential benchmark is Star Schema Benchmark[9] which uses a schema that is closer to the canonical form. A star schema is a normalized schema that has a main table and several dimension tables (fact tables) that represent the points of the star.

Since graph databases that form the backbone of the Semantic infrastructure of BigDataGrapes aim at different types of queries (e.g. subgraph, supergraph, pattern match, reachability, shortest path query, etc.), these widespread benchmarks are not adequate for evaluating their performance.

Benchmarking RDF systems presents different challenges than the ones posed in relational database engines. As a result, existing relational benchmarks are not really suitable for RDF benchmarking. This is due to the intricacies of RDF data, which a) are expressed using a simple data model based on the concept of triple and b) is not necessarily accompanied by a schema.

The most popular benchmark for semantic repositories developed to facilitate the evaluation of semantic repositories in a standard and systematic way with support for RDF and OWL is **Lehigh University Benchmark (LUBM)**, defined in "An Evaluation of Knowledge Base Systems for Large OWL Datasets" (Guo et al., 2004). The purpose of the benchmark is to measure the performance of storing and querying of large amounts of data that are created for realistic Semantic Web systems. It employs synthetically generated datasets using a fixed OWL ontology of university organizations, lecturers, teachers, students and courses. It consists of a university domain ontology, customizable and repeatable synthetic data, a set of test queries, and several performance metrics. The complexity of the language constructs used is between OWL Horst and OWL DLP (Fischer et al., 2008; Ontotext Lab., 2009). 14 queries are defined that are used to check the query evaluation correctness and speed of repositories that have loaded a given dataset. The biggest standard dataset is LUBM(50) (i.e. it contains synthetic data for 50 universities). Its size is 6.8 million explicit statements, distributed in 1,000 RDF/XML files with total size of 600 megabytes. For the purposes of scalability measurements many groups have used the LUBM generator to create bigger datasets. This is the dataset which is adapted practically by all major semantic repository vendors. Through the years it played a considerable role in the development of the field. LUMBL benchmark evaluates the performance with respect to extensional queries over a large data set against single ontology in OWL which makes it restricted and does not replicated the big data needs of BigDataGrapes project.

UniProt (Universal Protein Resource, http://www.uniprot.org) is the world's most comprehensive and most popular dataset of information on proteins, created by joining several other resources (Swiss-Prot, TrEMBL, and PIR). UniProt RDF[10] is an RDF representation of the dataset; it is based on an OWL ontology, expressed in a sub-language of OWL Lite, that is more expressive than OWL Horst, but still tractable. It represents one of the largest datasets distributed in RDF and OWL. Processing UniProt is often used as benchmark for scalability and

[8] http://www.tpc.org.

[9] http://simile.mit.edu/wiki/Dataset:_Barton.

[10] http://dev.isb-sib.ch/projects/uniprot-rdf

reasoning capabilities of semantic repositories. Still, very few of the repositories are capable to load UniProt and perform materialization on top of it or to interpret its semantics otherwise.

DBPedia[11] is a dataset represented in RDF the infobox of Wikipedia together with other information related to or derived from Wikipedia. It serves as a connectivity hub of the Linked Open Data (LOD) initiative[12] The diversity of the information represented in DBPedia and the fact that it represents encyclopedic knowledge, makes it an excellent resource for benchmarking semantic repositories. DBPedia version 3.3 includes 362 million unique statements without the **owl:sameAs** links. When used for benchmarking of semantic repositories it has several advantages, compared to the straight-forward usage of DBPedia, and it is more diverse as it represents data and data modelling patterns from several other datasets as well. One should consider for instance that DBPedia and Geonames are datasets of a very different nature. A repository which performs well on Geonames could show poor performance when dealing with DBPedia for various reasons; one of them being that Geonames uses few tens of different predicates, whereas DBPedia uses around a hundred thousand predicates, which makes it difficult to apply to BigDataGrapes datasets.

