A new Platform for Time-Series Analysis of Remote Sensing Images in a Distributed Computing Environment

Sávio S. Teles de Oliveira[1], Marcelo de C. Cardoso[1], Wisllay M. V. dos Santos[1], Paulo C. P. Costa[1], Vagner J. do Sacramento Rodrigues[1], Wellington S. Martins[2]

[1]GoGeo
Rua Leopoldo Bulhões, esquina com a Rua 1014
Quadra 31, Lote 07, Sala 9 Setor Pedro Ludovico
CEP 74820-270 - Goiânia - GO - Brazil

[2]Instituto de Informática - Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia
131 - CEP 74001-970 - Goiânia - GO - Brazil

{savio.teles, marcelo.castro, wisllay.vitrio, paulo.cezar, vagner}@gogeo.io,
wellington@inf.ufg.br

Abstract. *Time series analysis of remote sensing images is essential for the detection of patterns, trends and changes to allow the modeling and prediction of events in the earth's surface. Users of geographic information systems (GIS) are often involved in spatio-temporal remote sensing analysis. In case of applications with large volumes of data, this analysis should be carried out in an automated manner and allow spatiotemporal filtering in the image database. This work proposes a new platform, DistSensing, that can enable these analyses to be conducted with the aid of distributed indices. We show how this DistSensing platform outperforms the solutions found in the literature when there is the need to run queries on the database using temporal and spatial filters.*

1. Introduction

The Earth's surface is changing at an unprecedented rate, with the alarming disappearance of much of the forest ecosystem, while urban and farming areas are expanding around natural spaces. A time series analysis of remote sensing images is essential to detect these changes [Neves et al. 2015]. This entails, for example, providing information on shifts in the spatial distribution of bio-climatic zones, and indicating the variations in large-scale global circulation patterns or changes in land-use.

It is essential to make users aware of both the spatial and temporal dimensions in a Geographic Information System (GIS), because these may reveal implicit relationships which match the reality of the analyzed data [de Oliveira and de Souza Baptista 2012]. The GIS is a computer-based tool for mapping and analyzing feature events on earth, which must allow users to conduct a time-series analysis of remote sensing data filtering in specific geographical regions and at regular time intervals, for example, to trace the evolution of deforested areas in the Amazon forest from 1991 to 1997.

The incorporation of latest-generation sensors to airborne and satellite platforms has led to a nearly continuous stream of data [Smits and Bruzzone 2004], and this sharp

rise in the amount of collected information has raised new processing challenges with regard to the time-series analysis of remote sensing data. In addressing these computational requirements, several research endeavors have recently been geared to incorporating high performance computing models in the remote sensing field [Plaza and Chang 2007].

Our work introduces a new platform called DistSensing, to perform the distributed processing involved in the time series analysis of remote sensing data, which allows users to have real time spatio-temporal filters. DistSensing obtained more impressive performance gains than other time series processing platforms found in the literature ([Van Den Bergh et al. 2012, Song et al. 2015]), when spatiotemporal filters are required. The main contributions to this field of studies made by this work are as follows: i) it sets out a strategy for partitioning the remote sensing images into cluster nodes; ii) it includes a search algorithm for the processing of the time series analysis of remote sensing images by means of distributed indexing.

The remainder of this work is structured as follows. Section 2 describes strategies found in the literature to process remote sensing images. Section 3 provides a review of time series processing based on remote sensing data. Section 4 outlines the DistSensing platform. Section 5 describes the methodology and the experimental results. Section 6 summarizes the conclusions and includes a brief description of further work that will be carried out in the future.

2. Related Work

[Ferreira et al. 2015] created a new RDF vocabulary to access spatiotemporal datasets from different kinds of data sources, including remote sensing images. Some works sought to automatically detect changes by using remote sensing images, like [Neagoe et al. 2014] who adopted non-supervised approaches with less manual intervention. In [Romani et al. 2009] a new algorithm was designed to reveal changes in weather patterns by plotting the pixels location from an image that was created to partition them. All these works relied on a single server to process the images, instead of using the computing resources in a *cluster*.

Several studies employed the MapReduce model. [da Silva Ferreira et al. 2015] introduced the architecture of a distributed platform named InterIMAGE Cloud Platform (ICP) to handle with very large volumes of data using clusters of low-cost computers with the Hadoop framework. In [Lv et al. 2010] a K-means clustering algorithm was implemented for remote sensing images and in [Wang et al. 2012] a classification algorithm was created for high resolution remote sensing images. The MapReduce model was also used by [Almeer 2012] to create a parallel processing platform for remote sensing images based on a cloud computing system. In [Lin et al. 2013], a platform was proposed and implemented using Hadoop to process remote sensing algorithms with MapReduce models. The work by [Rathore et al.] relies on an architecture to analyze remote sensing images in real time with *Big Data* using Hadoop. However, these works fail to provide any solution for conducting time series analysis by means of remote sensing images.

