Big data streaming for remote sensing time series analytics using MapReduce

Luiz Fernando Assis[1], Gilberto Ribeiro[1], Karine Reis Ferreira[1], Lúbia Vinhas[1], Eduardo Llapa[1], Alber Sanchez[1], Victor Maus[1], Gilberto Câmara[1]

[1]Image Processing Department
INPE - National Institute for Space Research
Av dos Astronautas 1758
Caixa Postal Sao Jose dos Campos – SP – Brazil

{luizffga,gribeiro,karine,lubia,edullapa,alber.ipia,gilberto}@dpi.inpe.br

Abstract. *Governmental agencies provide a large and open set of satellite imagery which can be used to track changes in geographic features over time. The current available analysis methods are complex and they are very demanding in terms of computing capabilities. Hence, scientist cannot reproduce analytic results because of lack of computing infrastructure. Therefore, we propose a combination of streaming and map-reduce for time series analysis of time series data. We tested our proposal by applying the classification algorithm BFAST to MODIS imagery. Then, we evaluated account computing performance and requirements quality attributes. Our results revealed that the combination between Hadoop and* ***R*** *can handle complex analysis of remote sensing time series.*

1. Introduction

Currently, there is huge amount of remote sensing images openly available, since many space agencies have adopted open access policies to their repositories. This large data sets are a good chance to broaden the scope of scientific research that uses Earth observation (EO) data. To support this research, scientists need platforms where they can run algorithms that analysis big Earth observation data sets. Since most scientists are not data experts, they need data management solutions that are flexible and adaptable.

To work with big EO, we need to develop and deploy innovative knowledge platforms. When users want to work with hundreds or thousands of images to do their analysis, it is not practical to work with individual files at their local disks. Innovative platforms should allow scientists to perform data analysis directly on big data servers. Scientists will be then able to develop completely new algorithms that can seamlessly span partitions in space, time, and spectral dimensions. Thus, we share the vision for big scientific data computing expressed by the late database researcher Jim Gray: *"Petascale data sets require a new work style. Today the typical scientist copies files to a local server and operates on the data sets using his own resources. Increasingly, the data sets are so large, and the application programs are so complex, that it is much more economical to move the end-user's programs to the data and only communicate questions and answers rather than moving the source data and its applications to the user's local system"* [Gray et al. 2005].

For instance, the standard for land use and land cover monitoring includes to select and download a set of images, processing of each one using visual interpretation or

semi-automatic classification methods, to delineate the areas of interest. This approach is ineffective when there are too much data, or for example, when working on large extensions of land using high spatio-temporal resolution. In contrast to analyzing one image at a time, time-series analysis had become a valuable alternative in land use/land cover monitoring, including early warning of deforestation [Verbesselt et al. 2012a]. Although, we lack environments for validating and reproducing the analysis results of large remote sensing data [Lu et al. 2016, Maus et al. 2016]. To avoid this problem, streaming analytics have emerged as a solution by combining fast access, scalable storage and easy deployment for complex analysis. This approach is able to analyze data in near real-time with low latency and to point to events in regional and global scales without overhead.

Sensor and location-based social networks are common data sources analysis of spatial data in near real-time. Since these network users generate petabytes of data, they are provided through streaming APIs which have several applications, including the analysis the occurrence of events [Assis et al. 2015, Schnebele et al. 2014]. Unlike these streaming APIs, parallel streaming processing plug-ins deal with I/O interpreters in a more intuitively by allowing a powerful and flexible way to analyze data. Hadoop[1] and SciDB streaming [2] are APIs that gather large amounts of data from a file system and multidimensional database such as Hadoop and SciDB respectively. Specifically Hadoop streaming has the advantage of using a standard processing model called MapReduce, which optimized for specific features with different degrees of conformance to the model [Urbani et al. 2014, Dede et al. 2014].

However, most of the MapReduce-based approaches only provide an image library [Sweeney et al. 2011] by means of a customization, which is limiting for analysis. Besides only a small variety of analysis methods are provided at a instance and new complex algorithms are costly to develop and reproduce [Almeer 2012]. Furthermore, most of the available methods extract land use and land cover information using region-based classifications, even though they may cause loss of information [Giachetta and Fekete 2015]. For these reasons, a flexible, generic and broad solution is required to reuse remote sensing time series analysis methods, avoiding the burden of development and adaptation according to the scientific needs.

