PROCESSING OF SATELLITE DATA IN THE CLOUD

JERZY PROFICZ AND KRZYSZTOF DRYPCZEWSKI

Gdansk University of Technology
Academic Computer Centre TASK
Narutowicza 11/12, 80-233 Gdansk, Poland

(received: 22 August 2017; revised: 22 September 2017;
accepted: 29 September 2017; published online: 12 October 2017)

Abstract: The dynamic development of digital technologies, especially those dedicated to devices generating large data streams, such as all kinds of measurement equipment (temperature and humidity sensors, cameras, radio-telescopes and satellites – Internet of Things) enables more in-depth analysis of the surrounding reality, including better understanding of various natural phenomenon, starting from atomic level reactions, through macroscopic processes (*e.g.* meteorology) to observation of the Earth and the outer space. On the other hand such a large quantitative improvement requires a great number of processing and storage resources, resulting in the recent rapid development of Big Data technologies. Since 2015, the European Space Agency (ESA) has been providing a great amount of data gathered by exploratory equipment: a collection of Sentinel satellites – which perform Earth observation using various measurement techniques. For example Sentinel-2 provides a stream of digital photos, including images of the Baltic Sea and the whole territory of Poland. This data is used in an experimental installation of a Big Data processing system based on the open source software at the Academic Computer Center in Gdansk. The center has one of the most powerful supercomputers in Poland – the Tryton computing cluster, consisting of 1600 nodes interconnected by a fast Infiniband network (56 Gbps) and over 6 PB of storage. Some of these nodes are used as a computational cloud supervised by an OpenStack platform, where the Sentinel-2 data is processed. A subsystem of the automatic, perpetual data download to object storage (based on Swift) is deployed, the required software libraries for the image processing are configured and the Apache Spark cluster has been set up. The above system enables gathering and analysis of the recorded satellite images and the associated metadata, benefiting from the parallel computation mechanisms. This paper describes the above solution including its technical aspects.

Keywords: Apache Spark, Satellite Data, Sentinel-2, ESA, Big Data, Cloud, OpenStack
DOI: https://doi.org/10.17466/tq2017/21.4/y

1. Introduction

The dynamic development of digital technologies, especially those dedicated to devices generating large data streams, such as all kinds of measurement

equipment (temperature and humidity sensors, cameras, radio-telescopes and satellites – Internet of Things) enables more in-depth analysis of the surrounding reality, including better understanding of various natural phenomena, starting from atomic level reactions, through macroscopic processes (*e.g.* meteorology) to observation of the Earth and the outer space.

On the other hand such a large increase in the data quantity requires a great number of processing and storage resources resulting in the recent rapid development of Big Data [1] technologies having the following common characteristics (3V):

- volume: the quantity of processed, stored data, *etc.*;
- velocity: the speed of data generation;
- variety: many types and sources of data (structured and unstructured).

Sometimes two more characteristics are also included:

- veracity: the data quality and trustworthiness (accuracy, precision, noise and abnormality in a data-set);
- value: the ability to turn data into the market value.

In the current literature [1] there is a tendency to simplify the idea of Big Data and focus only on text or numerical data (*e.g.* sensor measurement stream) processing. Most important open-source platforms like Apache Spark [2] or Hadoop Map Reduce [3] provide specialized tools for analysis of text and (partially) structured data, however, image processing is typically not so well supported. Most work in the image processing and analysis, especially connected with machine learning (*e.g.* deep neural networks), operates on external platforms such as Tensor Flow [4], Torch [5], Theano [6] or Caffe [7].

Since 2015 the European Space Agency (ESA) has been providing a great amount of satellite data under the Copernicus [8] program. Data is gathered using advanced research apparatus – a group of Sentinel satellites. They are responsible for the acquisition of Earth Observation (EO) data in a cyclical manner and are equipped, depending on the mission, with a variety of passive and active sensors (*e.g.* Syntheric Aperture Radar (SAR), optical and Near Infrared (NIR) cameras). Sentinel-2 mission [9] provides imagery along with meta data along with additional analysis results such as polygon masks of cloud coverage.

