Ref. Ares(2018)4969492 - 27/09/2018


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** | 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** | 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) |
| **DATE OF DELIVERY (SUBMITTED)** | 28 September 2018 (M9) |
| **VERSION \| STATUS** | 1.0 \| Final |
| **NATURE** | DEM (Demonstrator) |
| **DISSEMINATION LEVEL** | PU (Public) |
| **AUTHORS (PARTNER)** | Nicola Tonellotto (CNR), Franco Maria Nardini (CNR), Raffaele Perego (CNR), Vinicius Monteiro de Lira (CNR), Ida Mele (CNR), Matteo Catena (CNR), Cristina Muntean (CNR) |
| **CONTRIBUTORS** | Florian Schlenz (Geocledian), Toni Frank (Geocledian), Johannes Sommer (Geocledian) |
| **REVIEWER** | Milena Yankova (ONTOTEXT) |

| 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) |

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

## 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.

Then 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.

Finally, 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 indexes. The first index is the normalized difference vegetation index (NDVI), a graphical indicator that assess at what degree the target being observed contains live green vegetation or not. The second index is instead the Normalized Difference Water Index (NDWI), most appropriate for water body mapping.

## TABLE OF CONTENTS

## LIST OF FIGURES

# 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.

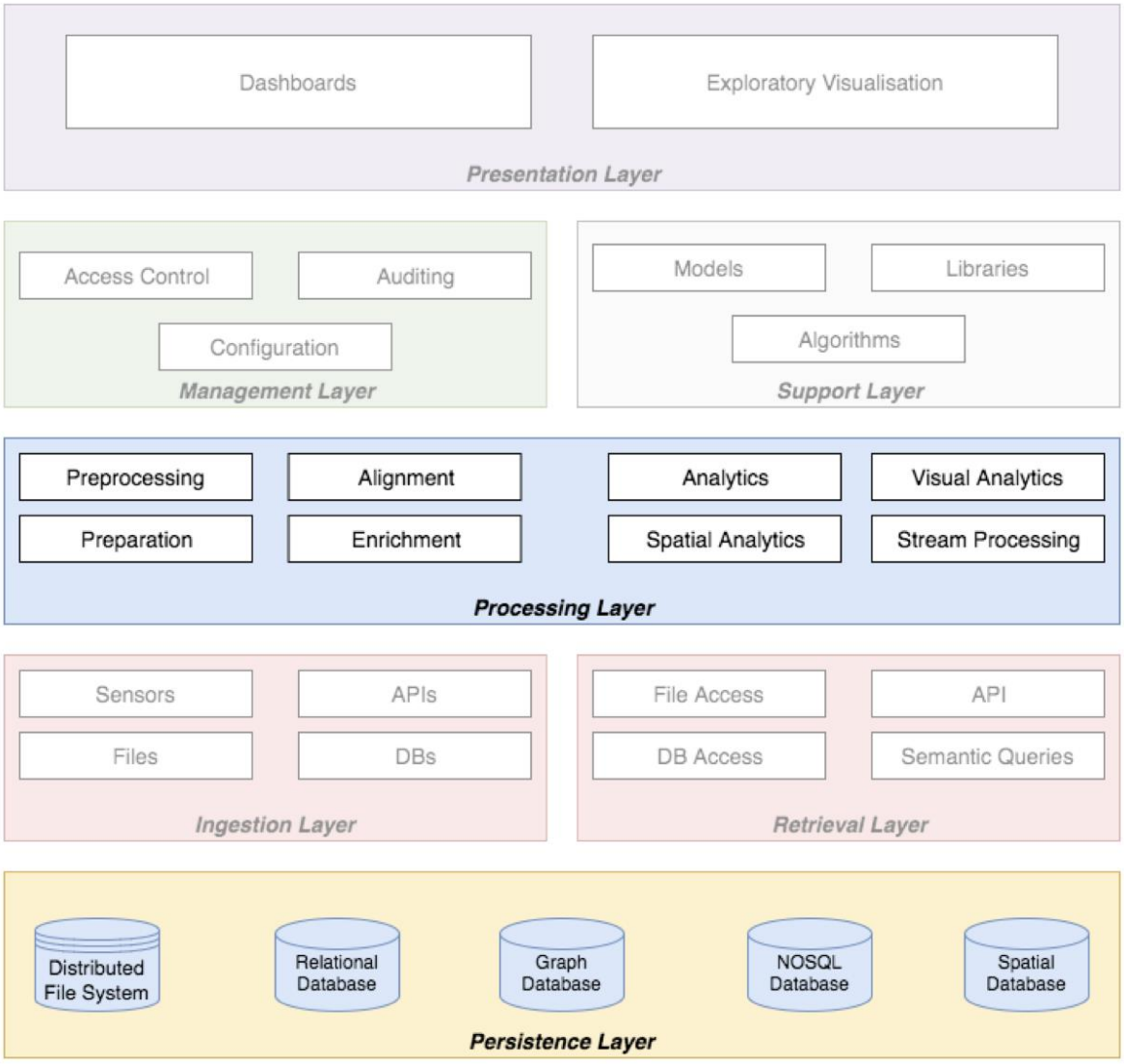


Figure 1: BigDataGrapes Architecture Layers

In this first 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. Finally, Section 5 concludes this document.

# 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 (http://hadoop.apache.org/) 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 is basically 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 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 (https://spark.apache.org/) 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 as interactive queries on a graph and streaming machine learning) that were not possible with previous systems.

