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

D4.1 - Methods and Tools for Scalable Distributed Processing

DELIVERABLE NUMBER	D4.1
DELIVERABLE TITLE	Methods and Tools for Scalable Distributed Processing
RESPONSIBLE AUTHOR	Vinicius Monteiro de Lira (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	Nikos Manouselis
ADDRESS	110 Pentelis Str., Marousi, GR15126, Greece
REPLY TO	nikosm@agroknow.com
PHONE	+30 210 6897 905
EU PROJECT OFFICER	Ms. Annamária Nagy
WORKPACKAGE N. | TITLE	WP4 | Analytics and Processing Layer
WORKPACKAGE LEADER	CNR
DELIVERABLE N. | TITLE	D4.1 | Methods and Tools for Scalable Distributed Processing
RESPONSIBLE AUTHOR	Vinicius Monteiro de Lira (CNR)
REPLY TO	vinicius.monteirodelira@isti.cnr.it
DOCUMENT URL	http://www.bigdatagrapes.eu
DATE OF DELIVERY (CONTRACTUAL)	30 September 2018 (M9), 31 December 2019 (M24, Upadated version)
DATE OF DELIVERY (SUBMITTED)	28 September 2018 (M9), 30 December 2019 (M24, Upadated version)
VERSION | STATUS	2.0 | Final
NATURE	DEM (Demonstrator)
DISSEMINATION LEVEL	PU (Public)
AUTHORS (PARTNER)	Franco Maria Nardini (CNR), Raffaele Perego (CNR), Vinicius Monteiro de Lira (CNR), Ida Mele (CNR), Cristina Muntean (CNR), Nicola Tonellotto (CNR)
CONTRIBUTORS	Florian Schlenz (Geocledian), Toni Frank (Geocledian), Johannes Sommer (Geocledian)
REVIEWER	Panagis Katsivelis (Agroknow)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	First draft	11/09/2018	Vinicius Monteiro de Lira (CNR)
0.2	Updates	12/09/2018	Nicola Tonellotto (CNR)
0.3	Improving Section 2 and Section 3	13/09/2018	Vinicius Monteiro de Lira (CNR), Florian Schlenz (Geocledian), Toni Frank (Geocledian), Johannes Sommer (Geocledian)
0.4	Updates	14/09/2018	Nicola Tonellotto (CNR)
0.5	Updates	14/09/2018	Raffaele Perego (CNR)
0.6	Updates	14/09/2018	Vinicius Monteiro de Lira (CNR), Nicola Tonellotto (CNR)
0.7	Pre-final draft	14/09/2018	Nicola Tonellotto (CNR), Franco Maria Nardini (CNR), Raffaele Perego (CNR), Vinicius Monteiro de Lira (CNR), Ida Mele (CNR), Matteo Catena (CNR), Cristina Muntean (CNR), Ida Mele (CNR)
0.9	Internal Review	26/09/2018	Milena Yankova (ONTOTEXT)
1.0	Partners review, final comments and edits	28/09/2018	Nicola Tonellotto (CNR)
1.1	Updates	01/12/2019	Vinicius Monteiro de Lira (CNR)
1.2	Updates	06/12/2019	Florian Schlenz (Geocledian)
1.3	Updates	16/12/2019	Franco Maria Nardini (CNR)
1.4	Updates, deliverable ready for internal review	18/12/2019	Vinicius Monteiro de Lira (CNR), Franco Maria Nardini (CNR), Florian Schlenz (Geocledian)
1.5	Internal Review	30/12/2019	Panagis Katsivelis (Agroknow)
2.0	Final polishing	30/12/2019	Franco Maria Nardini (CNR)

PARTICIPANTS	CONTACT
Agroknow IKE (Agroknow, Greece)	Nikos Manouselis Email: nikosm@agroknow.com
Ontotext AD (ONTOTEXT, Bulgaria)	Todor Primov Email: todor.primov@ontotext.com
Consiglio Nazionale Delle Ricerche (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
SYMBEEOSIS LONG LIVE LIFE S.A. (Symbeeosis, Greece)	Konstantinos Gardikis Email: c.gardikis@gmail.com

ACRONYMS LIST

BDG	Big Data Grapes
BDE	Big Data Europe
HDFS	Hadoop File System
NDVI	Normalized difference vegetation index
NDWI	Normalized difference water index
CPT	Cluster Performance Tool

EXECUTIVE SUMMARY

This accompanying document for deliverable D4.1 Methods and Tools for Scalable Distributed Processing describes the main mechanisms and tools used in the BigDataGrapes (BDG) platform to support efficient processing of large datasets in the context of grapevine-related assets. The BDG software stack designed provides efficient and fault-tolerant tools for distributed processing, aiming at providing scalability and reliability for the applications.

The document first introduces the big picture of the architecture of the BDG platform and the main technologies currently used in the *Persistence* and *Processing Layers* of the platform to perform efficient data processing over extremely large dataset.

Second, the requirements needed to run the BigDataGrapes platform are introduced and discussed, by also providing instructions to set up and to launch the platform. The platform has been built, re-using and customizing the software stack of the Big Data Europe (BDE, https://www.big-data-europe.eu/). Besides the customization of some existing components, the BigDataGrapes software stack extends the BDE to better support efficient processing and distributed predictive analytics of geospatial raster data in the context of precision agriculture and Farm Management Systems. Furthermore, all the platform components have been designed and built using Docker containers. They thus include everything needed to deploy the BDG platform with a guaranteed behavior on any suitable system that can run a Docker engine.

Third, to provide the reader with practical examples of usage of the current release of the BDG platform, we report about two demos that have been already developed on the top of it by the project's partner. Specifically, the two demonstrators perform scalable operations on geospatial raster data using the Spark-based GeoTrellis geographic data processing engine provided by the BDG platform. The first demo regards the tiling of large raster satellite images. Tiling is a mandatory process that allows the large raster datasets to be split-up into manageable pieces that can be processed on parallel and distributed resources. As a second demonstrator, the tiles previously computed are processed to extract from each tile image two relevant vegetation indexes. The first index is the normalized difference vegetation index (NDVI), an indicator that allows to assess at what degree the target being observed contains live green vegetation or not. The second index is a version of the Normalized Difference Water Index (NDWI), most appropriate for vegetation water content mapping.

Fourth, we present the BDG Cluster Performance Tool (CPT), an application developed to monitor the status of the cluster, i.e., the spark nodes, dedicated to process satellite images. On the front-end side, the tool has a Web application that runs a dashboard built upon *Kibana*[1]. Moreover, on the back-end side, the tool uses: i) an API that collects the performance of the nodes composing the cluster and ii) an *ElasticSearch*[2] database to store the performance statistics of the nodes.

Finally, we report the lessons learned from the exploitation of the BDG data processing tools in a real-world satellite processing architecture. We report how GEOCLEDIAN moved from a monolithic architecture to a modular one after an analysis aiming at identifying bottlenecks and congestion. The new modular architecture allows GEOCLEDIAN to speed up the processing of satellite images by 3x. Moreover, the new architecture will be used to deploy the new BDG distributed data processing components presented in this deliverable for a fast tiling and index computation from satellite images.

1 https://www.elastic.co/products/kibana
2 https://www.elastic.co/

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The BigDataGrapes platform aspires to provide components that go beyond the state-of-the-art on various stages of the management, processing, and usage of grapevine-related big data assets thus making easier for grapevine-powered industries to take important business decisions. The platform employs the necessary components for carrying out rigorous analytics processes on complex and heterogeneous data helping companies and organizations in the sector to evolve methods, standards and processes based on insights extracted from their data.

For this purpose, the BigDataGrapes software stack has been designed and built using core technologies and frameworks for efficient processing of large datasets, such as Apache Spark and Apache Hadoop, making sure that the execution environment and methodology retain scalability and efficient use of the available computational resources. The distributed execution paradigm serves as the basis for efficiently solving the equally urgent challenge of Heterogeneity and Scalability in the context of Big Data processing.

Figure 1 shows a logical view of the BigDataGrapes software stack organized into architectural layers.

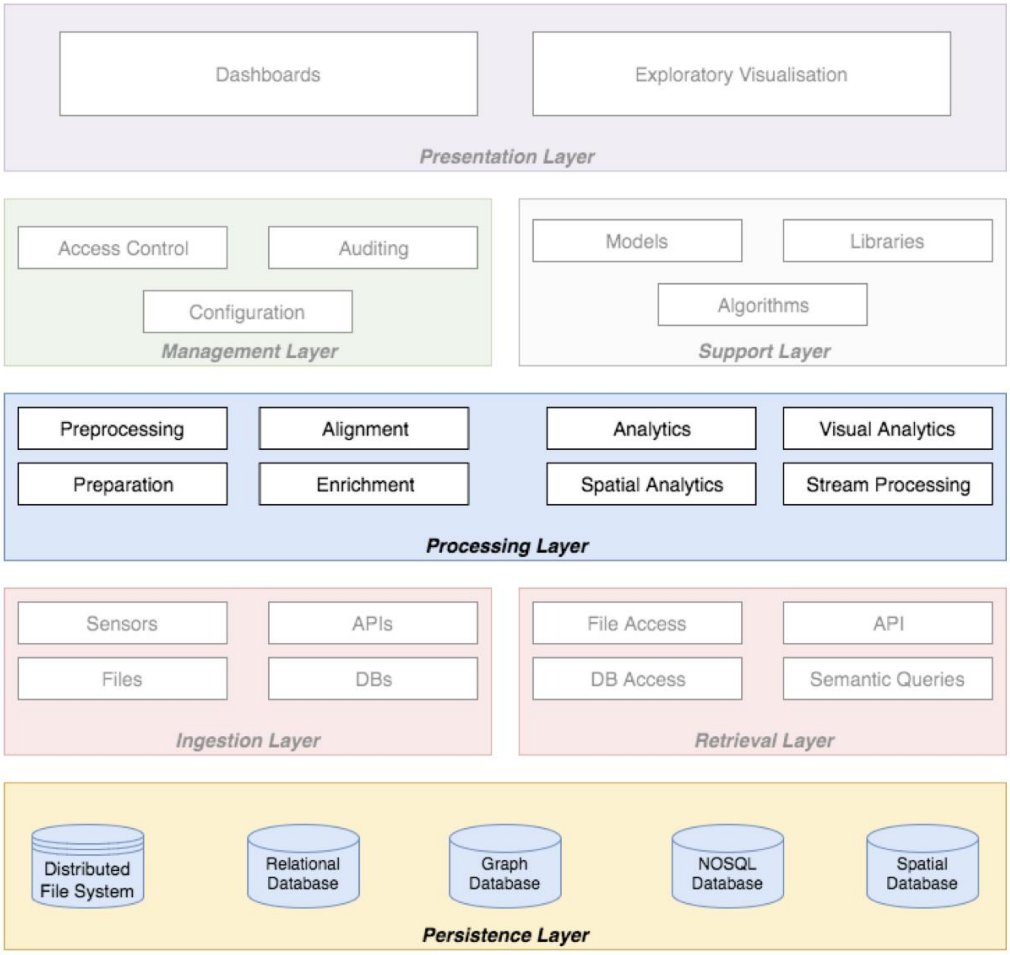

Figure 1: BigDataGrapes Architecture Layers

In this revised version of the deliverable **we focus on the *Persistence* and *Processing Layers*** of the architecture shown in the figure. These two layers are in fact the ones providing the core methods and tools for scalable