The main topic of the article is an experimental platform for image analysis, using a typical open-source Big Data [1] processing software: Apache Spark [2]. The proposed solution is used to process Sentinel-2 satellite data – photography and metadata covering the area of the southern Baltic Sea and the Republic of Poland. The above data is publicly available on ESA servers from where it is continuously downloaded to the prepared platform and processed using typical Apache Spark [2] tools available for users in Scala, Python, Java, and R languages.

In the next section, we describe the processed data, both qualitatively (formats, types, *etc.*) and quantitatively (size, growth rate, *etc.*), then we provide

a technical description of both the cloud computing environment (Tryton supercomputer) and the data transfer system itself. Then, satellite data collection along with the main technical aspects of the proposed solution and its usage is described. Finally, the conclusions and future work are presented.

2. Sentinel-2 satellite data

Copernicus [8] is a European Union Earth Observation (EO) program aimed to create services based on a combination of satellite remote sensing and terrestrial sensor data. The ESA has created a Sentinel mission including a pair of Sentinel-2 satellites. Sentinel-2A and Sentinel-2B [9] satellites were deployed in orbit in 2015 and 2017, respectively, and their main task is acquisition and transfer of digital imagery of the Earth's surface. Both satellites are located at an altitude of 786 km above the Earth, in a sun-synchronous orbit such that the angle between its plane and the direction of the Sun's rays is constant, *i.e.* observation is made at the same local time over a given area. For Sentinels-2 it is 10:30 am. This time was chosen as a compromise between the conditions associated with the corresponding angle of the sunlight and the statistically minimal cloud coverage of the area under study.

Sentinel-2 satellites provide 14 band products including spectral ranges such as visible light, infrared, *etc.* The resolution of the recorded images varies from 10 to 60 m. Sentinel-2 covers areas of the Earth's latitude: from 56degS to 84degN, traversing the space above the same area and with the same viewing angle once every 5 days with a 100 kilometer field of view. Each satellite weighs approximately 1.2 tons and is designed to be shipped with a small rocket, such as VEGA [10] or ROCKOT [11]. Its lifespan is estimated for 7.25 years including 3 months for orbital maneuvers, however, batteries can operate up to 12 years. Figure 1 shows the Sentinel-2 satellite model.

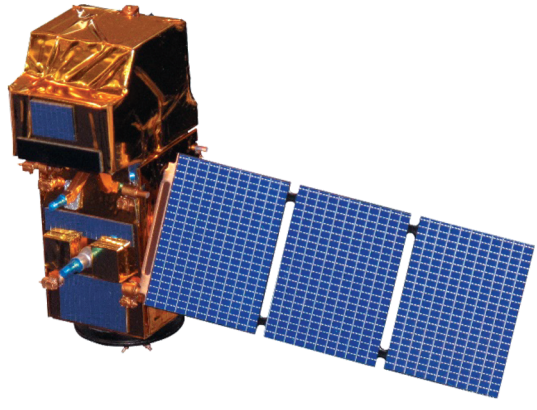

Figure 1. Sentinel-2 satellite model, source: Wikipedia

The Copernicus project [8] aims to provide open data, so that everyone has the opportunity to collect and analyze the datasets. Access to the data is possible

through the Copernicus Open Access hub [12]. Sentinel data can be applied in many areas of research including:

1. climate change;
2. land, atmosphere, maritime environment monitoring;
3. emergency management;
4. security: border and maritime surveillance.

Figure 2 presents the data structure for a single measurement:

- manifest file – an XML document containing general information;
- a preview image with HTML files – a low resolution digital photo;
- subdirectories including metadata, and high definition images (JPEG 2000);
- subdirectories with data processing level information;
- subdirectory with auxiliary data.