**MLlib.** MLlib 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 (https://flink.apache.org/). The 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 (https://geotrellis.io/) is a geographic data processing engine that uses Spark to work with raster data for high performance applications. It support 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 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.

**GeoPySpark.** GeoPySpark (https://github.com/locationtech-labs/geopyspark) 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 notebooks. Several GeoPySpark tutorials have been developed that leverage the visualization capability of GeoNotebook, an open-source Jupyter extension that provides interactive map displays.

## 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, https://www.big-data-europe.eu/) 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 (https://www.docker.com/) 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 (https://docs.docker.com/engine/swarm/) as clustering tool that turns a group of Docker hosts into a single virtual server. Docker Swarm 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.

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

- Docker engine version 18.x+, installed.
- Docker-compose version 2.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.

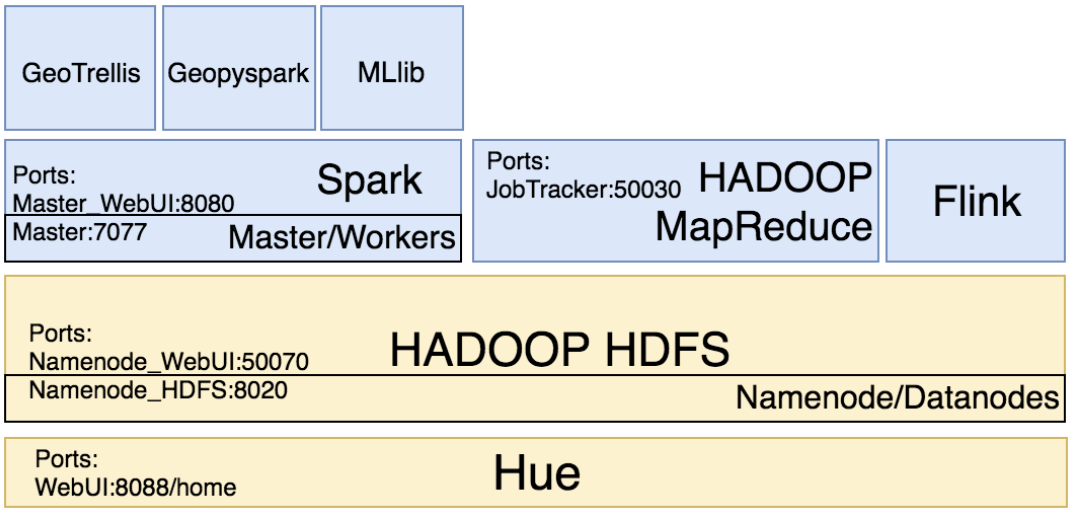


Figure 2: BigDataGrapes Docker Containers Schema

## 3.3 RUNNING THE DOCKER CONTAINERS

First, download or clone the git repository of the project in (https://github.com/BigDataGrapes):
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 here:
https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html

Additional information about the BigDataGrapes platform can be found here:
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 immediately launches in the BigDataGrapes platform as many spark-workers containers as specified, in this case 5.