Therefore, we propose a combination of distributed file systems and complex analysis environments in a MapReduce streaming processing analytics. It is implemented as $<$*key*, *values*$>$ pairs, where *key* is an image pixel location and *values* is the time series associated to that given location. We evaluated this approach, using the BFAST algorithm that iteratively estimates the time and number of abrupt changes within time series, and characterizes change by its magnitude and direction [Verbesselt et al. 2010]. We use BFAST to detect and characterize changes in time series of MODIS (Moderate Resolution Imaging Spectroradiometer) data [Rudorff 2007]. Briefly, the main contributions of this work are:

1. To present a time series-based streaming processing analytics using MapReduce;
2. To discuss the learned lessons from a case study to evaluate our approach in terms of performance and quality requirements;

[1]https://hadoop.apache.org/docs/r1.2.1/streaming.html
[2]https://github.com/Paradigm4/streaming

The remainder of this paper is structured as follows. Section 2 presents a discussion about the time-first, space-later vs space-first, time-later analysis. Section 3 describes the related works while Section 4 outlines our approach using MapReduce for remote sensing time series. Section 5 depicts the evaluation of our approach and its results. Section 6 concludes this paper with recommendations for future works.

2. Time-first, Space-later vs Space-first, Time-later

Scientists have analyzed time series of remote sensing imagery, to detect changes, in three different ways: 1) process each image independently and compare the results for different time instances, 2) build time series of each pixel and process them independently and 3) develop algorithms that process multiple pixels at multiple time instances . The first type of analysis will be called hereinafter as *space-first, time-later* approach. This type of analysis aims to evaluate and compare the results of a pixel classification independently in time. For example, if more than one method of an image classification based on forest cover percentage (see Figure 1) are applied, a pixel may be classified in distinct land cover types. The error resulted in one of them can lead the results to a classification inconsistency when analyzing the pixels of each scene separately. Also, this inconsistency may also increase with the number of scenes and leading to an analysis mistake depending on the application.

Due to this limitation, scientists have used an alternative approach in which the methods are based on what we define as *time-first, space-later* approach. The key is to consider the temporal auto-correlation of the data instead of the spatial auto-correlation [Eklundha and Jönssonb 2012], which is really important for remote sensing time series analysis. In this case, scientists analyze each pixel independently taking into consideration all the values of the pixel along the time (see Figure 2).

For example, given a set $S = \{s_1, s_2, ..., s_n\}$ of remote sensing satellite imagery that depicts the same region at n-consecutive times, we can define them as a 3-D-dimensional array in space-time. For each digital image $s_i \in S$, millions of pixels are associated with their respective spatial location (*latitude*, *longitude*), which corresponds to the (x, y, z) position in a 3D matrix. The z-component of the matrix corresponds to the time axis in the satellite imagery. Each pixel location (x, y, z) contains a set $A = \{a_1, a_2, ...a_m\}$ of attributes values, represented by spectral bands of the set of images. These attributes can provide land-use and land-cover information as each kind of target (forest, water, soil, among others) on the ground has a different spectral reflectance signatures based on the wavelength.

Time-first, space-later approach is more suitable, for example, to detect deforestation or forest degradation from time series of remote sensing imagery. Supposing that we are working with images that have an spectral attribute a that is associated to the forest cover. We can think of a situation in which an area was a prestine forest until 2000, it was cut out in 2001 and started to regenerate in 2010. If we follow the value of a along the time, using the time-series complex analytics, we can monitor this dynamics. If we consider large databases of imagery, with high spatial and temporal resolutions and covering large extensions we will need the best and robust methods to deal with the big EO data. The streaming processing analytics approach presented in this paper, is a contribution to fulfill this demand.