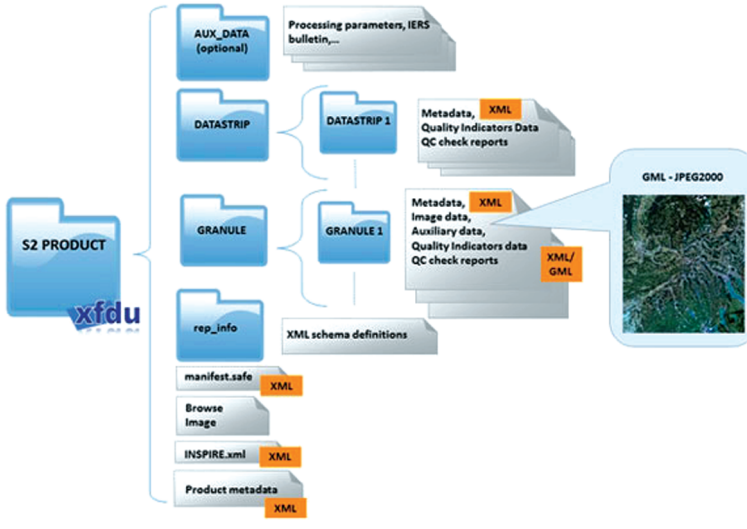

Figure 2. Sentinel-2 data format, source [13]

The existing observations indicate that the amount of data received and collected by the platform (images and metadata) is approximately 1.5–2 TB per month. The above estimates concern selected areas: the Baltic Sea and Poland.

3. Computing environment

Gdansk Academic Computer Centre TASK (CI TASK) has an advanced computing environment for transferring and processing large data sets (Big Data [1]). It is operated by specialized staff with more than 20 years of experience in high performance computing including Big Data and parallel processing and it is the leading center of this type in Northern Poland. Its location in Gdansk is conducive to maritime applications, and extensive cooperation with units of the Gdansk University of Technology (especially with the Faculty of Electronics, Telecommunications and Informatics), as well as other Polish and international universities.

CI TASK's flagship computing unit is Tryton, a cluster-based supercomputer, purchased in 2014 and continuously upgraded. Currently (July 2017) it is composed of over 1600 computing nodes: servers, containing 2 Intel Xeon E5 v3 2.3 GHz (Haswell) processors each, with 12 physical cores with the Hyperthreading (HT) technology. In total, the cluster has over 3200 processors, 38,500 cores, 48 accelerator cards (GP GPUs) and 218 TB RAM [14]. Some of the above resources are used as a computing cloud via the OpenStack platform [15] which allows users to use individual nodes by placing virtual machines (VMs) on them, each of which can use all the resources of a single node, or only a portion of them. This approach supports the IaaS (Infrastructure as a Service) cloud model. An experimental platform for Setinel-2 data processing was installed on the aforementioned supercomputer in an OpenStack environment.

In addition to 1.5 TFLOPS of computing power, the Tryton supercomputer enables the use of the mass storage resources available in CI TASK. Over 5 PB of disk space is directly available for nodes as block or object storage, provided by Ceph [16] and Swift [17] specialized cloud file systems. This solution enables flexible use of resources, both from the cloud level and the perspective of the high performance computing (HPC) users. In the above environment, Apache Spark 2.2 [2]: an open platform for the processing of large data sets (Big Data) has been deployed. It enables efficient analysis and processing of data from a variety of sources, including block and object file systems. Apache Spark has the following features:

- high processing speed (in comparison with Apache Hadoop Map Reduce);
- ease of use – direct support for Scala, Java, Python and R languages;
- generality – it combines a variety of applications: SQL, machine learning, graphical modeling, data streaming and processing (Figure 3);
- portability – integration with Hadoop YARN [18], Apache Mesos [19], Kubernetes [20] and the use of external data sources: HDFS [21], Swift 18, HBase [22] Cassandra [23].