Berlin SPARQL Benchmark, (Bizer and Sch, 2008) is another benchmark that focuses on evaluating performance of query engines in an e-commerce use case: searching products and navigating through related information. Randomized "query mixes" (of 25 queries each) are evaluated continuously through a client application that communicates with the repository through SPARQL endpoint. The benchmark enables evaluation with respect to the changing sizes of the dataset and differing numbers of simultaneous clients.

Although created for the benchmarking of SPARQL engines, the design of BSBM favors relational databases and other raw-store-based implementations, as long as they deal with a single fixed data schema and uniform dense data representation. Generally, RDF databases are designed to deal efficiently with diverse data, integrated from multiple data sources, encoded against different data schema, resulting in sparse data tables in relational databases.

A particularly suited set of benchmarks for desired graph database functionalities is **LDBC**[13] **(Linked Data Benchmark Council)**. As a starting point, LDBC considers the following proposal of desirable characteristics for performance metrics (Lilja, 2005):

- Linearity:
  - It means that the value of the performance metric should be linearly proportional to the actual system performance. That is, if the valued of the metric changes by a ratio, the actual performance of the system should change by the same radio.
- Reliability:
  - A performance metric is reliable if a system A always outperforms system B when the corresponding values of the metric for both systems indicate that system A should outperform system B (i.e. larger value better performance)
- Repeatability:
  - A performance metric is repeatable if the same value of the metric is measured each time the experiment is performed. It implies that a good metric is deterministic.
- Easy to measure: If a metric is not easy to measure, it is unlikely that anyone will actually use it

LDBC benchmarks includes performance metrics as well as cost-based metrics (price/performance). Additionally, LDBC considers new approaches according to the requirements in current technologies and application domains. For example, a metric for robustness can be used to measure the ability of a database to

[11] More information about DBPedia is provided by its developers in Chapter 6 "Semantic Annotation and Retrieval: Web of Data".

[12] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/

[13] http://ldbcouncil.org/

perform well under a variety of conditions, including adverse runtime conditions such as unexpected data skew or resource contention (Wierner et al., 2009).

The LDBC currently has developed two new benchmarks: one for the Social Network Benchmark (SNB) and one for the Semantic Publishing Benchmark (SPB). The SNB aims at testing graph data management technologies for three scenarios: interactive (transaction query workload), business intelligence (analytical query workload) and graph analytics (graph analysis algorithms, such as PageRank). A wide variety of systems could potentially execute one or more workloads of the SNB. Currently, the Interactive Workload is in draft release stage. The Semantic Publishing Benchmark is based on the BBC news website and models a mixed query and update workload with a limited amount of semantic inferencing.

LDBC is open for submissions of new industry specific benchmarks (datasets) to reflect the specificity of data distribution in big data applications which makes it particularly suitable for BigDataGrapes project.

5 BDG PERFORMANCE CRITERIA

5.1 SCALABILITY ISSUES AND METRICS

We now propose some metrics that can be used for assessing the performance of the big data system tailored on the BDG project. The project has 4 use cases which share datasets and hence bottlenecks that would need to be considered in the benchmarking.

Use Case A. Earth Observation Data Anomaly Detection and Classification. The purpose of this use case is to develop models that differentiate between Earth Observation data issues and anomalies. This is for triggering warnings to farmers concerning farm management practices or damage events. The datasets involved in this use case are: Sentinel-2, Landsat-8, TripleSat, provided by GEOCLIDIAN partner. Plus, data from AUA PA Lab consisting of: soil and elevation maps, IoT stationary data, canopy sensing and laser scanner as well as images captured by drones. The first type of data is in GEOTIFF and PNG format plus metadata in JSON format. The remaining type of data is mostly semi-structured such as .csv and .xls files.

Use Case B. Yield and Quality Prediction. The purpose of this use case is to leverage historical earth observation data combined with additional relevant information to make educated guesses about yield and wine quality. Regarding the yield prediction, GEOCLIDIAN would provide Earth Observation data (e.g., Sentinel-2, Landsat-8, TripleSat) plus other datasets for weather, soil, elevation, exposition, variety, anomalies (e.g., pests/diseases), management practices, and historical yield/quality. Earth Observation datasets are in GEOTIFF and PNG format plus metadata in JSON format. For predicting biological efficacy, the data would be provided by AUA PA Lab and APIGEA. Most of this data is semi-structured (.csv and .xls files). Finally, for the crop quality prediction, data used for predicting biological efficacy would be enriched with data collected by proximal sensors, multispectral and thermal cameras in the JSON, GEOTIFF, PNG format and by the land-based weather data which is binary.