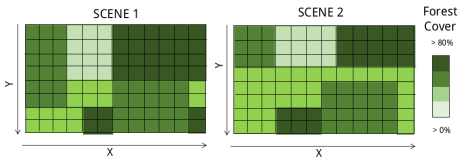

Figure 1. Space First, Time Later

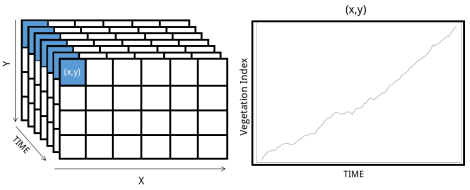

Figure 2. Time First, Space Later

3. Related Works

Due to the increasing interest on EO applications, a set of additional mechanisms have emerged to load, process and analyze remote sensing imagery. These mechanisms aim to convert the images into different data formats since storage components sometimes only accepts a specific representation. Analytic algorithms have been built to enrich existing storage components with more statistical and mathematical operations, but they still lag far behind statistical software packages such as those presented in the CRAN repository. In order to reduce the data movement and the communication overhead between storage and analysis, integrating these storage components and **R** by letting each do what they do best is still a better approach. This combination aims to scale for analytic methods over massive datasets by exploiting the parallelism of storage components in an analyst-friendly environment [Integrating 2011]. The problem about this integration is that a sophisticated understanding of their particular characteristics are mandatory and functionalities need to be re-implemented. For these reasons, data should be acquired, processed and analyzed continuously in an easily and flexible manner in near real-time.

For this, location-based social networks streams analytics have been emerged as the most common approaches in the literature provided by means of APIs. Most of the existing studies that use these streamings aim to provide location-based eventful visualization, statistical analysis and graphing capabilities [Schnebele et al. 2014]. They also aim to explore the spatial information involved in social networks messages. For example, social network messages can be used to detect events in near real-time such as floods and elections [Assis et al. 2015, Song and Kim 2013]. The challenge here is in the combination of different data flows and data formats to support the analysis of high value social network messages in near real-time. In distributed parallel processing, streaming APIs[3][4] have been mainly used to perform an arbitrary set of independent tasks that can be broken into parts, and run separately in another environment with a reusable code. It takes into consideration input/reading and output/writing commands by using stdin and stdout.

Hadoop Streaming is an exemplary API that has an advantage of using MapReduce, a standard processing model, to process in near real-time by customizing how input and output are splitted into key/value pairs. One of the most important features of this open implementation is that Hadoop is fault-tolerant. Its main goal is to support the execution of tasks using a scalable cluster of computing nodes [Rusu and Cheng 2013]. Hadoop-GIS, MD-HBase and SpatialHadoop are exemplary GIS tools that require an extra overhead for more flexible functions [Aji et al. 2013, Nishimura et al. 2013,

[3]https://hadoop.apache.org/docs/r1.2.1/streaming.html
[4]https://github.com/Paradigm4/streaming

Eldawy and Mokbel 2015]. Unlike dedicated proprietary services such as Google Earth Engine that offer minimal standards for scientific collaboration, alternative interfaces of Hadoop can abstract highly technical details for image processing from the point of view of computer vision [Sweeney et al. 2011].

However, when a large amount of analytics algorithms are necessary, these approaches burden the developers and scientists since there is a clearly limitation of available operations and functions, mainly regarding remote sensing time series analysis. Furthermore, existing studies address this approach with a more spatial focus in image classification algorithms [Almeer 2012, Giachetta and Fekete 2015], which result in more loss of information. For these reasons, the high technical complexities involved in developing new applications should be hide from them, and consequently, a more flexible and generic approach is required.

4. Streaming Processing Analytics using MapReduce

Since remote sensing time series analytics require dealing with a large amount of satellite imagery of the same place at different times, it is necessary to build an approach that provides a fast access, a scalable storage and more flexible complex analysis methods. This makes easier to other scientists to reproduce and validate scientific research on this topic. With this in mind, we propose an approach that combines a streaming processing mechanism based on MapReduce with a complex statistical analysis environment. These choices were made based on the flexibility offered by the existing streaming processing that allows the implementation of algorithms in different languages, as well the several analysis components provided by these environments with specific purpose. At first, we stored all the images in a distributed file system so that they are processed by means of two methods (Mapper and Reducer) aiming to build the timeline values and analyze them calling a complex algorithm.