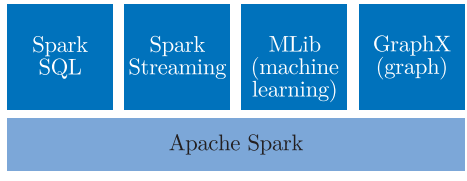

Figure 3. Apache Spark libraries, source: [2]

4. Data processing system

Using the configuration described in the previous section, an experimental system for processing Sentinel-2 imagery and meta-data was proposed. Figure 4 presents the simplified data flow for such processing. The measurements made by the Sentinel-2 satellites (images and accompanying information) are transmitted

(down-link) by radio to the Earth and received by the ESA data reception stations. Raw data is then preprocessed to obtain level 1 and 2 imagery that are then collected on the ESA servers. The prepared client (deployed in CI TASK) is a virtual cloud computing machine, which continually queries the published data and, if the imagery contains the relevant areas (the Baltic Sea and Poland), downloads and stores them in the Swift file system [17]. Such data is ready to be used by any application deployed in the dedicated Swift container.

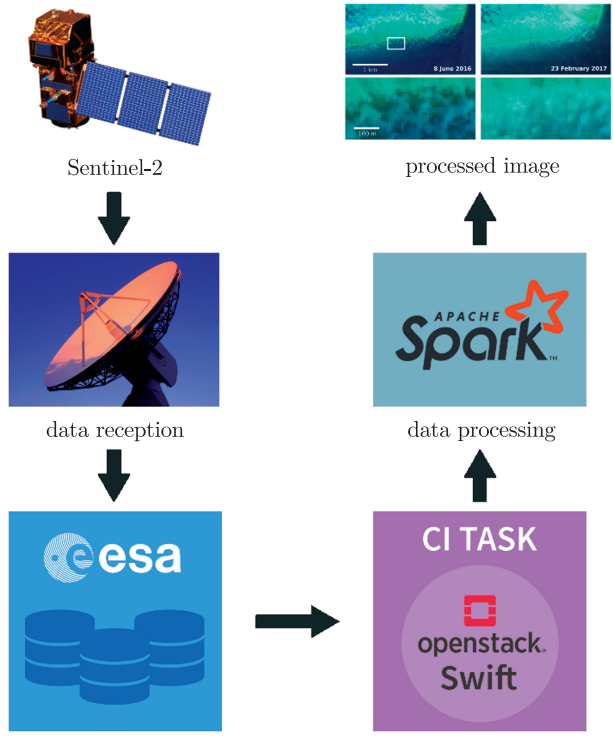

Figure 4. Satellite data processing diagram; source: T. Ziółkowski

The proposed system utilizing the collected data is a cluster of virtual machines running in the OpenStack computing cloud [15] and managed by Apache Spark v. 2.2 [2]. It performs all decompression, decoding, analysis and processing of collected data and then passes the processed images and metadata as output according to the user request. Users can program appropriate processing routines using Java, Python, R, or Scala languages. The language used during the tests was Scala [22]. It is both an object-oriented and functional language that allows the use of existing classes and libraries from Java due to its operation on the Java Virtual Machine (JVM). It also greatly simplifies functional programming supporting stateless and statefull functions, with extensive use of mechanisms such as recursion and parallel processing. The Sentinel Application Platform (SNAP) [24], including the Sentinel 2 Toolbox, implemented in the Java and

provided by ESA that supports image processing can be used to extract individual data from packets corresponding to each measurement. Processing of satellite imagery encoded with JPEG 2000 [25] was done using Java Advanced Imaging (JAI) [26], ImageIO-Ext [27] and OpenJPEG [28] libraries.

5. Example of usage

One of the possible applications of Apache Spark in Sentinel-2 data processing is selection of satellite products based on the information contained in the images and the metadata. In the presented example, it is only those products that: a) present a given area (*e.g.* city of Gdansk) and b) cloud coverage for chosen image has not exceeded a certain value (*e.g.* no more than 30% of the pixels in the image contain clouds) that are selected. The basic metadata documents (Figure 2) describing level 1C and 2A products, are the `MTD_MSIL1C.xml` and `MTD_MSIL2A.xml` files, respectively. They contain information about the cloud coverage (Figure 5) and the Global Footprint (Figure 6), *i.e.* a list of geographical positions (in decimal degrees) that form the polygon describing the imagery area.

```
<n1:Quality_Indicators_Info>
   <Cloud_Coverage_Assessment>75.0991</Cloud_Coverage_Assessment>
</n1:Quality_Indicators_Info>
```