# 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 operation 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 NVDI/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.

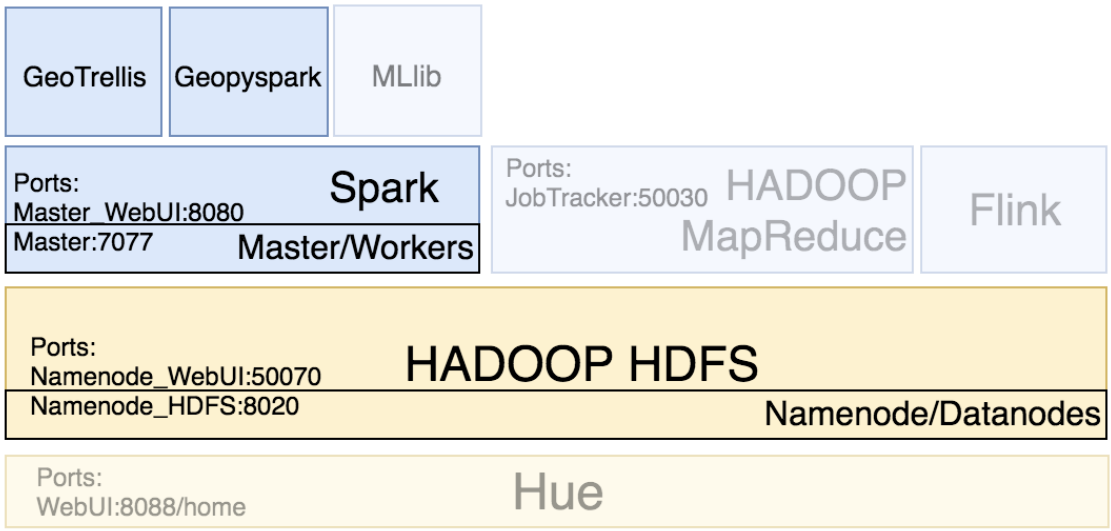


**Figure 3: Docker components used to implement the demonstrators**

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.

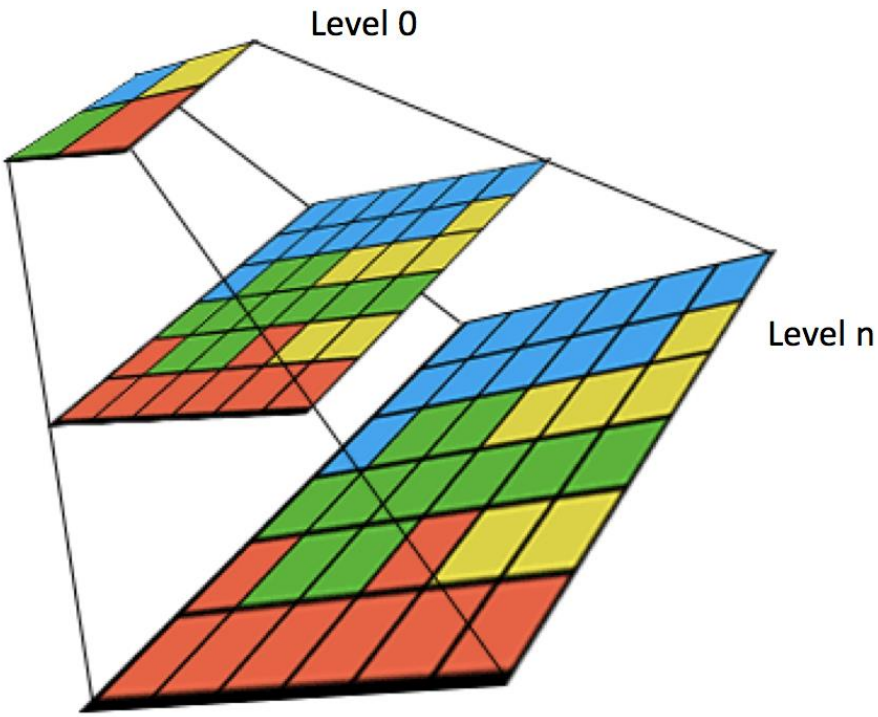


**Figure 4: Tile pyramid representation.**
**(Adapted from: http://edndoc.esri.com/arcsde/9.2/concepts/rasters/basicprinciples/pyramids.htm)**

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.