distributed processing. Specifically, the *Persistence layer* deals with the long-term storage and management of data handled by the platform. Its purpose is to consistently and reliably make the data available to the processing layer. In turn, the *Processing Layer* implements the core processes for data management and analysis towards serving the analytics and decision support requirements of the BigDataGrapes use cases. An example of how the above stack can effectively support scalable machine learning and analytics solutions is given in Deliverable 4.3 "Models and Tools for Predictive Analytics over Extremely Large Datasets", presenting the first version of the BDG components for big data analytics. Together, these are the keys layers of the platform to provide Heterogeneity and Scalability. The remaining layers are deeply discussed in the Deliverable 2.3 of this project "BigDataGrapes Software Stack Design".

The core components have been developed individually as parts of an integrated software stack. Driven from the data problems and challenges coming from the real-world operation and business models of the involved grapevine-powered industries, BigDataGrapes has developed methodologies and implemented software stacks to serve their specific requirements. Nevertheless, it will be extended and expanded in the following of the project to better fit their evolving needs. Thanks to the modular and portable design, the BDG architecture and its individual components can also be easily adapted to be reused in different contexts and scenarios.

The rest of this document is organized as follows: Section 2 highlights the main technologies used in the BDG software stack. Section 3 describes how to set up and run the current version of the platform. Section 4 exhibits two demonstrators that process raster data using the BDG platform. Section 5 presents an analysis of the scalability of the BDG platform in a specific use case: tiling of satellite images. Moreover, Section 6 presents the BDG Cluster Performance Tool, an application that allows for a user-friendly and real-time monitoring of the performance of the BDG architecture. Then, Section 7 reports some lessons learned from the exploitation of the BDG architecture in a real-world infrastructure like the one developed by GEOCLEDIAN. Finally, Section 8 concludes the deliverable by providing a summary of what have been done.

2 BIG DATA PROCESSING TECHOLOGIES

In this section we provide an overview of the main technologies (frameworks and libraries) used for the implementation of the *Persistence* and *Processing Layers* of the BigDataGrapes platform that perform efficient data processing over extremely large dataset.

2.1 PROCESSING LAYER TECHONOLOGIES

2.1.1 Apache Hadoop

The Apache Hadoop software library[3] is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale out from single servers to thousands of machines, each offering local computation and storage. Internally, the framework does not rely on hardware to deliver high-availability. Instead, the framework is designed to detect and handle failures at the application layer. In this way, the Hadoop offers highly-available service on top of a cluster of computers, each of which may be prone to failures. Among the Apache Hadoop's core components, there is the MapReduce programming model.

MapReduce. MapReduce[4] is a programming model that is designed to process the big data. MapReduce runs on large clusters of commodity hardware. It supports a wide range of languages for developers, including C++, Java, or Python. MapReduce is designed to match the massive scale of HDFS (explained in 2.2.1) and Hadoop, allowing to process unlimited amounts of data, fast, all within the same platform where it's stored. It provides reliability a built-in job and task trackers allow processes to fail and restart without affecting other processes or workloads.

2.1.2 Apache Spark

Apache Spark[5] is a fast and general cluster computing system for Big Data. It provides high-level APIs in Scala, Java, Python, and R, and an optimized engine that supports general computation graphs for data analysis. It also supports a rich set of higher-level tools including Spark SQL for SQL and DataFrames, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for stream processing. While MapReduce continues to be a popular batch-processing tool, Apache Spark's flexibility and in-memory performance make it a much more powerful batch execution engine.

Spark has a programming model similar to MapReduce but extends it with a data-sharing abstraction called "Resilient Distributed Datasets," or RDDs. Using this simple extension, Spark can capture a wide range of processing workloads that previously needed separate engines, including SQL, streaming, machine learning, and graph processing. These implementations use the same optimizations as specialized engines (such as column-oriented processing and incremental updates) and achieve similar performance but run as libraries over a common engine, making them easy and efficient to compose. Spark has several important benefits. First, applications are easier to develop because they use a unified API. Second, it is more efficient to combine processing tasks; whereas prior systems required writing the data to storage to pass it to another engine, Spark can run diverse functions over the same data, often in memory. Finally, Spark enables new applications (such

3 http://hadoop.apache.org/

4 https://en.wikipedia.org/wiki/MapReduce

5 https://spark.apache.org/

as interactive queries on a graph and streaming machine learning) that were not possible with previous systems.

MLlib. MLlib[6] is the machine learning library for Apache Spark. It is a fully-fledged tool containing many algorithms and utilities for several tasks ranking from classification (logistic regression, naive Bayes, etc.) to regression (generalized linear regression, survival regression, isolation regression, etc.) and gradient-boosted algorithms based on trees, e.g., random forests, multiple additive regression trees. It also provides distributed methods for clustering (K-means, Gaussian mixtures (GMMs), etc.) and for mining frequent itemsets, association rules, and sequential pattern mining. MLLib also allows for a complete pre-processing of the dataset to use exploiting the underlying distributed Spark environment. The design and deployment of the BDG component providing the MLib functionalities along with simple demonstrators of its use in the context of the BigDataGrapes project are included in Deliverable 4.3 "Models and Tools for Predictive Analytics over Extremely Large Datasets".

2.1.3 Flink

For the purposes of stream processing in BigDataGrapes, a powerful, well-supported combination of technologies is the usage of Apache Flink[7] . Apache Flink is an open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications. Flink is based on the DataFlow model i.e. processing the elements as and when they come rather than processing them in micro-batches. Dataflow allows Flink to process millions of records per minutes at milliseconds of latencies on a single machine. Micro-batches can contain huge number of elements and the resources needed to process those elements at once can be substantial. In the case of a sparse data stream in which you get only a burst of data at irregular intervals, it can become a major pain point. Furthermore, Flink provides robust fault-tolerance using checkpointing (periodically saving internal state to external sources such as HDFS).

2.1.4 Geotrellis

GeoTrellis[8] is a geographic data processing engine that uses Spark to work with raster data for high performance applications. It supports data types for working with rasters in the Scala language, as well as fast reading and writing of these data types to disk. Furthermore, GeoTrellis provides a number of operations to manipulate raster data, including cropping/warping, Map Algebra operations, and rendering operations, as well as vector to raster operations such as Kernel Density and vectorization of raster data. It also provides tools to render rasters into PNG image files or to store metadata about raster files as JSON. It aims to provide raster processing at web speeds (sub-second or less) with RESTful endpoints as well as provide fast batch processing of large raster datasets.

GeoPySpark. GeoPySpark[9] is a Python bindings library for GeoTrellis and can do many of the operations present in GeoTrellis. GeoPySpark can be integrated with other tools in the Python ecosystem, such as NumPy, sci-kit-learn, and Jupyter notebooks10. Several GeoPySpark tutorials have been developed that leverage the visualization capability of GeoNotebook, an open-source Jupyter extension that provides interactive map displays.

6 https://spark.apache.org/mllib/
7 https://flink.apache.org/
8 https://geotrellis.io/
9 https://github.com/locationtech-labs/geopyspark
10 https://jupyter.org/

2.2 PERSISTENCE LAYER TECHONOLOGIES

2.2.1 Hadoop File System

Among the Apache Hadoop's core components, there is also the *Hadoop File System (HDFS)*. HDFS or Hadoop File System is the file system in which the data is stored in a Hadoop cluster. The stored data may be structured, semi-structured or unstructured in relational database tables, JSON files or log files respectively. HDFS is designed for massive scalability, it stores unlimited amounts of data in a single platform. As the data grow, it is possible to simply add more servers to scale linearly. Furthermore, a key feature of HDFS is reliability. It automatically performs multiple copies of the data stored, letting it always available for access and protection from data loss. Built-in fault tolerance means servers can fail but a system will remain available for all workloads.

Hadoop Hue. Hadoop Hue (http://gethue.com/) is an open source user interface for Hadoop components developed by the Cloudera. The user can access Hue right from within the browser and it enhances the productivity of Hadoop developers. The main benefit of using Hue in BigDataGrapes platform is the HDFS Browser through which user can interact with the HDFS files.

In the next section, we show how to setup and run the BigDataGrapes Platform in a distributed cluster.

3 BIG DATA GRAPES PLATFORM SETUP

In this section, we detail the requirements needed to run the platform, as well as provide instructions how to set up and to launch the BigDataGrapes platform. The platform has been created re-using and customizing the software stack of the Big Data Europe (BDE[11]) platform. Besides the customization of some of the existing components, the BigDataGrapes software stack extends the BDE to better support efficient geospatial processing and distributed predictive analytics in the context of grapevine data assets and Farm Management Systems.

3.1 TECHNOLOGY REQUIREMENTS

The software stack components have been built in Docker containers to facilitate their deployment in a distributed environment. Docker[12] is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud. Docker containers are the fastest growing cloud-enabling technology and driving a new era of computing and application architecture with their lightweight approach to bundle applications and dependencies into isolated, yet highly portable application packages.

Furthermore, the BDG Platform uses Docker Swarm[13] as clustering tool that turns a group of Docker hosts into a single virtual server. Docker Swarm or simply Swarm is an open-source container orchestration platform and it is the native clustering engine for Docker. Any software, services, or tools that run with Docker containers run equally well in Swarm. Swarm turns a pool of Docker hosts into a virtual, single host. It ensures availability and high performance for an application by distributing it over the number of Docker hosts inside a cluster. Docker Swarm also allows to increase the number of container instance for the same application.