Figure 5. Cloud coverage indicator in `MTD_MSIL2A.xml` file

```
<Product_Footprint>
         <Global_Footprint>
           <EXT_POS_LIST>
              55.00678356187177   17.87240396241576
              55.038802909805916  19.588520320630224
              54.05231184875538   19.622172463921235
              54.021430711029694  17.946857574749785
              55.00678356187177   17.87240396241576
           </EXT_POS_LIST>
         </Global_Footprint>
</Product_Footprint>
```

Figure 6. The area presented in the image is described by a fragment of `MTD_MSIL2A.xml`

The example application (Figure 7) has been implemented in Scala [29] using methods for data and collection stream operations that take a lambda expression *i.e.* an object representing function that is executed on each of the elements of the collection, as a parameter. Especially:

- *map (lambda)* – used to convert each element in a dataset using the lambda expression;
- *filter (condition)* – allows selecting only those elements that meet the condition passed as the parameter.

Apache Spark [2] provides a functionality to create data collection that is automatically distributed to cluster nodes and then to perform operations

```
import scala.xml.XML
import java.awt.geom.Path2D
import org.apache.hadoop.fs.{FileSystem, Path}
import java.net.URI
import java.io.{DataInputStream, EOFException, InputStreamReader}
import org.apache.spark.SparkContext
import java.io.BufferedInputStream
import java.io.DataInputStream
import java.io.File
import java.io.FileInputStream
import java.io.FileNotFoundException
import java.io.IOException
import java.awt.Graphics
import java.awt.Image
import java.awt.image.BufferedImage
import javax.media.jai._
import javax.imageio.ImageIO;
import java.awt.Color
import java.io.ByteArrayOutputStream

def filter_bounding_box(positionsString: String, lat_test: Double,
lon_test: Double): Boolean = {
  val positions: Array[Double] = positionsString.split(" ").map(_.toDouble)
  val product_area: Path2D = new Path2D.Double();
  product_area.moveTo(positions(0), positions(1));
  for (Array(lat, lon) <- positions.grouped(2).drop(1))
    product_area.lineTo(lat, lon)
  product_area.contains(lat_test, lon_test)
}

def create_RGB(path: Path): Option[String] = {
  var fs = FileSystem.get(URI.create(path.toString()),
  SparkContext.getOrCreate.hadoopConfiguration)
  var path2 = None: Option[Path]
  var path3 = None: Option[Path]
  var path4 = None: Option[Path]
  val file_status = fs.listStatus(path)
  for (status <- file_status) {
    val name=status.getPath().toString()
    if (name.contains("B02_20")) path2 = Some(status.getPath())
    else if (name.contains("B03_20")) path3 = Some(status.getPath())
    else if (name.contains("B04_20")) path4 = Some(status.getPath())
  }

  if (!(path2.isEmpty || path3.isEmpty || path4.isEmpty)) {
    val i2 = ImageIO.read(fs.open(path2.get))
    val i3 = ImageIO.read(fs.open(path3.get))
    val i4 = ImageIO.read(fs.open(path4.get))
    val w = i2.getWidth
    val h = i2.getHeight
    val final_image =
    new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB)
    for (x <- 0 until w) {
      for (y <- 0 until h) {
        val red = (i4.getRGB(x, y) >> 16) & 0xff;
        val green = (i3.getRGB(x, y) >> 8) & 0xff;
        val blue = i2.getRGB(x, y) & 0xff;
```