The main advantage of using a standard processing model such as MapReduce is in the fact that both methods receive and transmit data as <*key*, *values*> pairs, giving to the scientists more interoperability and clear capacity of processing data. In our approach, the Mapper input is a <*key*, *values*> pair, in which the *key* is an image identifier and the *values* are all of the desired pixel locations (x,y), that is, the image content itself. The Mapper is responsible for extracting the features from the images for each desired pixel, transforming them into a time series data and emit them to the Reducer. The Mapper output is a <*key*, *values*> pair, in which the *key* is a pixel location (x,y) and the *values* are time series data (e.g., x = 10, y = 45, values = ”0.5 0.7 0.4 0.6” are represented as a <(10, 45), (0.5 0.7 0.4 0.6)> pair). As the Mapper output is the Reducer input, the Reducer receives the combination of pixel and time series values, and analyze them by means of a complex method. The result in this case is stored in the distributed file system. A high level architecture of this time series-based streaming processing analytics for remote sensing data can be seen in Figure 3.

4.1. Data Model and Storage

As a distributed file system is able to store any data type and format without any restriction, its schema-on-read approach offers a more adequate design for our case. Unlike schema-on-write approaches such as database management systems that require a predefined schema to store and query the data, schema-on-read approaches lead to load raw

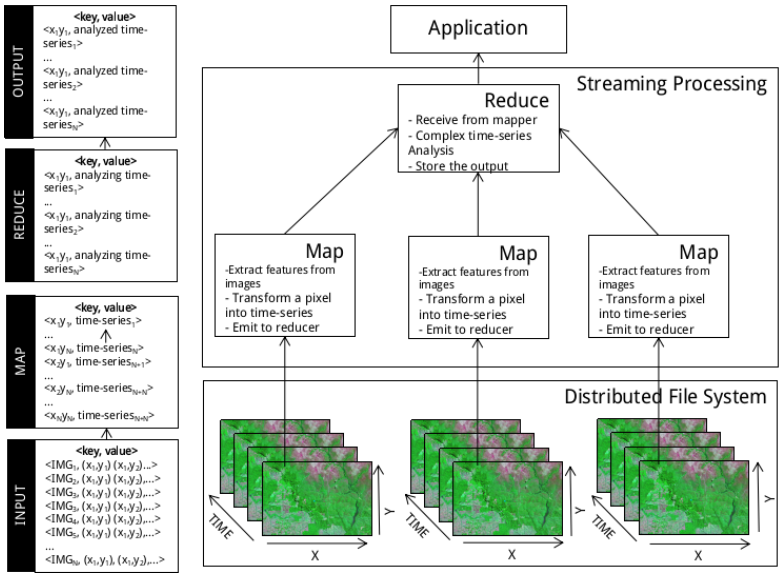

Figure 3. MapReduce Streaming Analytics Processing

and unprocessed data with a structure based on a versatile processing according to the applications requirements. As a result, data not previously accessible are interpreted as it is read, that is, scientists learn the data over time in near real-time. The distributed file system enables the storage of binary files such as raster and shapefiles. Additional tools can help scientists organizing the data either defining a structure or not around their data. In our case, the images gathered by the satellites are stored into years in a sequence it was processed by the provider so that it makes easier to build the time series.

4.2. MapReduce Programming Model

The MapReduce programming model consists of two methods responsible for extracting the features from the images and processing the complex algorithms for remote sensing time series applications in a independently and reusable manner. Both Mapper and Reducer methods receive their input and output by means of standard input (*stdin*) and standard output *stdout* as <key, values> pairs. Unlike other approaches, the <key, values> pairs are line oriented and processed as it arrives, since the Mapper and Reducer controls the processing. In this work, the Mapper performs the filtering and sorting of both pixel and the attributes values into lines, while Reducer performs the complex analysis and stores the result.

An informal high-level description of Mapper can be seen in the Algorithm 1. At first, the Mapper get the dataset names for standardized stored images before creating raster layer objects for them according to the spectral band id chosen by the scientists. The input is a *<IMG, (x_1,y_1), (x_1,y_2), ..., $(x_(n),y_(n)$>* pair, where *IMG* is an identifier for each image and the latter is a list of pixel coordinates to be analyzed. At second, the Mapper builds the time series by getting the values for each pixel. In this part, the scientist define the pixel interval and get the values for each pixel of them. For example, for an entire image, the scientist would define the interval from 1 to 23040000 (4800x4800 - MODIS data resolution). At third, the Mapper calculate the pixel by ceiling the number of the

pixel divided by the image resolution for the row and getting the remainder for the col. Lastly, the Mapper emit the time series built to the Reducer.