Technically, to run the platform it requires a server/cluster with the following software specifications:

- Docker engine version 18.x+, installed.
- Docker-compose version 3.x.
- Git version 2.x. (optional)

3.2 DOCKER COMPONENTS SCHEMA

Figure 2 shows the current schema of Docker components of the Big Data Grapes Platform, exhibiting also in which ports these containers are responding. Particularly, the Spark component is composed of one master node/container and many workers nodes/containers linked to the master. Similarly, the HDFS component has one namenode container and many datanodes containers.

11 https://www.big-data-europe.eu/

12 https://www.docker.com/

13 https://docs.docker.com/engine/swarm/

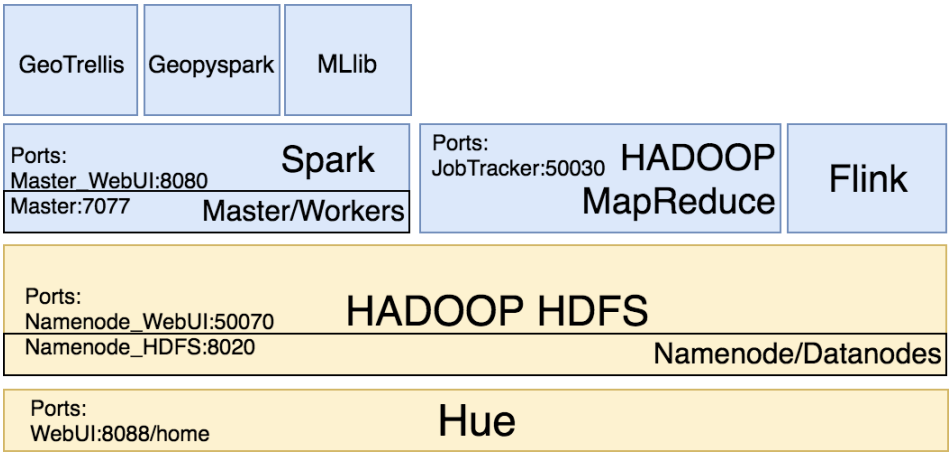

Figure 2: BigDataGrapes Docker Containers Schema

3.3 RUNNING THE DOCKER CONTAINERS

First, download or clone the git repository of the project in GitHub[14]:
To clone the project from github, use the following command:

```
$ git clone https://github.com/BigDataGrapes-EU/deliverable-D4.1.git
```

Initialization of the Big Data Grapes Platform is stored in a bash file "run-components.sh" inside the project directory.

Run the bash script:

```
$ ./run-components.sh
```

This above command will download the Docker images, build the environment according to the pre-defined configuration settings and start the Docker containers of the BigDataGrapes software stack components.

Custom configuration can be obtained by changing Hadoop cluster environmental variables using the following command.

```
$ vim ./config/hadoop.env
```

More details regarding the environment variable can be found in the Apache Hadoop official online documentation[15].Additional information about the BigDataGrapes platform can be found on GitHub[16]

14 https://github.com/BigDataGrapes

15 https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html

16 https://github.com/BigDataGrapes-EU/deliverable-D4.1/blob/master/README.md

3.4 SCALING OUT SPARK WORKERS DOCKER CONTAINERS

Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth. There are two ways to achieve this: **scaling up/down** by upgrading (increasing CPUs, memory, and storage) existing servers, or **scaling out/in** by adding more nodes/servers.

Here we provide a simple guideline to scale out Spark Workers in the BigDataGrapes platform to improve processing performance by scale out.

Run the bash script:

```
$ ./scale-out-spark-worker.sh <NUM_INSTANCES>
```

This command sets the number of containers to run for a service.

Numbers are specified as arguments in the form service=<NUM_INSTANCES>.

For example:

```
$./scale-out-spark-worker.sh 5
```

The command above launches in the BigDataGrapes platform as many spark-workers containers as specified, in this case five. This will work fine on a single host. The command creates multi-container applications but the applications are still restricted to a single host. This is because the Docker Compose does not use swarm mode to deploy services to multiple nodes. Thus, all containers are scheduled on the current node. To enable **multiple nodes deployment**, Docker Swarm can be enabled.

If Swarm mode is enabled, the following command can be used to deploy the components into the different nodes:

4 SUPPORTING FAST PROCESSING OF GEOSPATIAL RASTER DATA

In this section, we introduce some core concepts related to the processing of geospatial raster data and discuss the main components in charge of this task in the BigDataGrapes Platform. We also show two demonstrators that use the platform to process efficiently common operations applied to geospatial raster data.

Geospatial Data represents the records in a dataset that have locational information tied to them such as geographic data in the form of coordinates or spatial polygons. Geospatial data can originate from GPS devices, satellite imagery, and geotagging. The main challenge in processing this kind of data is to support the heterogeneity, for proper representation of its variety of forms, and scalability, for the processing of very large raster data.

A raster consists of a matrix of cells (or pixels) organized into rows and columns (or a grid) where each cell contains a value representing information, such as temperature and vegetation. Raster data is used in a GIS application when we want to display information that is continuous across a geospatial area. Thus, raster data are regular grids of data that have spatial properties.

In the BigDataGrapes platform, most of the core functionalities for processing raster data lay in the GeoTrellis and Spark libraries. GeoTrellis provides tools to render rasters into PNGs or to store metadata about raster files as JSON. It aims to provide raster processing at web speeds (sub-second or less) with RESTful endpoints as well as provide fast batch processing of large raster datasets.

For this report, we have made available two demonstrators performing common operations on raster data using the BigDataGrapes platform:

- Tiling Very Large Geospatial Rasters;
- Raster NDVI/NDWI layers computation;

Similar to Figure 2, Figure 3 shows the Docker components of the BigDataGrapes Platform highlighting the ones that have been used to implement the above two demonstrators. We made transparent all the other components not needed by these functionalities.

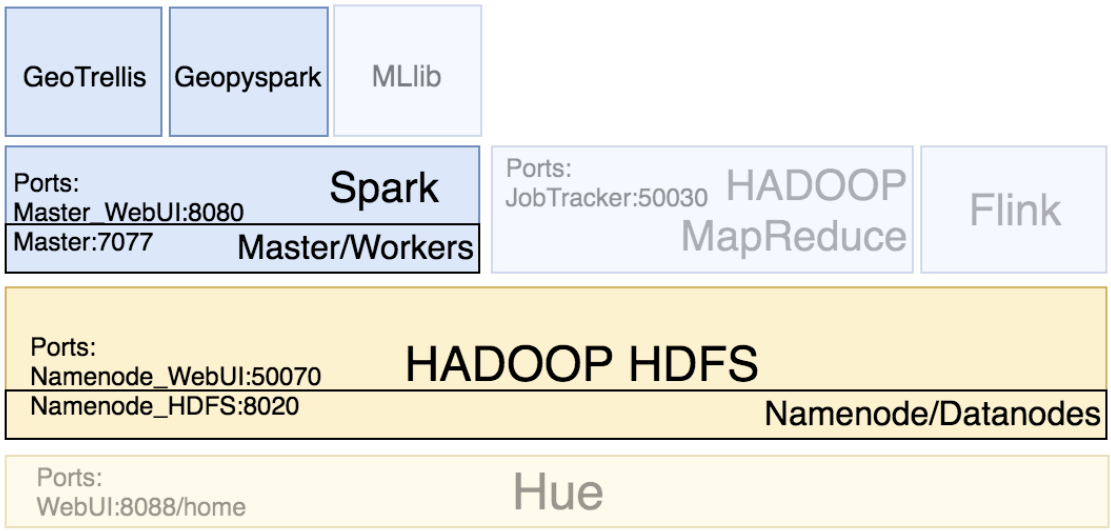

Figure 3: Docker components used to implement the demonstrators

These two demonstrators are presented respectively in the Section 4.1 and 4.2 of this report.

4.1 TILING VERY LARGE GEOSAPTIAL RASTERS

When working with rasters, a system often can operate on the grid of data separately from the spatial information. The grid of data held inside a raster is called a **tile**. Tiling allows the large raster datasets to be broken-up into manageable pieces. This approach is mostly chosen for efficiently handling very large raster datasets with limited memory and space. Thus, when operating with raster images, one benefit from subdividing rasters into tiles is the improvement in performance.

To show the appropriate detail at a certain zoom level while conserving bandwidth, images of large geographic areas are tiled to optimize delivery. When a raster has to be represented in a series of reduced/increased resolutions, a pyramid is built for that particular raster. A pyramid is a series of reduced resolution representations of the dataset, mainly used to improve the display performance of rasters when one is not working with the pixel information at full resolution. It contains a number of layers, each resampled at a more generalized level. Thus, each level of the pyramid is a resampled representation of the raster at a coarser spatial resolution.

Figure 4 shows a representation of the aforementioned process.

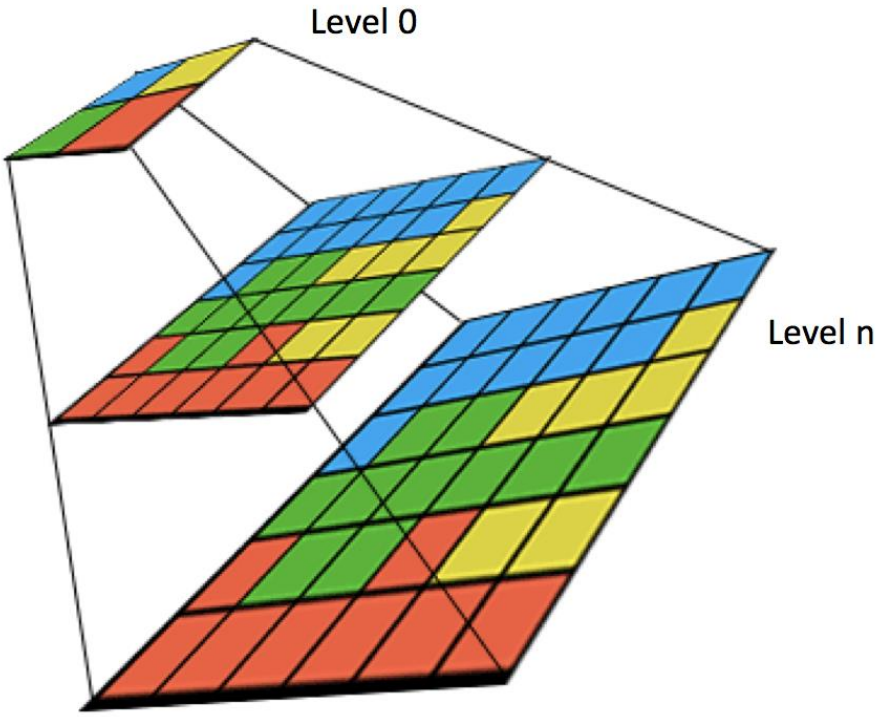

Figure 4: Tile pyramid representation.
(Adapted from: http://edndoc.esri.com/arcsde/9.2/concepts/rasters/basicprinciples/pyramids.htm)

At each higher pyramid level, the pixel size doubles resulting in four times higher pixels. The pyramid begins at the base, or level 0, which contains the original pixels of the image and proceeds toward the apex by coalescing four pixels from the previous level into a single pixel at the current level.