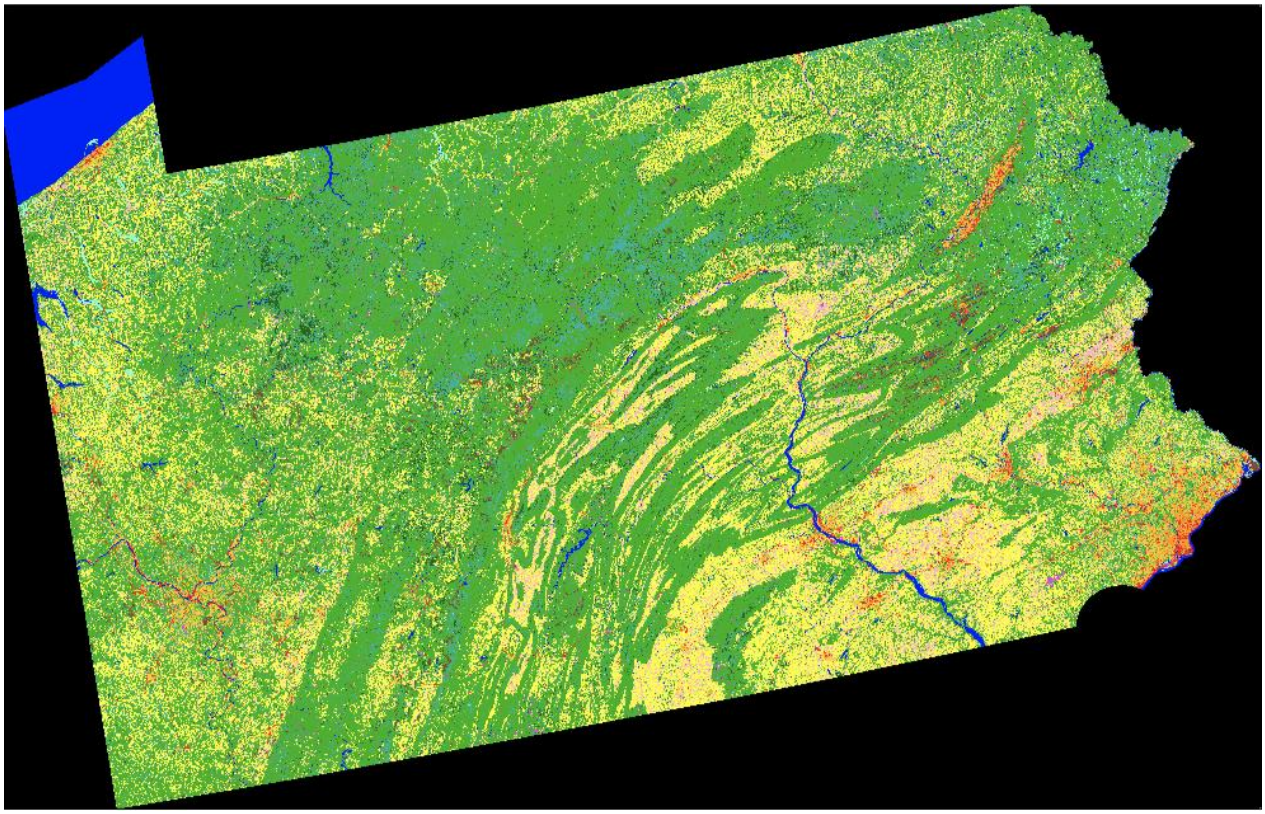

Figure 5: Geotiff image with dimension 17370 × 11105

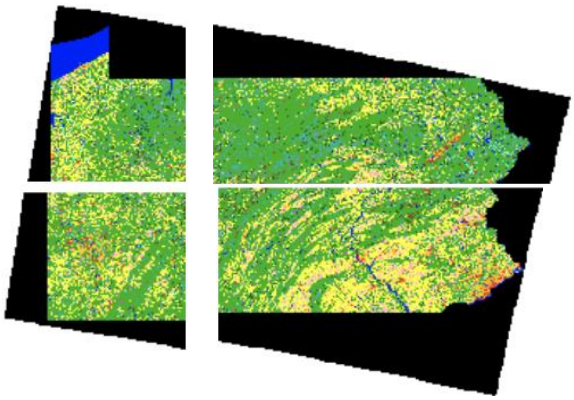

Figure 6: Tiles with dimension 256 x 256 zoom level 6

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 NVDI/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) is a simple graphical indicator 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) is a suitable index for water body mapping. The water body has strong absorbability and low radiation in the range from visible to infrared wavelengths. The index uses the green and near infra-red bands of remote sensing images based on this phenomenon. The NDWI can