Algorithm 1 Transform <key, values> input into a intermediate <key, values>

```
procedure MAPPER
    connection ← openFile("stdin", open ← "readbynary")
    while length(path ← readLines(connection) do
        files ← insert(files, openDirectory(path))
    end while
    closeFile(connection)
    for i←1 to length(files) do
        r[i] ← raster(getDatasets(files[i])[bandId])
    end for
    for pixel←beginInterval to endInterval do
        initialize(values)
        for j←1 to length(files) do
            values ← concatenate(values, getValues(r[j], row←ceiling(j/imageRes),
col←remainder(j/imageRes))
        end for
        emit("stdout", pixel, values)
    end for
end procedure
```

On the other hand, the Reducer receives each <(x,y), *time series*> pair as an input, so that (x, y) is a pixel coordinate and the *time series* are the attributes found in a pixel of an image for a spectral band defined. Similar to the Mapper, the Reducer get the dataset names for standardized files before creating the *time series*. Then, it adapts the time series format as an input for the complex analysis. Finally, the Reducer emit the output as the result of the complex analysis by storing them into the distributed file system (see Algorithm 2).

Algorithm 2 Transform <key, values> from Mapper into output <key, values>

```
procedure REDUCER
    connection ← openFile("stdin", open ← "readbynary")
    while length(line ← readLines(connection)) do
        timeseries ← getTimeSeries(line)
        ts ← preProcess(timeseries)
        analysis ← complexAnalysis(ts)
        emit("stdout", pixel, analysis)
    end while
    closeFile(connection)
end procedure
```

5. Evaluation and Results

5.1. Experimental Setup

***Runtime Environment*:** The experiments were run on a single-node computer with Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz and 16GiB GB RAM memory running Ubuntu 14.04.4 LTS (64 bit).

***Dataset*:** The MODIS scientific instruments launched in the Earth's orbit by NASA in 1999 were used in our experiments since they are able to capture 36 spectral bands ranging in wavelength from 0.4 μ m to 14.4 μ m. They are designed to provide measures description of the land, oceans and the atmosphere that can be used for studies of processes on local to global scales. In our case, we considered the MOD13Q1 Normalized Difference Vegetation Index (NDVI) due to the large amount of remote sensing studies that have focused on time series analysis using this index [Verbesselt et al. 2010, Grogan et al. 2016]. Since MODIS data are provided every 16 days at 250-meter spatial resolution in the Sinusoidal projection and has more than 18,000 satellite images covering Brazil from 2000 to 2016, we built a time series only using a fraction of these data regarding time and space (92 images with 21 Giga Bytes in total).

5.2. Application Case Study: Deforestation Detection

For handling remote sensing imagery as MODIS time series, at first we organized the MODIS data into years. This organization enables us to build an infrastructure able to extract, transform and load all the images by converting them into standard input for the desired methods. In this work, we considered a method, that is part of an **R** package called BFAST, that aims to detect iteratively breaks in seasonal and trend components of a time series [Verbesselt et al. 2011]. This package is not only helpful for deforestation and phenological change detection, but also for forest health monitoring [Verbesselt et al. 2012b]. After running BFAST for a specific pixel (latitude=-10.408, longitude=-53.495), we obtained a breakpoint in 01-17-2011 (see Figure 4). As this processing can be performed for a large amount of other pixels, we are not considering here to check the accuracy of such algorithm. Our focus in this work is on presenting how these kind of analysis can be validate by using a high variety of systems. For example, the deforestation detection in this pixel situated in the state of Mato Grosso in Brazil (see Figure 5) can be seen in DETER[5], a system for deforestation detection in near real-time. The problem here is in the distinct date of breakpoint found when using both sources (BFAST and DETER).

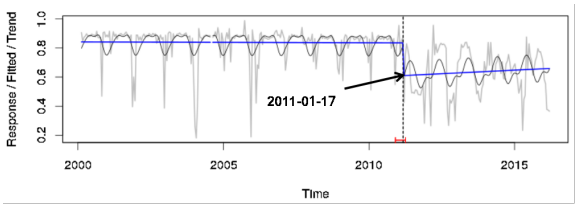

Figure 4. BFAST for a NDVI time series (latitude=-10.408, longitude=-53.495).