Figure 7. Example Sentinel-2 data filtering

```
58          val rgb = (red << 16 ) | (green << 8) | blue
59          final_image.setRGB(x, y, rgb)
60        }
61      }
62      val output_URI = "swift2d://output.sentinel/"
63      val file_name = path.toString().substring(path.toString()
64          .lastIndexOf('/')+1) + "/B432RGB.jpg"
65      fs = FileSystem.get(URI.create(output_URI),
66      SparkContext.getOrCreate.hadoopConfiguration)
67      val out = fs.create(new Path(output_URI+file_name))
68      val baos = new ByteArrayOutputStream();
69      ImageIO.write(final_image, "jpg", baos);
70      out.write(baos.toByteArray())
71      out.close()
72      Some(output_URI + file_name)
73    }
74    else
75      None
76  }
77
78  val sentinel_metadata = sc.wholeTextFiles
79  ("swift2d: //sentinel_products.sentinel/sentinel2/../*MSIL[12][AC].xml")
80
81  val xml = sentinel_metadata.map{
82    case (file_name, content) =>
83    (file_name,XML.loadString(content))
84  }
85
86  val details = xml.map{
87    case (file_name, xml_nodes) =>
88    (file_name, xml_nodes \\ "EXT_POS_LIST" text,
89    xml_nodes \\ "Cloud_Coverage_Assessment" text)
90  }
91
92  val filtered = details.filter{
93    case (file_name, bounding_box,cloud_percentage) =>
94    (cloud_percentage.toFloat < 40 &&
95    filter_bounding_box(bounding_box, 54.3520500, 18.6463700))
96  }
97
98  val products = filtered.map{
99    case (file_name, bounding_box, cloud_percentage) =>
100   file_name.substring(0, file_name.lastIndexOf('/'))
101 }
102
103 products.foreach{
104   case (file_name) =>
105   create_RGB(new Path(file_name))
106 }
```

Figure 7 – continued. Example Sentinel-2 data filtering

such as a *map* or *filter*. In the first step, the application retrieves information about satellite data stored in the Swift object storage (Figure 7, lines 78–79). It is only XML files that contain the desired metadata (`MTD_MSIL1C.xml` and `MTD_MSIL2A.xml`) that are selected. The `wholeTextFiles` method returns a collection of type (*String, String*), where the first parameter is the name of the file, and the second parameter is its content. Then (lines 81–84) the content of the

metadata files is loaded into the XML parser and information about the cloud coverage and imaging area (lines 86–90) is retrieved. The `filter_bounding_box` method (lines 21–29) is used to determine whether the given geographic position is presented in the satellite image. It takes a list of geographic locations (as shown in Figure 6) as a parameter and builds a polygon describing the area of the image. In the next step it is verified if the position represented by the `lon_test`, `lat_test` parameters is in the specified area. In lines 92–96 data filtering occurs, only those products that meet both conditions are selected: a) the cloud coverage of a given product does not exceed the set value (40% in this case) and b) the position (in this case, the center of Gdansk) is placed in the image (`filter_bounding_box` condition). In the next step, (lines 98–101) the path names to the main catalogs of the selected products are retrieved. In the last phase, data from the selected products is merged into RGB images (lines 103–106).

Images containing bands 4, 3 and 2, *i.e.*, those corresponding to RGB channels [30], are downloaded from the Swift object storage (`create_RGB` function, lines 31–77) and then the data is merged into one image (lines 46–61) and saved in the file system (lines 62–72).

6. Experiments and results

The Spark platform allows easy addition and removal of computing nodes. Two scalability tests were performed to check the impact of the change in the number of workers on the execution time,. The test cluster contains six machines (1 master and 5 slaves) with the following configuration: 32 GB of RAM, a 64-bit processor (32 cores), a 48 GB SSD disk. The data set used contains approximately 12 000 Sentinel-2 level 2A products.

In the first test, satellite data was queried for images presenting the city of Gdansk, with no more than 40% of the pixels containing clouds (program from Figure 7). Then, RGB images where generated from the selected products. In the second test, RGB images were generated for the entire dataset, without any selection process.

In both cases, computations were run on a cluster with 5, 4, 3, 2 and 1 worker nodes. In each of the executions, the number of partitions which is a Spark variable controlling parallelism was set as a result of multiplication of the total number of cores in a cluster by 4.