Use Case C. Farm management. This use case would take care of optimizing the farm practices and of management. This would mean modelling climate, sunlight exposure, soil quality, slope and topography to predict the vine specific needs. Indeed, an inadequate management, such as over or under cropping, irrigating, spraying, inadequate pruning and poor canopy management, can affect the grape quality. Each plant has its own specific needs and can be monitored by sensors. AUA PA Lab would provide sensor data which would be integrated with the satellite imagery geo-spatial data from GEOCLIDIAN in order to initiate the management procedures required to produce high-quality grapes.

Use Case D. Risk Assessment. The purpose of this use case would be minimizing the waste in the production as we all as ensuring minimal impact on the environment. This would be possible by analyzing the weather and land surface elevation data for a specific field and also its neighbors. Data is represented by images from North American cartographic plus land-based weather data, IoT stationary data, canopy sensing, laser scanner, drone images, as well as farm data and Eca sensing.

The use cases rely on satellite imagery geo-spatial data. Due to the heavy task of image processing, the main bottleneck would be represented by the num. of images/sec and the num. of pixels/sec in images. Moreover, other datasets with measurements and sensor data would be leveraged to predict the grape quality and improving the management and production in vineyards. Due to this variety of data (i.e., GEOTIFF, JSON, and PNG plus .csv, .xls, and binary files), the benchmarking should be able to model heterogeneous data and to handle their potentially huge size. Finally, one would need fast processing of streaming data in order to take decisions on-the-fly.

We summarized in a table the datasets with the purpose of highlighting the metrics that would be used to evaluate the performance of the system during the benchmarking process.

In Table 1, for each dataset we report its operations, the format and the proposed metrics. Moreover, for each dataset, we show the use cases that make use of it.

Table 1: Datasets of the BDG project with corresponding operations, format, and proposed metrics. The last column reports the use cases in which the dataset is needed.

Dataset	Operations	Format	Metrics	Use Case Where the Dataset is used
Sentinel-2	Preprocessing, Atmospheric Corrections, Parcel Extraction, Normalized Difference Vegetation Index (NDVI), other Vegetation Indices	JSON, GEOTIFF, PNG	Num. of images/sec, Num. of pixel/image, Total Test duration, Request latency, Total Throughput	Use case A Use case B (Yield Prediction) Use Case B (Crop quality prediction) Use Case C
Landsat-8	Preprocessing, Atmospheric Corrections, Normalized Difference Vegetation Index (NDVI), other Vegetation Indices	JSON, GEOTIFF, PNG	Num. of images/sec, Num. of pixel/image, Total Test duration, Request latency, Total Throughput	Use case A Use case B (Yield Prediction Use Case B (Crop quality prediction) Use Case C
TripleSat	RGB, NDVI	GEOTIFF	Num. of images/sec, Num. of pixel/image	Use case A Use case B (Yield Prediction) Use Case B (Crop quality prediction) Use Case C
North American Cartographic Information Society (NACIS)	Geographical data (Hypsometric, Physical, and Administration Data)	ESRI Shapefile format	Num. of records/sec, Num. of vertices/sec	Use case A Use case B (Yield Prediction) Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D
Land-based Weather Data	Soil Temperature, Soil Moisture/Water Content, Humidity	Binary	Num. of bytes/sec	Use case A Use case B (Yield Prediction) Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D
IoT stationary data	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use case A Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D