In our approach, we decided to integrate Hadoop and **R** since we were able to take the best of massively scalable capabilities and research-friendly programming environment of complex analytics. For evaluating this integration, we performed a set of experiments by using BFAST and other R packages to see how this integration behaves in terms of processing time and scalability (varying the amount of pixel and images). Our tests also allowed us to see how the overhead of these tools affected this kind of processing. The results are shown in Figure 6 for four different amount of images consisting of

[5]http://www.obt.inpe.br/deter/

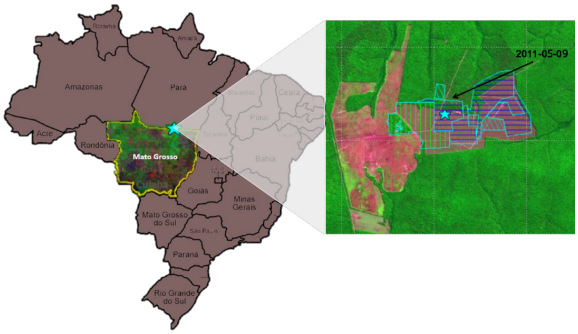

Figure 5. Deforested Area in the state of Mato Grosso in Brazil (latitude=-10.408, longitude=-53.495).

one, two, three and four year MODIS time series data. As we can see, the integration between Hadoop and **R** has a stable, adequate and linear performance even when the amount of information increase with the time. The limitation of the performance is upon to the hardware infrastructure, that is, an extension of the hardware capabilities would provide a better performance in terms of storage and computation power. By comparison, for each thousand of pixels, an amount of 6000 seconds is necessary to analyze using a complex algorithm such as BFAST. The flexibility of running complex algorithms using the familiarity of an **R** script overcome the high cost related to the learning curve of Hadoop. The reason is that in **R** is easy to install and load new packages and a high variety of complex algorithms can be easily deployed.

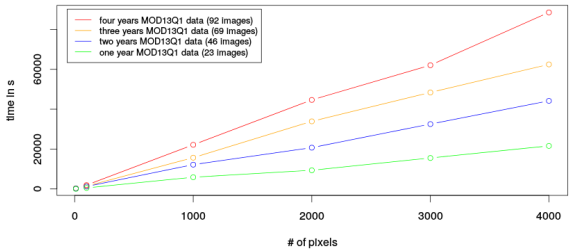

Figure 6. Processing Time to apply BFAST to different amount MOD13Q1 images using MapReduce.

We also calculated the output size files in bytes produced by BFAST in the MapReduce programming model (see Table 1). As we can see, the variation of the image amount change few the size of the output using an algorithm such as BFAST. On the other hand, as the amount of pixel increase the size of the output increase proportionally. The output files contain the timestamps when the break of the time series were detected for each pixel.

In addition, we deployed similar packages in R aiming to detect breaks in time series since they can also be applied to remote sensing time series applications. We considered R packages that help to perform behavioral change point analysis (*bcpa*), change point detection methods (*changepoint*), structural changes detection in regression models (*strucchange*) and behavioral change detection in several other applications (*BreakoutDetection*). The processing time spent for each algorithm is almost the same and can be seen

Table 1. Size Files in Bytes of MapReduce output to apply BFAST

	23 images	46 images	69 images	92 images
10 pixels	171	171	171	171
100 pixels	1792	1792	1783	1750
1000 pixels	18881	18868	18820	18662
2000 pixels	38863	38859	38771	38290
3000 pixels	58849	58844	58722	58107
4000 pixels	78827	78819	78675	77694

in Figure 7. In this experiment, we vary the amount of pixel to a smaller scale compared to the previous one.

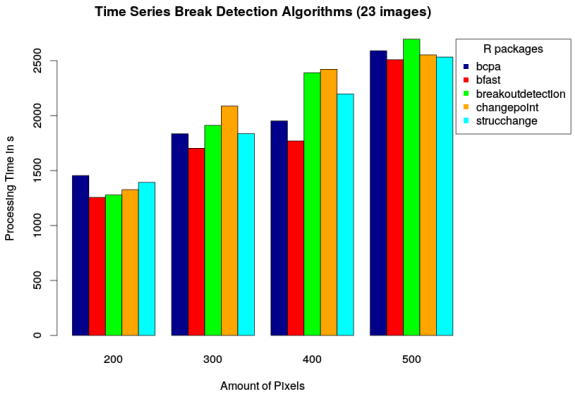

Figure 7. Processing Time to apply several other R packages aiming to detect breaks using 23 images.