For this first demonstrator, the images are retrieved from HDFS component from the following directory.

```
$ hadoop fs –ls /demos/raster_tiling/input
```

In order to generate pyramided tiles, we need to resample the source image at different sizes, and then cut the resulting image into tiles.

To execute this demo, run the following bash command from the folder of downloaded git project as described in Section 3.3 :

```
$ ./run-tiling_pyramid.sh
```

The source image has full detail for the entire area at the maximum resolution.
The tiles are generated by zoom level as follow:

```
$ ls ingest/land-cover-data/tiles/<input_image_name>/
0  1  10 11 12 13 2  3  4  5  6  7  8  9
```

Each folder corresponds to a zoom level. They go up to 13 since the source image has 30m of resolution and its maximum zoom level is 13, being the closest one.

As example, Figure 5 shows a geotiff image with dimension 17370 × 11105, while the Figure 6 shows the four tiles of dimension 256x256 at the zoom level 6.

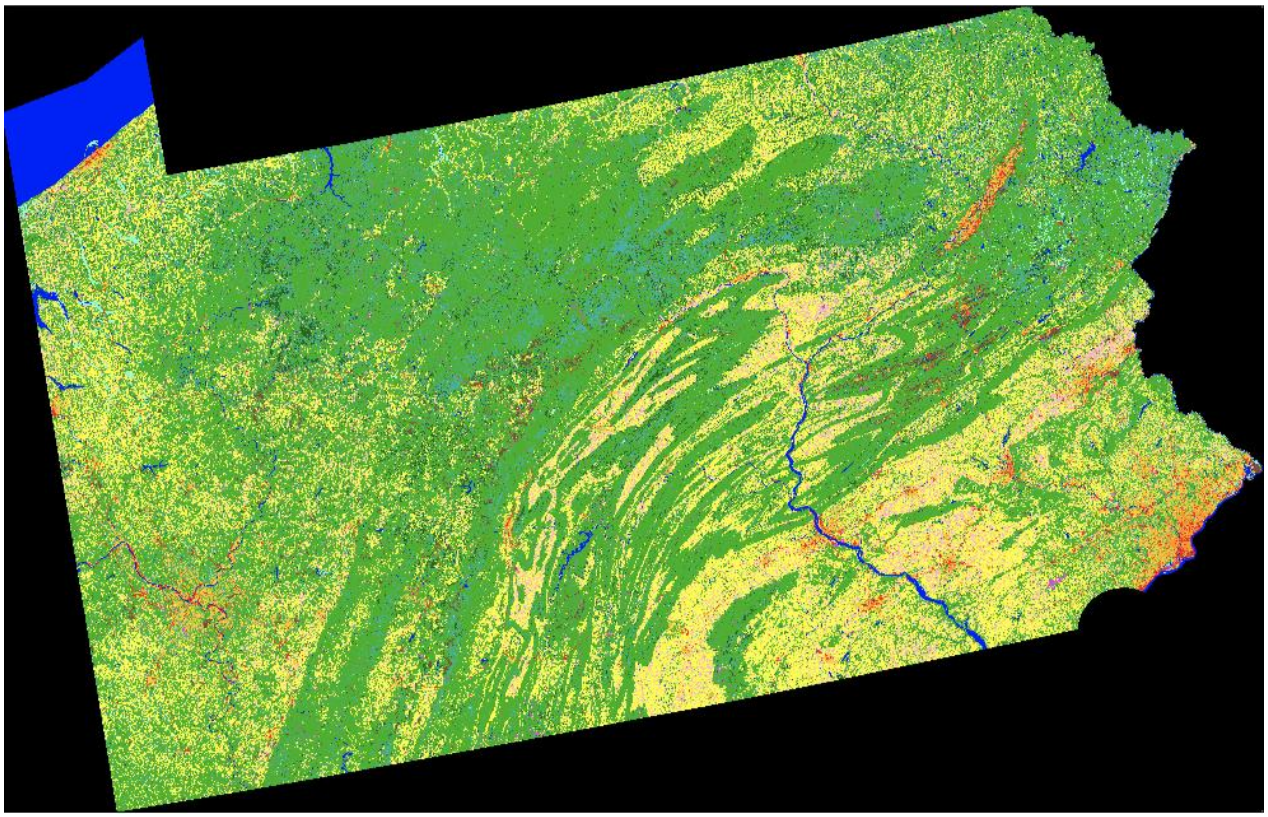

Figure 5: Geotiff image with dimension 17370 × 11105

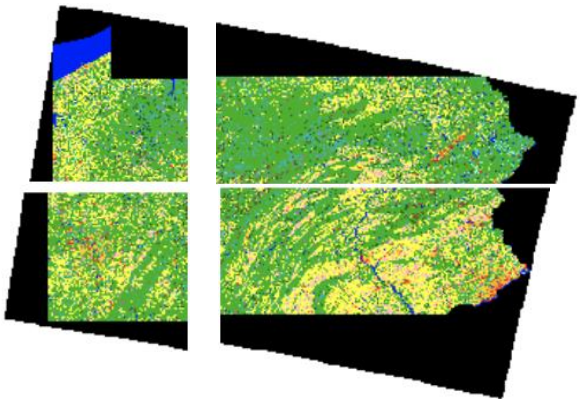

Figure 6: Tiles with dimension 256 x 256 zoom level 6

The PNG of the processed tiles for the different zoom levels can be accessed through a web browser using the port 8001 (http://<server_address>:8001) or in the following HDFS directory command:

```
$ hadoop fs –ls /demos/ndvi_ndwi/output
```

4.2 RASTER NDVI/NDWI LAYERS COMPUTATION

In our second demonstrator, we process a Landsat image into NDVI/NDWI tiles through the BigDataGrapes Platform. The Normalized Difference Vegetation Index (NDVI) (Rouse et al. 1974) is a simple vegetation index that can be used to analyse remote sensing measurements and assess whether the target being observed contains live green vegetation or not. In turn, the Normalized Difference Water Index (NDWI) (Gao 1996) we used is a suitable index for vegetation water content monitoring. The vegetation water content can be observed in the shortwave infrared part of the spectral domain. The index uses the near infrared and shortwave infrared bands of remote sensing images based on this phenomenon.

In this demo, we compute the NDVI and NDWI of a given tile. The resulting tiles are rendered as PNG and stored in the HDFS. For this task, the platform retrieves images stored in the HDFS and uses the GeoTrellis libraries to compute these indexes in a distributed fashion using Spark. The GeoTrellis libraries provide a rich set of Map Algebra operations and other tile processing features that can be used with RasterFrames via Spark's user-defined functions support.

The input images are retrieved from HDFS component from the following directory:

```
$ hadoop fs –ls /demos/ndvi_ndwi/input
```

Figure 7 shows an example of a tile, while Figure 8 shows the respective NDVI and NDWI resulting tiles.

Figure 7: Example of a tile

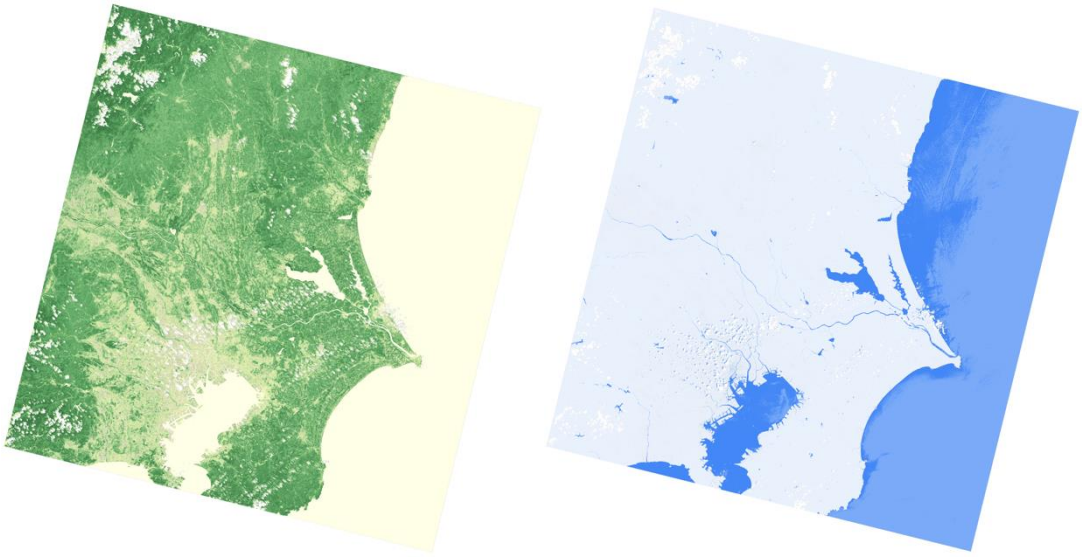

Figure 8: The NDVI (left) and NDWI (right) resulting tiles

To execute this demo, run the following bash command:

```
$ ./run-ndvi_ndwi_computation.sh
```

Both the NDVI and the NDWI PNG images can be accessed through a web browser using the port 8002 (http://<server_address>:8002).

The images can also be accessed through the HDFS in the following directory:

```
$ hadoop fs –ls /demos/ndvi_ndwi/output
```

5 SCALABILITY PERFORMANCE ANALYSIS

In this section, we aim to assess the scalability of the BDG Platform in processing remote sensing data. The BDG Platform has been designed as a distributed system to leverage the satellite image processing capabilities. The purpose of enabling distributed computing includes the ability to deal with a problem that is either bigger or longer to process by an individual computer. Thus, distributing computation across multiple machines is seen as a key approach when these machines are observed to interact with each other over the distributed network to solve a bigger problem in reasonably less time. In this way, the machines that take part in distributed computing appear as single machines to their users.