Canopy sensing	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use case A Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D
Laser scanner	Data filtering for establishing thresholds	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use case A Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D (Environmental Risk Assessment) Use Case D (Insurance risk management)
Drone imagery	Autocallibrated	GEOTIFF	Num. of images/sec, Num. of pixel/image, Total Test duration	Use case A Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D
Eca sensing	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use case A Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction) Use Case C Use Case D (Environmental Risk Assessment) Use Case D (Insurance risk management)
Soil/Elevation Maps				Use Case A
Catalogue of vine varieties registered in France				Use case B (Yield Prediction)
viticulture experimental data				Use case B (Yield Prediction)
Grape berry quality analysis (non destructive analysis from experimental field)				Use case B (Yield Prediction)
Plants phenotype variables (automatic measurement under controlled or semi-controlled				Use case B (Yield Prediction)

climatic scenarios)				
Crop Calendar		doc, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case B (Predicting Biological Efficacy) Use Case B (Crop quality prediction)
Vine Leaf Variety 1 – Extraction Method 1	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case B (Predicting Biological Efficacy)
Vine Leaf Variety 1 – Extraction Method 2	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case B (Predicting Biological Efficacy)
Vine Leaf Variety 2 – Extraction Method 1	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case B (Predicting Biological Efficacy)
Vine Leaf Variety 2 – Extraction Method 2	Data filtering for outliers	csv, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case B (Predicting Biological Efficacy)
SRTM DEM		JSON, GEOTIFF, PNG	Num. of pixels/sec, Num. of images/sec, Num. of pixel/image	Use Case B (Crop quality prediction)
HWSD soil data		Raster (.bil) & MS Access DB(.mdb)	Num. of images/sec, Num. of pixel/image, Num. of records/sec, Num. of tables/sec	Use Case B (Crop quality prediction)
Winemaking offline data				Use Case B (Crop quality prediction)
Winemaking online data				Use Case B (Crop quality prediction)
Offline data from biological analyses				Use Case B (Crop quality prediction)
Sensory analysis				Use Case B (Crop quality prediction)
Weather Data		csv	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case C (Optimization of Farm Practices in the Vineyard)
Grape berry and plant microclimate monitoring		xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case C (Optimization of Farm Practices in the Vineyard)
French soil database				Use Case C (Optimization of Farm Practices in the Vineyard)
Farm Data		doc, xls	Num. of rows/sec, Num. of columns/sec, Num. of bytes/sec	Use Case C Use Case D
Toxicity on human skin cells				Use Case D (Grape and Wine Quality Risk Assessment (safety))

6 SUMMARY

The deliverable 7.1, "Scalability and Robustness Experimental Methodology", belongs to WP7 which is about "Cross-sector Rigorous Experimental Testing". The purpose of this first version of the deliverable is to report on the methodology that can be used for testing the scalability and robustness of a big data system and to propose the metrics for the assessment.

We have presented the process for implementing an automated and rigorous experiment which would allow to measure the efficiency of the big-data-system components. The benchmarking has to consider not only the characteristics of the system but also the properties of big data.

As further contribution, we have provided the guidelines that could be followed in the assessment of the BDG system. They are still general since the system is in at an embryonal stage, but we plan to extend this part and to delve more into details in the next version of the deliverable when the system will be available.

Part of this deliverable is a review of the state-of-the-art about benchmarking methodologies. Also, we describe a tool (Nagios) that can be used for network monitoring.

Finally, we have proposed some metrics that could be used in the assessment of the big data system. Since they depend on the datasets used in the use cases of the project, this is only a tentative list of metrics which can be further expanded or modified based on the changes that will involve both the use cases and the datasets.

7 REFERENCES

Armstrong, T., G., Ponnekanti, V., Borthakur, D., Callaghan. M. (2013). Linkbench: a database benchmark based on the Facebook social graph. In Proceedings of the SIGMOD, ACM, 1185-1196.

Bizer, C. and Schultz, A.(2008): Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints. In: Proceedings of the 4th International Workshop on Scalable Semantic Web knowledge Base Systems, SSWS2008.

Čermak, M., Tovarnak, D., Lastovicka, M. and Čeleda, P.(2016). A performance benchmark for NetFlow data analysis on distributed stream processing systems. NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, pp. 919-924. doi: 10.1109/NOMS.2016.7502926.