Figure 8 and Figure 9 present the tests results. The charts present the relation between the number of workers and the execution time.

Figure 10 and Figure 11 present the speedup of the presented tasks in relation to the number of workers.

An increase in the number of workers increases the speed of data processing and allows faster data analysis. The speedup achieved in the second test is much more significant than in the first one. It is a result of using the Spark `wholeTextFiles` method (Figure 7, lines 79–80). This method downloads metadata files available in the Swift storage and reads their content. This operation

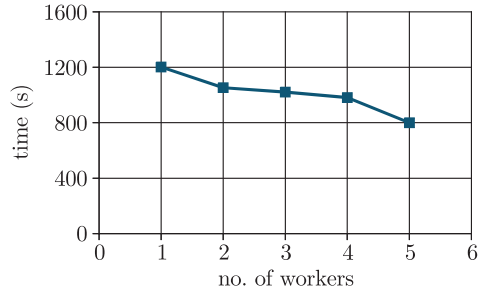

Figure 8. Execution time for Test No. 1

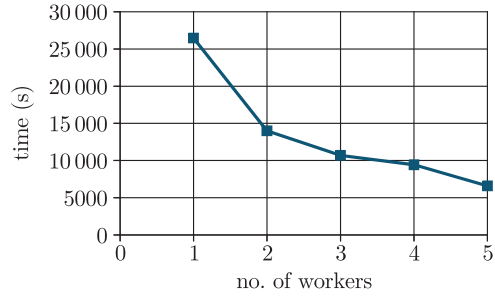

Figure 9. Execution time for Test No. 2

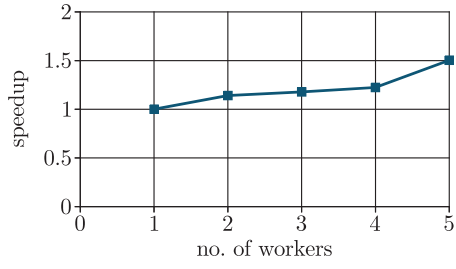

Figure 10. Spark cluster speedup, Test No. 1

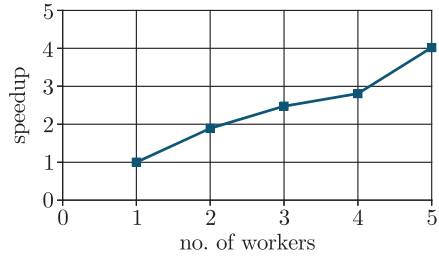

Figure 11. Spark cluster speedup, Test No. 2

is performed entirely on a driver node and only after it has finished, the data is send to workers and parallel processing starts. Since the dataset in the first test is much smaller than in the second one, the metadata processing becomes a bigger part of the entire execution time and leads to a smaller speedup.

7. Conclusions

Currently (November 2017) the proposed system is an experimental deployment that is a basis for further development. Future works should be focused on increasing the efficiency and reliability of data processing and user convenience. We plan to automate the Spark cluster management (in the area of installation and changing the number of computational nodes) [2] in the OpenStack environment [15]. More extensive performance and scalability tests as well as integration with libraries supporting advanced machine learning (*e.g.* H2O platform [31]) are also planned. The aim of the proposed platform is to provide an environment for scientific and educational purposes. It is available to CI TASK users, especially those from the Pomeranian scientific community that are interested in performing Sentinel-2 satellite data analysis of the Baltic Sea as well as for other individuals and organizations interested in processing the gathered data. In addition, the platform is to be used in the "Space applications of advanced information technology" course that is conducted within the interdisciplinary faculty "Space and satellite technologies" at the Gdansk University of Technology (since October 2017).