In the next subsections, we report an analysis of scalability achieved by the BDG distributed architecture when processing satellite images. The ultimate goal of our assessment is to investigate the results achieved by the BDG platform when varying the number of nodes composing the distributed infrastructure, i.e., the cluster of machines, dedicated to process the satellite images.

5.1 EXPERIMENT PROCESSING TASK

As test case of our experiments, we have chosen the process of **tiling satellite images**. As discussed in Section 4.1, the tiling of an image consists in splitting a large raster into smaller ones. It is a common strategy used to improve the management and the processing of raster data. Specifically, in our experiments, the implemented task creates tiles of 512x512 pixels.

5.2 EXPERIMENTAL SETUP

We have used as input data the Sentinels-2 (S2) satellite scenes collected over some areas of Greece corresponding to the S2 tile (ids 34SGH and 34SFH). Figure 9 shows these scenes on a map. The collected scenes are from July 2018. Specifically, the scenes are from the following dates: 02, 05, 08, 10, 13, 15, 18, 20, 23, 25, 28 and 30. For each one of the scenes, we consider four spectral bands: Band 2 Blue (10m resolution), Band 3 Green (10m resolution); Band 4 Red (10m resolution) and Band 8 Near infrared 1 (10m resolution).

The images can be collected from an online service called Remote Pixel[17].

17 https://search.remotepixel.ca/#8.16/38.016/23.172

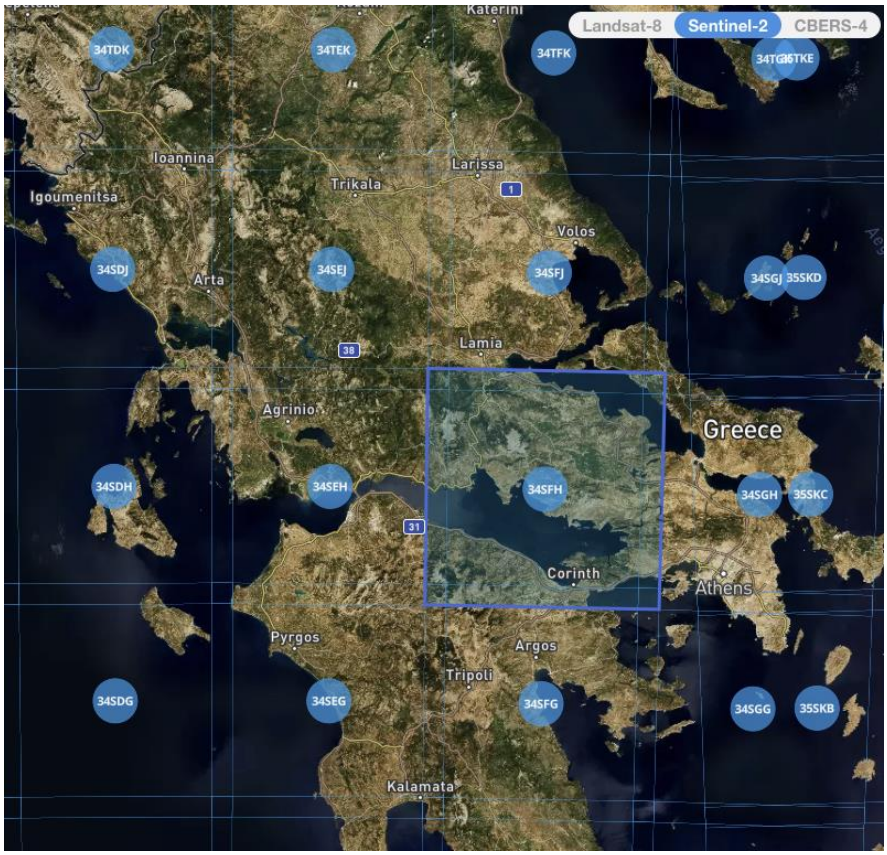

Figure 9: 34SGH and 34SFH scenes tiles

Our BDG platform exploits a cluster of four spark nodes (node 1, node 2, node 4 and node 5). In Table 1, we report the configuration of the nodes in terms of cores and memory.

Table 1: Cluster Nodes Configuration

Node Name	CPU (Cores)	Memory (GB)
Node 1	8	32
Node 2	8	24
Node 4	8	24
Node 5	8	32

The Cluster uses HDFS to store the satellite images using a replication factor of 4. Thus, the images are replicated in all the nodes.

For the processing of each scene and for all the aforementioned days of July 2018, we create a Spark session so that we have as much as possible scenes being processed in parallel. Also, we recall that each scene has 4 image bands (B03.tif, B03.tif, B04.tif and B08.tif), thus each session processes a total of four satellite images. Specifically, we report the results for two different settings of scene processing: (1) 50 and (2) 60 scene processing sessions.

5.3 EXPERIMENTAL RESULTS

Metrics. To evaluate the results, we use the following scalability metrics: throughput, latency and elapsed time (min, max and total).

- Throughput. The actual number of images that get processed.
- Latency. How much time it takes to process an image. It is the inverse of throughput.
- Elapsed time. It is simply the amount of time that passes from the beginning of the reading of the image from the HDFS and the end of the tilling processing.

Experiment 1. The first experiment employs a processing of fifty scenes in parallel. Table 2 reports the results of the experiment.

Table 2: Processing Statistics for the 50 Spark parallel sessions

#nodes	#cores	#sessions	throughput (/minute)	latency (/minute)	Scene elapsed time (min)	Scene elapsed time (max)
1	16	50	24.34	0.041	2.38	8.18
2	32	50	43.86	0.023	1.96	4.53
3	48	50	54.96	0.018	1.32	3.62
4	64	50	62.12	0.016	1.21	3.13

The results show that the throughput increases linearly when we add more nodes to the cluster. Also, we observe that the elapsed time to process all the images reduces to almost a third of the time when using a single node. Figure 10 shows the throughput result of Experiment 1 as a line plot.

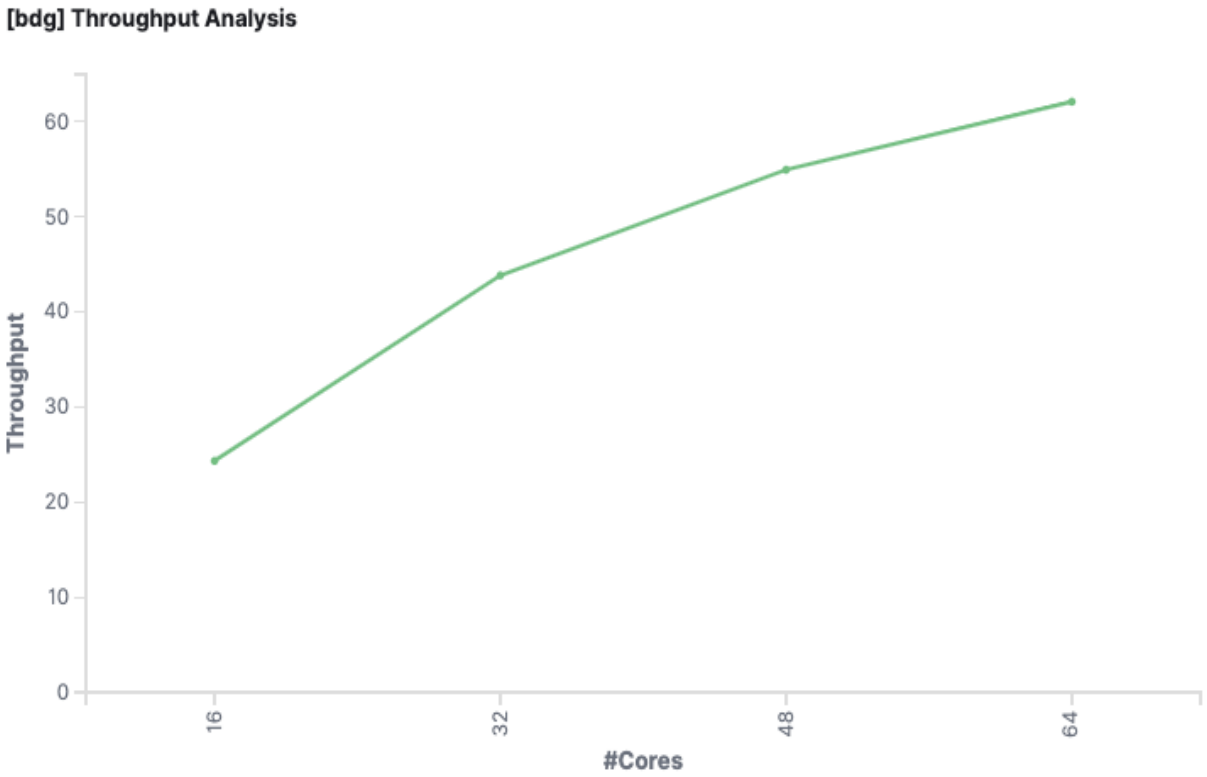

Figure 10: Throughput analysis considering 50 Spark parallel sessions

Experiment 2. The second experiment employs a processing of sixty scenes in parallel. Table 3 reports the results of the experiment.

Table 3: Processing Statistics for the 60 Spark parallel sessions

#nodes	#cores	#sessions	throughput (/minute)	latency (/minute)	Scene elapsed time (min)	Scene elapsed time (max)
1	16	60	25.17	0.039	2.461	9.505
2	32	60	45.26	0.022	2.402	5.286
3	48	60	55.66	0.018	2.129	4.300
4	64	60	67.08	0.015	2.352	3.564

Again, we can see that the throughput linearly increases when we increase the number of nodes in the cluster. We also observe that the use of a higher number of parallel sessions increases the throughtput. For instance, when using four nodes, we had an improvement from 62.122 imgs/minute (50 parallel sessions) to 67.075 imgs/minute (60 parallel sessions). Figure 10 shows the result of the Experiment 2 in terms of throughput as a line plot. We conclude that the use of the BDG distributed architecture allows to increase the throughput almost linearly with the number of available nodes.