capture the water information effectively in most cases, but it is sensitive to built-up land and can results in over-estimated water bodies.

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**

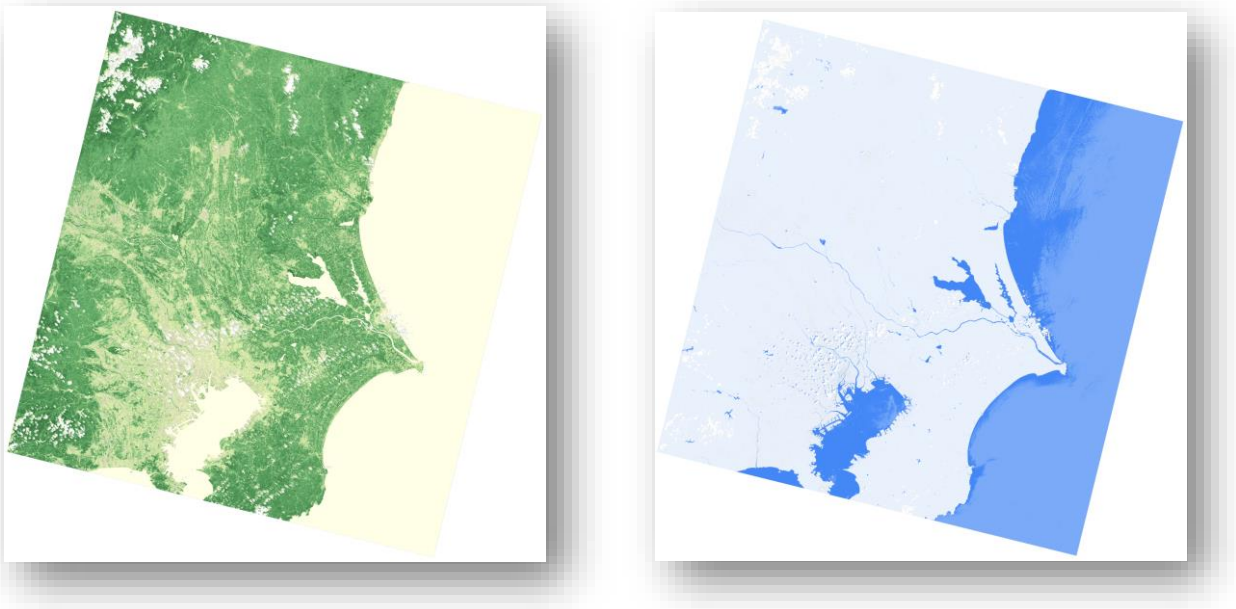

Figure 8: The NDVI (left) and NDWI (right) resulting tiles

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 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.

For this purpose, this report presents two demonstrators, both compliant with distributed computation. The first one performs tiling operation on very large geospatial rasters. In turn, the second demo computes the NVDI and NDWI of the given tiles.