References

[1] Demchenko Y, De Laat C and Membrey P 2014 *CTS*, IEEE 104
[2] Apache Spark – Lightning-fast cluster computing [Online] available at: `https://spark.apache.org/` [Accessed: 25-July-2017]
[3] Apache Hadoop – website [Online] available at: `https://hadoop.apache.org/` [Accessed: 23-June-2017]
[4] Thensorflow Homepage [Online] available at: `https://www.tensorflow.org/` [Accessed: 23-June-2017]
[5] Torch – a scientific computing platform for LuaJIT [Online] available at: `http://torch.ch/` [Accessed: 23-June-2017]
[6] Theano – a library to define, optimize, and evaluate mathematical expressions [Online] available at: `http://deeplearning.net/software/theano/` [Accessed: 26-June-2017]
[7] Caffe – Deep learning framework [Online] available at: `http://caffe.berkeleyvision.org/` [Accessed: 26-June-2017]
[8] Copernicus – observing the Earth, ESA programe [Online] available at: `http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus` [Accessed: 26-June-2017]
[9] Sentinel-2 Homepage [Online] available at: `http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-2` [Accessed: 23-June-2017]
[10] Vega – Arianspace [Online] available at: `http://www.arianespace.com/vehicle/vega/` [Accessed: 06-July-2017]
[11] Rockot launch vehicle [Online] available at: `http://www.russianspaceweb.com/rockot.html` [Accessed: 06-July-2017]
[12] Copernicus Open Access Hub [Online] available at: `https://scihub.copernicus.eu/` [Accessed: 27-July-2017]
[13] Sentinel-2 MSI, Data formats [Online] available at: `https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/data-formats` [Accessed: 23-July-2017]
[14] Tryton Supercomputer, Academic Computer Centre TASK [Online] available at: `https://task.gda.pl/kdm/sprzet/tryton/`
[15] OpenStack, Open Source Cloud Computing Software [Online] available at: `https://www.openstack.org/` [Accessed: 27-June-2017]
[16] Ceph Homepage [Online] available at: `http://ceph.com/` [Accessed: 27-June-2017]

[17] Swift – OpenStack [Online] available at: https://wiki.openstack.org/wiki/Swift [Accessed: 27-June-2017]
[18] Apache Hadoop YARN [Online] available at: https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-site/YARN.html [Accessed: 27-June-2017]
[19] Apache Mesos [Online] available at:http://mesos.apache.org/ [Accessed: 28-June-2017]
[20] Kubernetes – Production-Grade Container Orchestration [Online] available at: https://kubernetes.io/ [Accessed: 28-June-2017]
[21] HDFS Architecture Guide [Online] available at: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html [Accessed: 28-June-2017]
[22] Apache HBase Home [Online] available at: https://hbase.apache.org/ [Accessed: 28-June-2017]
[23] Apache Cassandra Homepage [Online] available at: http://cassandra.apache.org/ [Accessed: 28-June-2017]
[24] SNAP – Sentinel Application Platform [Online] available at: http://step.esa.int/main/toolboxes/snap/ [Accessed: 28-June-2017]
[25] JPEG 2000 Homepage [Online] available at: https://jpeg.org/jpeg2000/ [Accessed: 28-June-2017]
[26] Java Advanced Imaging API Home Page [Online] available at: http://www.oracle.com/technetwork/java/iio-141084.html [Accessed: 28-June-2017]
[27] The ImageIO-Ext Project Home [Online] available at: https://github.com/geosolutions-it/imageio-ext/ [Accessed: 29-June-2017]
[28] OpenJPEG, an open-source JPEG 2000 codec [Online] available at: http://www.openjpeg.org/ [Accessed: 29-June-2017]
[29] Odersky M 2014 *Scala by Example, Programing Methods Laboratory* [Online] available at: http://www.scala-lang.org/docu/files/ScalaByExample.pdf, EPFL [Accessed: 26-July-2017]
[30] Sentinel-2 User Handbook [Online] available at: https://sentinel.esa.int/documents/247904/685211/Sentinel-2_User_Handbook, ESA [Accessed: 03-July-2017]
[31] H2O Homepage [Online] available at: https://www.h2o.ai/ [Accessed: 03-July-2017]