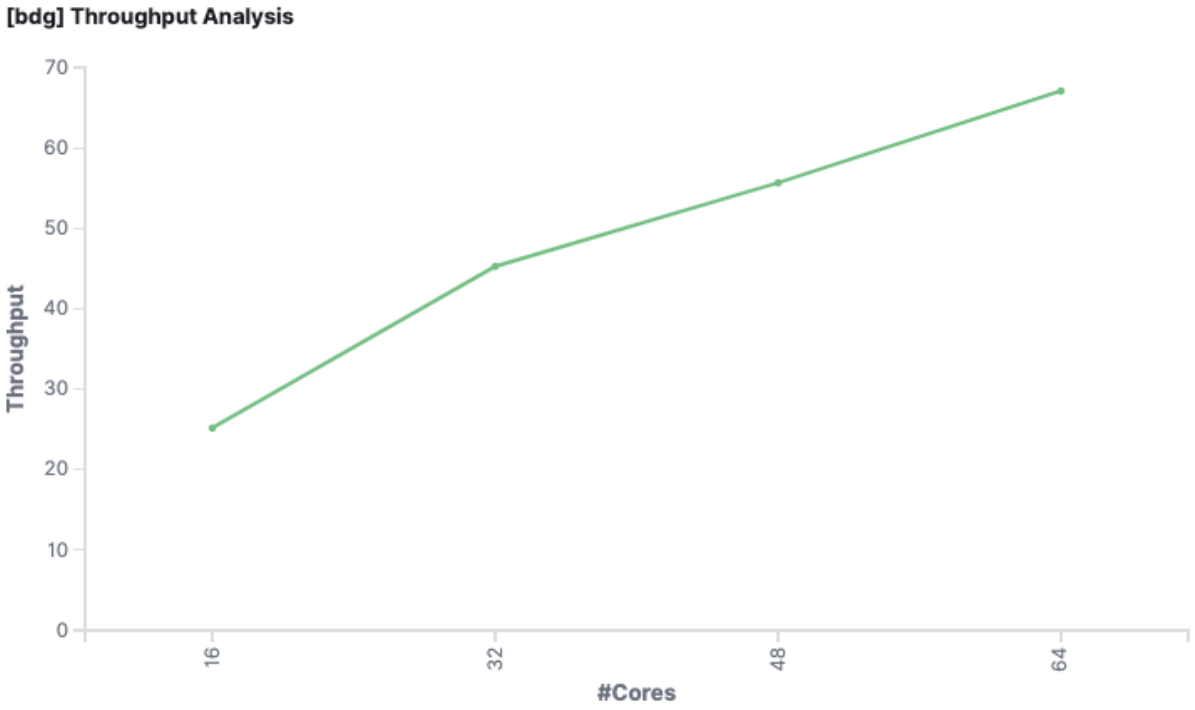

Figure 11: Throughput results 60 Spark parallel sessions

6 CLUSTER PERFORMANCE TOOL

The BDG Cluster Performance Tool (CPT) is an application developed to monitor the status of the cluster, i.e., the spark nodes, dedicated to process satellite images. CPT has been designed to visually provide fast insight of the performance of the satellite image processing pipeline. On the front-end side, the tool has a Web application that runs a dashboard built upon *Kibana*[18]. Moreover, on the back-end side, the tool uses: i) an API that collects the performance of the nodes composing the cluster and ii) an *ElasticSearch*[19] database to store the performance statistics of the nodes.

The aim of CPT is to help to quickly identify whether the BDG Platform is experiencing a performance bottleneck during the processing of satellite images. Some common performance problems that CPT can help to identify include: CPU bottlenecks (what images bands and scenes are consuming the most of the CPU resources) and memory bottlenecks (which queries are performing the most of the I/O). Figure 12 provides a general view of CPT.

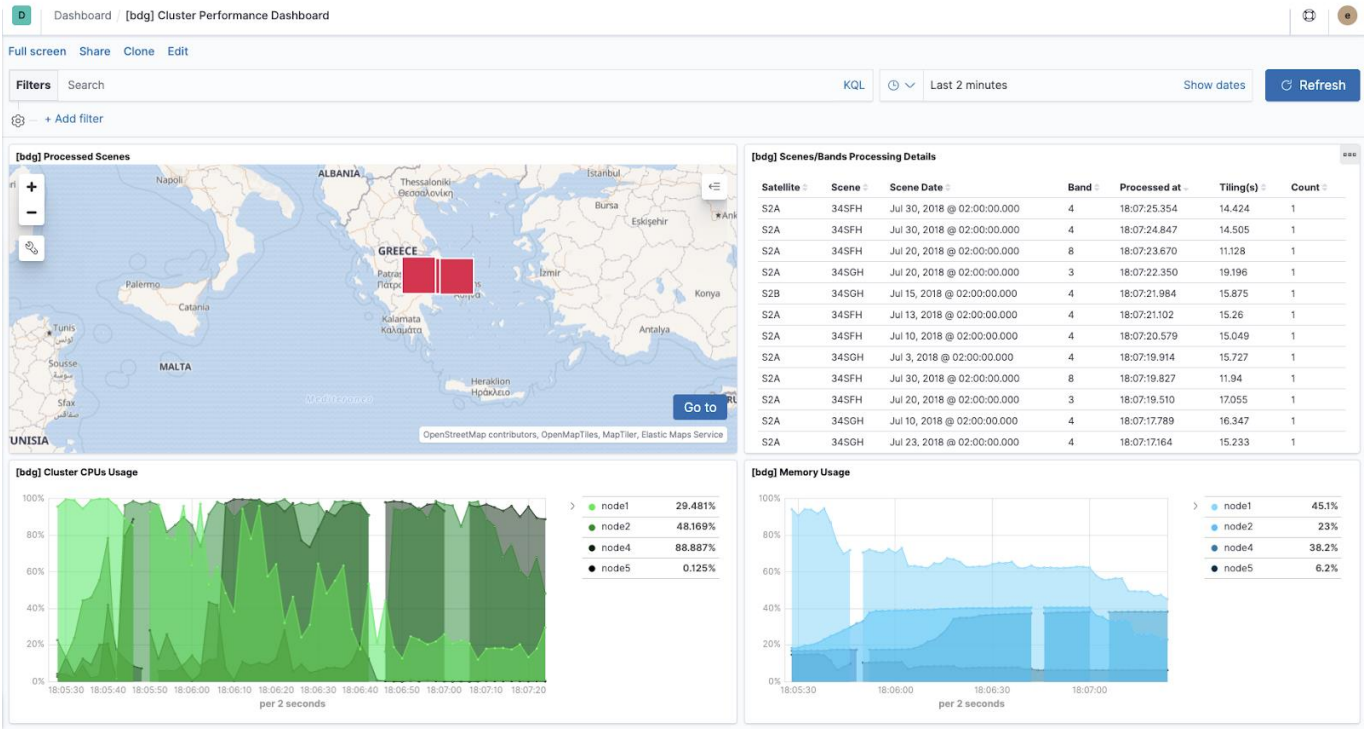

Figure 12: Performance Tool Dashboard

We now discuss more in details the components and functionalities of CPT.

6.1. TOOL SETUP AND COMPONENTS

The Figure 13 shows an overview of the components and their interactions.

18 https://www.elastic.co/products/kibana
19 https://www.elastic.co/

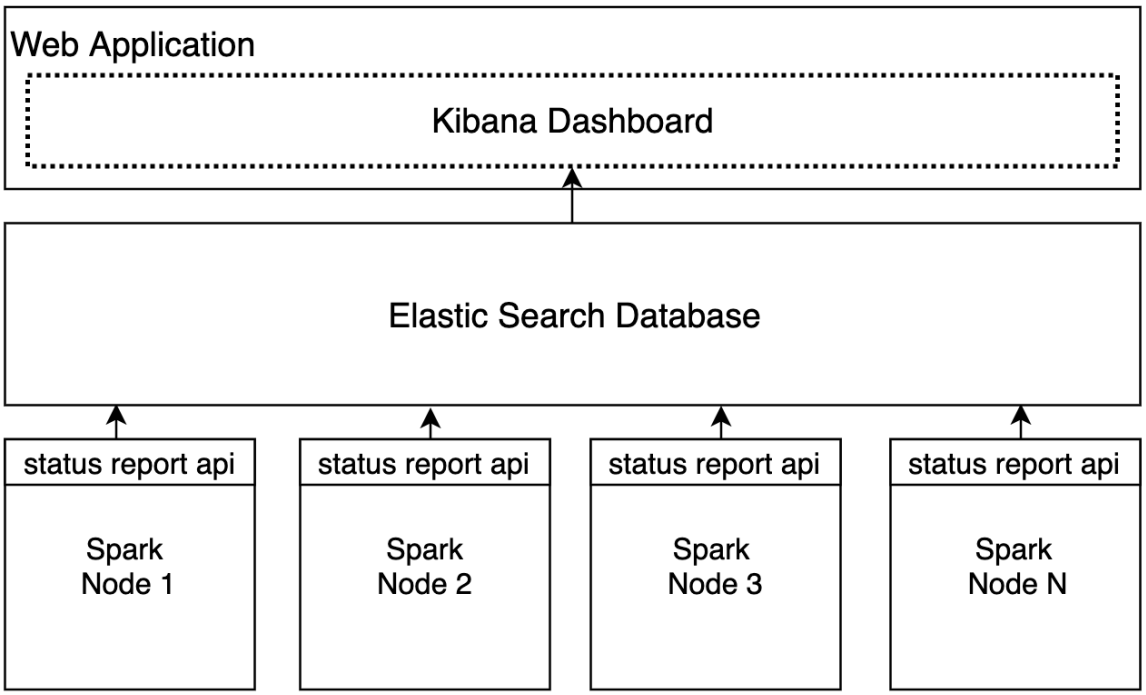

Figure 13: Performance Tool Components

The Tool is composed by the following components:

1) Spark Nodes: They are spark node workers of the BDG Platform dedicated to process the satellite images using the GeoPySpark library (Section 2.1.4). All workers are coordinated by the same spark master node.
2) Real-Time Status API: A real-time service reports the cluster nodes status. This component is useful mainly for system monitoring of running processes. This service uses the Psutil library[20] (process and system utilities) , which is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python. Psutil currently supports both 32-bit and 64-bit architectures for the following platforms: Linux, Windows, macOS, FreeBSD, OpenBSD, NetBSD, Sun Solaris and AIX.
3) Elastic Search Database: Each spark node reports its status to an ElasticSearch database. The Elastic Search Databases have interesting features such as: automatic node recovery, high availability and horizontal scalability.
4) Web Application: A web application runs a Kibana dashboard as an iframe. This allows a custom visualization of the web page while keeping the full user-interaction functionalities of a Kibana dashboard.