5.3. Quality Architectural Requirements

According to [Pressman 2005], external quality architectural requirements correspond to the attributes of the systems that can be recognized by users and are important for design evaluation, which includes performance, flexibility, portability, reusability, interoperability, etc. In this work, we aim to use a qualitative evaluation of these attributes with the main purpose of generating results that can respond whether the designed system meets the architecture quality requirements of domain specialists. For example, decide whether the performance of the software fail or not to compromise the previously planned information processing time.

The chosen method is an adaptation of the most used scenario-based evaluation by industry, also known as Architecture Trade-off Analysis Method (ATAM). ATAM considers how the goals interact with each other in an achieved balance between desirable and compatible features aiming to provide an adequate detail about architectural documents [Nord et al. 2003]. This method guide all the stakeholders to search for conflicts in the architecture, and consequently, solve them. In Table 2 we list the quality attributes found in each architectural decisions. In Figure 8 is depicted the quality attributes in terms of ISO/IEC 25010. We also aim to highlight the level of how hard is to implement each of them and how important they are to the application domain (H: high; M: medium; L: low).

Table 2. List of architectural decisions

Id	Architectural Decision	Quality Attributes	Description
D1	Distributed File System	Performance Fault-Tolerance Reusability	The file system provide fast access to unstructured data in a properly, continuously and reusable operating manner
D2	MapReduce processing model	Modifiability Adaptability	The programming model is easily modifiable for different purposes
D3	Multilayered Architectural	Modularity	The storage, processing and analysis occur in several layers by means of decoupling
D4	Complex Analysis Environment	Learnability	The complex analysis environment should be easy to learn

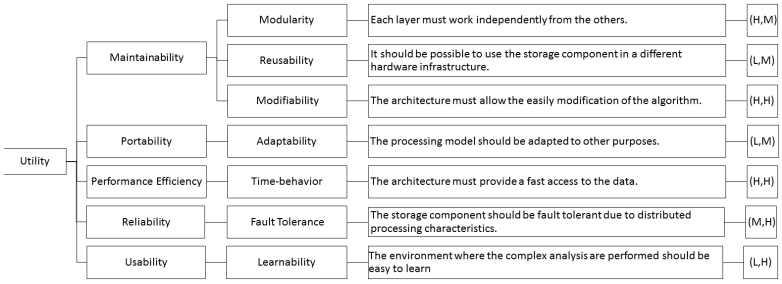

Figure 8. Utility tree.

6. Conclusions

Complying with the memory limitations of the **R**, data scientists often have to restrict their analysis only to a subset of the data. Integrating technologies such as Hadoop with **R** language offer not only a strategy to overcome its memory challenges of large data sets, but also provides a more flexibility programming of complex analysis in storage components. This paper presents an approach for analyzing big remote sensing time series in near real-time using a processing model known as MapReduce.

Our results guide the processing analytics streaming approaches as a more generic way in terms of performance and capacity. They highlighted that for different amount of pixels, and MODIS time series (one, two, three and four years), the processing time was linear for complex algorithms such as those found in deforestation detection applications. Exemplary situations in which such algorithms are important were demonstrated for a specific region in Brazil. Future works will comprise studies about alternative approaches that perform streaming analytics processing in other sources of information such as SciDB, a multidimensional array database. We also plan to evaluate this approach in a multi-node cluster experiment focusing more on data, memory and CPU intensive tests. The Spark framework is also a promising and efficient approach to be tested in our approach.

7. Acknowledgments

Gilberto Camara, Luiz Fernando Assis and Alber Sanchez are supported by São Paulo Research Foundation (FAPESP) e-science program (grants 2014-08398-6, 201515/19540-0 and 2016-03397-7). Gilberto is also supported by CNPq (grant 312151-2014-4). Eduardo Llapa is also supported by a BNDES Fundo Amazonia grant.

References

Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., and Saltz, J. (2013). Hadoop GIS: A High Performance Spatial Data Warehousing System over Mapreduce. *Proc. VLDB Endow.*, 6(11):1009–1020.

Almeer, M. H. (2012). Hadoop mapreduce for remote sensing image analysis. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):443–451.