Chintapalli, S., Dagit, D., Evans, B., Farivar, R., Graves, T., Holderbaugh, M., Liu, Z., Nusbaum, K., Patil, K., Peng, B.J. and Poulosky, P.(2016). Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming. IEEE International Parallel and Distributed Processing Symposium Workshop.

Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R. and Sears, R.(2010). Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing ACM, pages 143-154.

Dimitrov, M., Kumar, K., Lu, P., Viswiswanathan, V., Willhalm, T.(2013). Memory System Characterization of Big Data Workloads. IEEE International Conference on Big Data.

Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., Kaynak, C., Popescu, C., A.D., Ailamaki, A. and Falsafi, B.(2011). Clearing the Clouds: A Study of Emerging Workloads on Modern Hardware. In Proceedings of the 17th Conference on Architectural Support for Programming Languages and Operating Systems-ASPLOS 2012, pages 1-11.

Fischer, F., Keller, U., Kiryakov, A., Huang, Z., Momtchev, V. and Simperl, E. (2008): Initial Knowledge Representation Formalism. LarKC project deliverable D1.1.3, http://www.larkc.eu/wp-content/uploads/2008/01/larkc_d113-initial-knowledge-representation-formalism_m7.pdf.

Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A. and JACOBSEN, H.A. (2013). BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. In SIGMOD ACM, New York, New York, 2013, pages 197-1208.

GridMix. (2013). Available online: https://hadoop.apache.org/docs/r1.2.1/gridmix.html.

Guo, Y., Pan, Z. and Heflin, J. (2004): An Evaluation of Knowledge Base Systems for Large OWL Datasets. Journal of Web Semantics, 3(2), 2005, pp. 158-182.

Han, R., Zhen, J., Gao, W., Xinhui, T. and Wang, L. (2015). Benchmarking Big Data Systems: State-of-the-Art and Future Directions. TECHNICAL REPORT. ICT, ACS.

Huang, S., Huang, J., Dai, J., Xie, T. and Huang B. (2010). The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on IEEE, pages 41-51.

Karimov, J., RABL, T., Katsifodimos, A., Samarev, R., Heiskanen, H., Markl, V. (2018). Benchmarking Distributed Stream Data Processing Systems. ICDE.

Lilja, D. J.(2005). Measuring Computer Performance - A Practitioner's Guide. Cambridge University Press.

Ming, Z., Luo, C., Gao, W., Han, R., Yang, Q., Wang, L. and Zhan, J. (2014). BDGS: A Scalable Big Data Generator Suite in Big Data Benchmarking. In CoRR.

Ontotext Lab. (2009). RDF(S), Rules, and OWL Dialects. http://www.ontotext.com/inference/rdfs_rules_owl.html

Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., Dewitt, D. J., Madden, S. and Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, SIGMOD '09, pages 165–178.

Ren, Z., Xu, X., Wan, J., Shi, W., and Zhou, M. (2012). Workload characterization on a production Hadoop cluster: A case study on Taobao. In IEEE International Symposium on Workload Characterization, pages 3–13.

Shukla, A., Chaturvedi, S., Simmhan, Y.(2017). RIoTBench: An IoT benchmark for distributed stream processing systems. Concurrency Computat: Pract Exper.

Shukla A. and Simmhan, Y. (2016). Benchmarking Distributed Stream Processing Platforms for IoT Applications. CoRR abs/1606.07621. arXiv:1606.07621 http://arxiv.org/abs/1606.07621.

Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., Zhen, C., Lu, G., Zhan, K. and Qiu., B. (2014). BigDataBench: a Big Data Benchmark Suite from Internet Services. In Proceedings of the The 20th IEEE International Symposium On High Performance Computer Architecture (HPCA 2014).

Wierner, J. L., Kuno, H. and Graefe, G.(2009). Benchmarking query execution robustness. In TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC).

Zhu, Y., Zhan, J., Weng, C., Nambiar, R., Zhang, J., Chen, X. and Wang, L.(2014). Generating comprehensive big data workloads as a benchmarking framework. The 19th International Conference on Database Systems for Advanced Applications (DASFAA 2014).