The source code of the Cluster Performance Tool Components can be found on GitHub[21]

Furthermore, a demo video22 has been prepared for documenting the Cluster Performance Tool.

20 https://pypi.org/project/psutil/

21 https://github.com/BigDataGrapes-EU/deliverable-D4.1/cp_tool/

22 https://www.dropbox.com/s/rgtdc0032plhrp7/demo_video_bdg2.mov?dl=0

6.2. CPT FUNCTIONALITIES

Here we describe the main functionalities of the CPT dashboard. CPT is divided into the following sub-reports: a) CPU Cluster Utilization; b) Memory Cluster Utilization; c) Visualization on Map; and d) Image Processing Details.

a) **CPU Cluster Utilization.** Through this functionality it is possible to visualize the average usage of the CPU cores of each work node that composes the cluster. This visualization helps to identify what are the nodes currently consuming high processing resources in the cluster. Figure 14 shows the current and the last two minutes of CPU usage per node, for a cluster composed by four nodes.

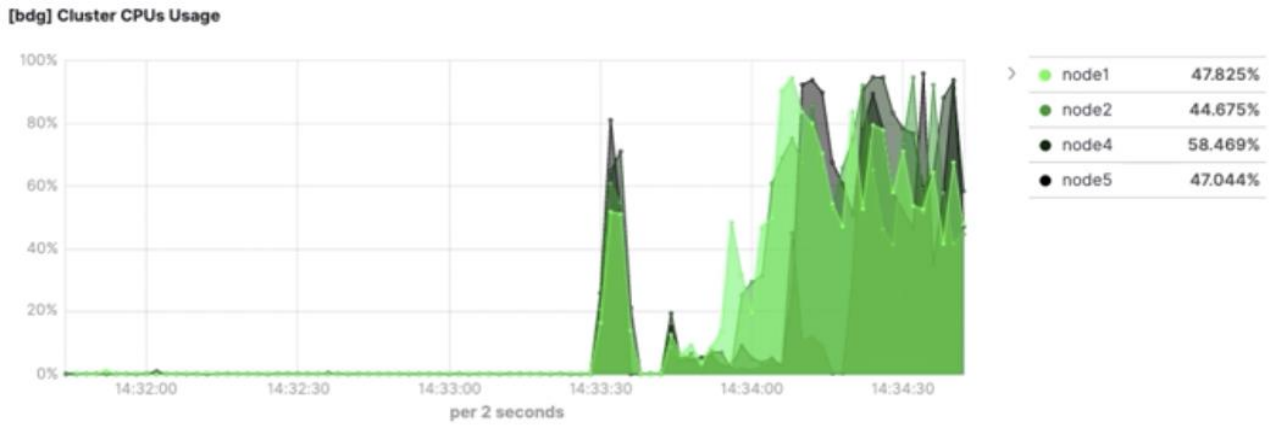

Figure 14: Cluster CPUs Usage Report

b) **Memory Cluster Utilization.** This functionality assists the cluster administrator for an easy monitoring of the memory usage of each work node of the cluster. Similar to Figure 14, Figure 15 shows the current and the last two minutes of memory usage per node.

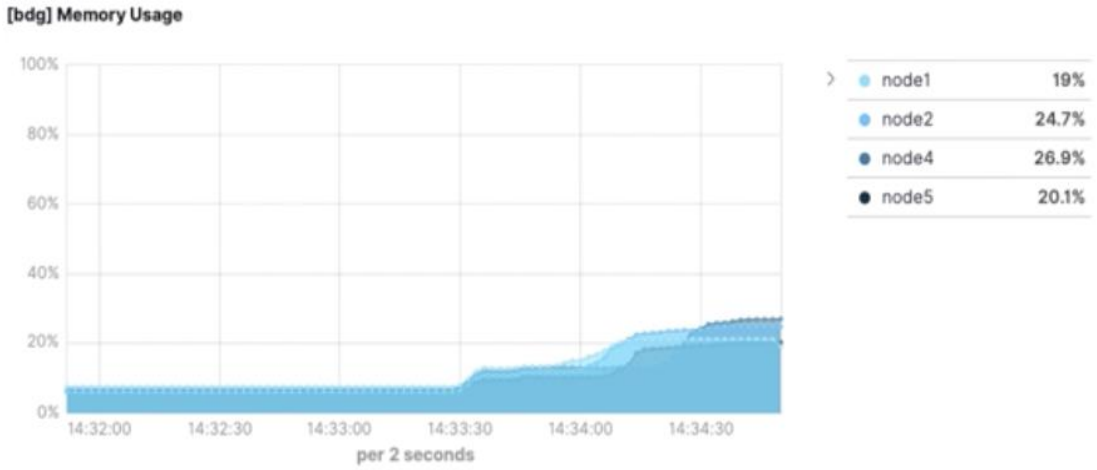

Figure 15: Memory Cluster Utilization Report

c) **Visualization on map.** As the parcel images are processed by the cluster, it is possible to see on the map the area covered by the parcel image. Figure 16 shows an example of a parcel in Greece that has just been processed by the platform. The user can click on the area of the image to get a few more information. To plot the image area, we store in ElasticSearch the geoinformation of the satellite images which are present in the image metadata.

Figure 16: Scene Processed Map Visualization

d) **Image Processing details.** The dashboard reports in tabular format the details of the processing of satellite images. The following information about the last processed images are reported: the satellite source, scene, date, band, time that the processing has been concluded and elapsed time. Figure 17 illustrates an example of an Image Processing Details report.

[bdg] Scenes/Bands Processing Details

Satellite	Scene	Scene Date	Band	Processed at	Tiling(s)
S2B	34SGH	18/07/2018	8	14:35:01	7.971
S2A	34SFH	20/07/2018	8	14:35:03	8.327
S2A	34SFH	10/07/2018	8	14:35:03	8.482
S2A	34SGH	23/07/2018	8	14:35:01	8.581
S2B	34SGH	08/07/2018	8	14:34:59	8.832
S2B	34SGH	18/07/2018	4	14:34:53	8.887
S2A	34SFH	13/07/2018	8	14:35:03	9.001
S2A	34SGH	03/07/2018	8	14:34:51	9.026
S2A	34SFH	20/07/2018	4	14:34:54	9.09
S2A	34SFH	10/07/2018	4	14:34:54	9.217
S2A	34SGH	23/07/2018	4	14:34:53	9.584
S2B	34SGH	05/07/2018	4	14:34:56	9.654

Figure 17: Scene Processing Details Report

7 EXPLOITATION IN A REAL-WORLD SATELLITE PROCESSING INFRASTRUCTURE

GEOCLEDIAN's satellite processing infrastructure provides field-based satellite image product timeseries (e.g. vegetation indexes) to the BDG stack and allows thus to provide field-based vegetation monitoring to the partners in the frame of the project. The system relies on open satellite data that is downloaded, stored, processed and then the image time series per parcel (field) to be monitored are extracted. The basic workflow is shown in Figure 18. The architecture of the old system was monolithic, centered around a system running only on one machine which led to limits concerning the data processing capacity and scalability.

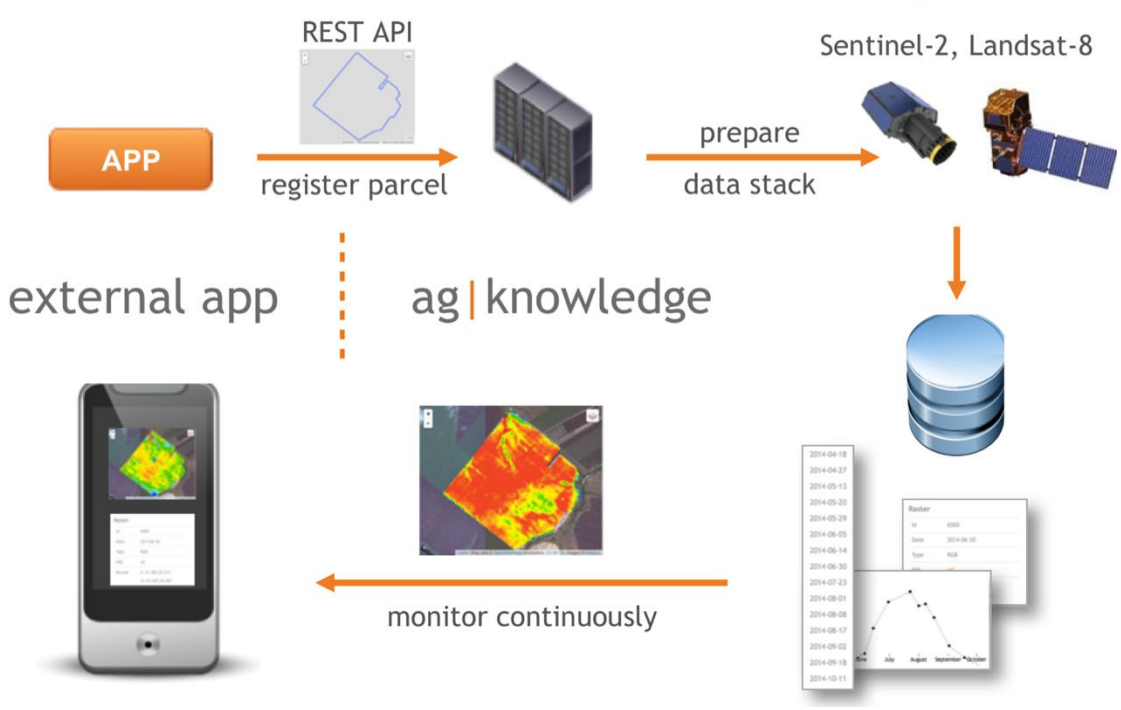

Figure 18: The basic workflow in Geocledian's satellite processing infrastructure, ag|knowledge

Performance tests and an analysis of GEOCLEDIAN's satellite data processing system have been performed to identify bottlenecks concerning the data processing capacity and scalability. Then, an architectural refactoring of the initial system into modular subsystems has been performed to enable scalability and processing optimizations. More specifically these tasks have been carried to tackle the data processing optimization and scaling issues:

- Performance monitoring and analysis of the system;
- Identification and analysis of performance bottlenecks;
- Architectural refactoring of the initial system into modular subsystems to enable scalability and processing optimizations;
- Development of scalable platform components.