Assis, L. F. F. G., Herfort, B., Steiger, E., Horita, F. E. A., and ao Porto de Albuquerque, J. (2015). Geographical prioritization of social network messages in near real-time using sensor data streams: an application to floods. In *XVI Brazilian Symposium on Geoinformatics (GEOINFO)*.

Dede, E., Fadika, Z., Govindaraju, M., and Ramakrishnan, L. (2014). Benchmarking mapreduce implementations under different application scenarios. *Future Generation Computer Systems*, 36:389–399.

Eklundha, L. and Jönssonb, P. (2012). Timesat 3.1 software manual. Technical report, Lund University.

Eldawy, A. and Mokbel, M. F. (2015). SpatialHadoop: A MapReduce framework for spatial data. *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, 1:1352–1363.

Giachetta, R. and Fekete, I. (2015). A case study of advancing remote sensing image analysis. *Acta Cybernetica*, 22:57–79.

Gray, J., Liu, D. T., Nieto-Santisteban, M., Szalay, A., DeWitt, D. J., and Heber, G. (2005). Scientific data management in the coming decade. *SIGMOD Rec.*, 34(4):34–41.

Grogan, K., Pflugmacher, D., Hostert, P., Verbesselt, J., and Fensholt, R. (2016). Mapping clearances in tropical dry forests using breakpoints, trend, and seasonal components from modis time series: Does forest type matter? *Remote Sensing*, 8(8):657.

Integrating, R. (2011). Bridging two worlds with rice. *Proceedings of the VLDB Endowment*, 4(12).

Lu, M., Pebesma, E., Sanchez, A., and Verbesselt, J. (2016). Spatio-temporal change detection from multidimensional arrays: Detecting deforestation from {MODIS} time series. {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, 117:227–236.

Maus, V., Câmara, G., Cartaxo, R., Sanchez, A., Ramos, F. M., and de Queiroz, G. R. (2016). A time-weighted dynamic time warping method for land-use and land-cover mapping. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.

Nishimura, S., Das, S., Agrawal, D., and El Abbadi, A. (2013). Md-hbase: design and implementation of an elastic data infrastructure for cloud-scale location services. *Distributed and Parallel Databases*, 31(2):289–319.

Nord, R. L., Barbacci, M. R., Clements, P., Kazman, R., and Klein, M. (2003). Integrating the architecture tradeoff analysis method (atam) with the cost benefit analysis method (cbam). Technical report, DTIC Document.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

Rudorff, B. F. R. (2007). *Sensor Modis e Suas Aplicações Ambientas no Brasil*. Editora Parêntese.

Rusu, F. and Cheng, Y. (2013). A survey on array storage, query languages, and systems. *arXiv preprint arXiv:1302.0103*.

Schnebele, E., Cervone, G., Kumar, S., and Waters, N. (2014). Real time estimation of the calgary floods using limited remote sensing data. *Water*, 6(2):381–398.

Song, M. and Kim, M. C. (2013). Rtˆ2m: Real-time twitter trend mining system. In *Proceedings of the 2013 International Conference on Social Intelligence and Technology*, pages 64–71.

Sweeney, C., Liu, L., Arietta, S., and Lawrence, J. (2011). Hipi: A hadoop image processing interface for image-based mapreduce tasks. Technical report, University of Virginia.

Urbani, J., Margara, A., Jacobs, C., Voulgaris, S., and Bal, H. (2014). Ajira: a lightweight distributed middleware for mapreduce and stream processing. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 545–554. IEEE.

Verbesselt, J., Hyndman, R., Zeileis, A., and Culvenor, D. (2010). Phenological change detection while accounting for abrupt and gradual trends in satellite image time series. *Remote Sensing of Environment*, 114(12):2970–2980.

Verbesselt, J., Zeileis, A., and Herold, M. (2011). Near real-time disturbance detection in terrestrial ecosystems using satellite image time series: Drought detection in somalia. Technical report, Faculty of Economics and Statistics, University of Innsbruck.

Verbesselt, J., Zeileis, A., and Herold, M. (2012a). Near real-time disturbance detection using satellite image time series. *Remote Sensing of Environment*, 123:98–108.

Verbesselt, J., Zeileis, A., Hyndman, R., and Verbesselt, M. J. (2012b). Package 'bfast'.