The following use case diagram (Figure 19) illustrates the main processing use cases inside the system:

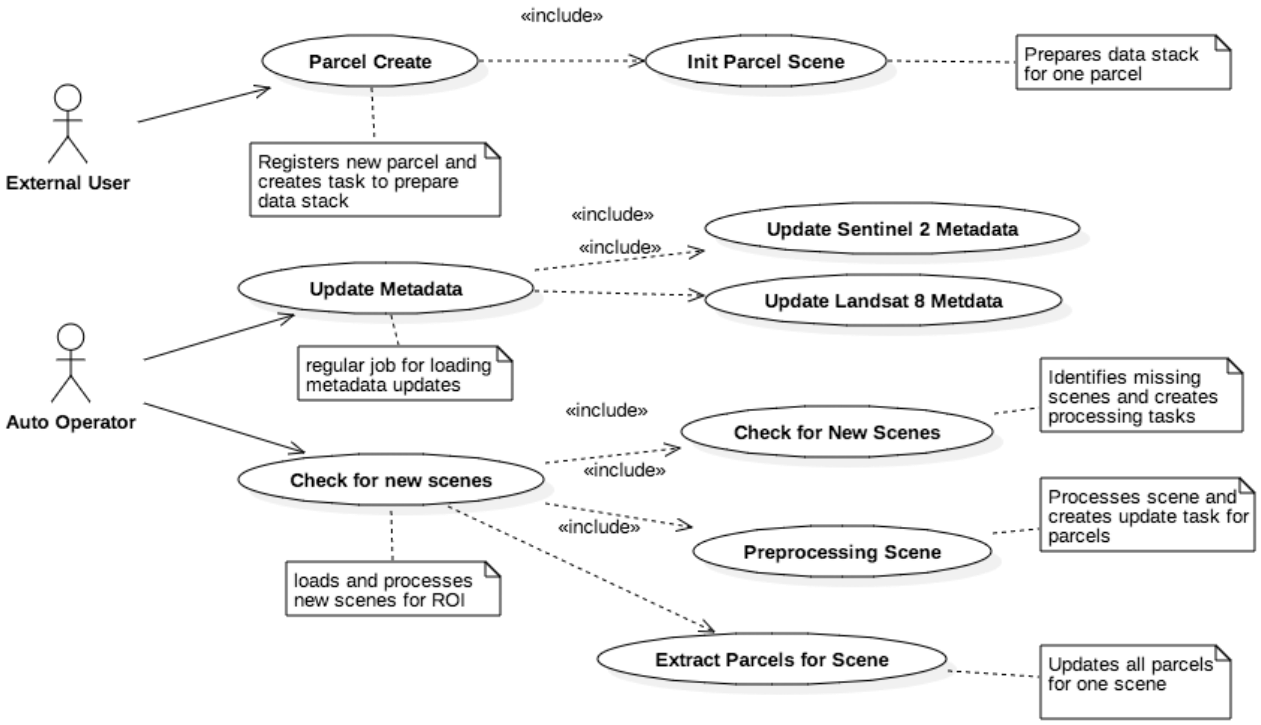

Figure 19: Use case diagram illustrating the main use cases happening inside the system that were analyzed

The main bottlenecks identified are the satellite scene preprocessing comprising amongst other steps the tiling and the parcel extraction which enables the generation of field-based vegetation index products.

The architectural refactoring of the initial system into modular subsystems has been performed to enable scalability and processing optimizations. In the new system, the critical modules that needed an improved scalability capability are isolated now so that they can be replaced with more scalable modules at a later stage. A high-level architecture of the new system that is already in operational use within the project is shown in Figure 20.

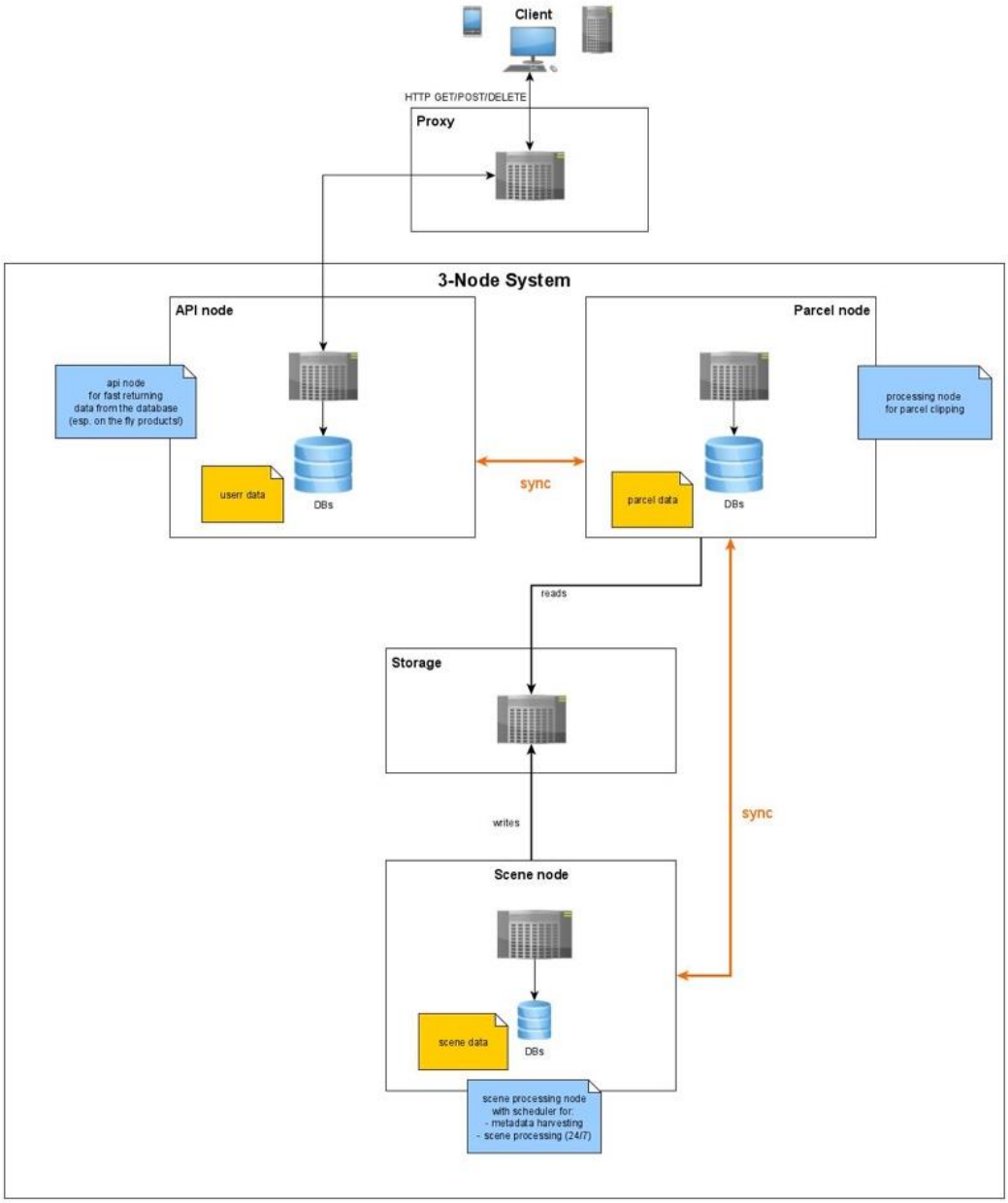

Figure 20: A high-level architecture of the refactored system

The main benefit of the new system is the modularity that allows to scale some parts of the system independently of the rest of the system. Another short-term benefit is the already improved capability to process more scenes and parcels in parallel. This can be shown with the number of concurrent tasks possible now (Table 4).

Table 4: The scalability on the old and new system (we assume four core machines for comparison).

	Pre-BDG system	Current system
Max. number of concurrent parcel extraction tasks	2	6
Max. number of concurrent scene processing tasks	2	6

The new modular system allows GEOCLEDIAN to speed up the processing of satellite images by 3x. Moreover, with the new system it is possible to process newly added regions of interest of the project partners more efficiently. Additionally, it is worth mentioning that the parcel extraction tasks and the scene processing tasks are on independent machines now, so they do not have to compete for CPU and memory resources.

GEOCLEDIAN worked with CNR on the implementation of the above described demos performing scalable operations on geospatial raster data using the Spark-based GeoTrellis geographic data processing engine provided by the BigDataGrapes platform. At a later stage parts of these demonstrators will be integrated in the satellite data processing system.

8 SUMMARY

This accompanying document for deliverable D4.1 Methods and Tools for Scalable Distributed Processing describes the main mechanisms and tools used in the BigDataGrapes platform to support efficient processing of large datasets in the context of grapevine-related assets.

The BDG software stack designed provides efficient and fault-tolerant tools for distributed processing, aiming at providing scalability and reliability for the applications. The BDG software stack and its underlying components have been built upon core tools and frameworks used for the processing of very large datasets, ensuring that both research and development work can be based on solid foundations and that the ultimate outcomes can be directly applicable to production level assets.

This report presents two demonstrators, both compliant with distributed computation. The first one performs tiling operation on very large geospatial rasters while the second one exploits the BDG platform to compute the NVDI and NDWI of the given tiles.

We also provide a description of the BDG Cluster Performance Tool (CPT), an application developed to monitor the status of the cluster, i.e., the spark nodes, dedicated to process satellite images. On the front-end side, the tool has a Web application that runs a dashboard built upon *Kibana*[23]. Moreover, on the back-end side, the tool uses: i) an API that collects the performance of the nodes composing the cluster and ii) an *ElasticSearch*[24] database to store the performance statistics of the nodes.

Finally, we report the lessons learned from the exploitation of the BDG data processing tools in a real-world satellite processing architecture. We report how GEOCLEDIAN moved from a monolithic architecture to a modular one after an analysis aiming at identifying bottlenecks and congestion. The new modular architecture allows GEOCLEDIAN to speed up the processing of satellite images by 3x. Moreover, the new architecture will be used to deploy the new BDG data processing components presented in this deliverable for a fast tiling and fast index computation from satellite images.

23 https://www.elastic.co/products/kibana
24 https://www.elastic.co/

9 REFERENCES

Gao, Boi-Cai, 1996. NDWI – A Normalized Difference Water Index for Remote Sensing of Vegetation Liquid Water from Space, Remote Sensing of Environment, Volume 58, Issue 3, pp 257-266.

Rouse, J.W., Haas Jr., R.H., Schell, J.A., Deering, D.W., 1974. Monitoring vegetation systems in the Great Plains with ERTS. In: Third ERTS-1 Symposium, NASA, Washington, DC, pp. 309–317.