[Song et al. 2015] created the Spatiotemporal platform for the time series processing of spatial objects, through cloud computing. This includes HDFS, which is used as a distributed file system, and a MapReduce based computing service, which is used to analyze spatial data. This paper does not discuss spatial and temporal indexing tech-

niques nor how the image distribution is carried out in the cluster. When using HDFS, it is important to define the data distribution algorithm, since images from the same geographical region and collected during the same time interval, should be stored in the same machine [Van Den Bergh et al. 2012]. This is to avoid generating a high volume of network traffic while the time series analysis is being processed.

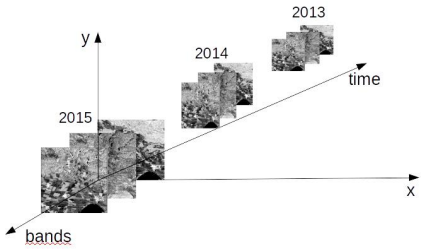
[Van Den Bergh et al. 2012] established the HiTempo platform to assist research in the area of time series analysis based on remote sensing images. The images were stored as 3D objects, with geographically closed images obtained in the same time interval, being stored in the same machine. HiTempo was designed to make possible the evaluation and comparison of remote sensing algorithms and, for that reason, the time-space filters are only applied before inserting the images into the platform, to prevent the filters from being applied at query time. This means that, it is not possible to perform spatial and temporal filtering after the data insertion and the query cannot apply filters to the image set stored in the HiTempo platform.

3. Time Series Analysis of Remote Sensing Images

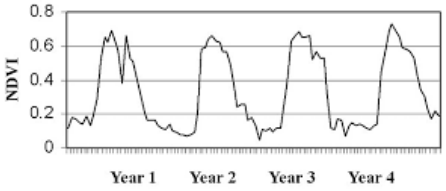
There are a wide range of satellites and sensors with different resolutions. In our study, remote sensing images were chosen from LandSat-8, because they are public and easy to access. Each scene taken by a satellite in a given geographical area and time is called a scene. Each scene from LandSat-8 consists of nine different spectral bands, covering an area of around 170 km north to south and 183 km east to west.

A time series analysis of remote sensing images collected by satellite is needed to evaluate the features of terrestrial surfaces. A time series is a sequence of images, collected from successive, and usually uniformly spaced, time intervals. A time series analysis includes methods that are employed to identify patterns and trends, detect changes, and cluster and model data.

In Figure 1(a) it is possible to visualize the four dimensions of the time series of remote sensing images. In this graph, the x and y axes represent the spatial limits of each band from a scene, where x is the scene's geographical longitude, and y is the scene's geographical latitude. The "bands' 'axis represents the spectral bands of each scene, and the "time" axis represents the satellite images obtained from various points of time.



(a) Dimensionality of a remote sensing time series.



(b) Changes in the NDVI in a historical series.

Figure 1. Time series of remote sensing images.

To validate the DistSensing platform in this work, an algorithm for a time series analysis of remote sensing images was implemented to validate the DistSensing platform

used in this study, by means the NDVI (Normalized Difference Vegetation Index) historical analysis. The NDVI is a numerical indicator correlated with certain physical properties of the vegetation canopy: leaf area index, vegetation condition, and biomass [Carlson and Ripley 1997]. Through this analysis, it is possible to visualize, for example, the increase in deforestation in certain regions throughout the period by looking at their NDVI time series [Morton et al. 2005]. The final result, for instance, is a graph, containing the NDVI mean for each date, as can be seen in the example from Figure 1(b).

The NDVI is calculated from the difference between the Infrared and Near Red bands, normalized by the sum of the same bands, using Equation 1. The NDVI is derived from spectral reflectance measurements acquired in the visible (RED) and near-infrared (NIR) regions, where NIR and RED are the reflectance in infrared and red spectral intervals, respectively. The result of the NDVI ranges between -1 and +1.

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{1}$$

4. DistSensing: A new Platform for Efficient Distributed Processing of Remote Sensing Time Series

This paper creates a new platform, DistSensing, for processing remote sensing time series in a distributed manner. DistSensing has an highly available, elastic and fault-tolerant architecture (Section 4.1) and provides efficient distributed algorithms for storing, indexing (Section 4.2) and querying images at its databases (Section 4.3).

4.1. DistSensing Architecture

DistSensing has a peer-to-peer architecture in which the cluster servers do not share CPUs, hard drives or memory and all communication is carried out via a message passing system. It is composed of a client layer and a server layer. Client applications interact with the platform through an API for updating and querying the databases. Each client application must use a client library for DistSensing API so that the platform's available services can be used. Read and write requests can be sent to any server of the cluster. The list of servers can be obtained from the platform's name service.

The server that receives the client request becomes the coordinator for that specific request. The coordinator acts as a proxy between the client application and the cluster servers. To ensure high availability, when the coordinator is unavailable another server is chosen as coordinator. The client library sends the request, receives the response and transfers the result of the operation to the client application.

Cluster nodes exchange information with each other every second using the gossip protocol [Subramaniyan et al. 2006]. The Gossip protocol is used for finding out and sharing location and status information about other cluster servers. Each server exchange messages not only about its status and other factors but also regarding other cluster data nodes with up to three other servers. Thus, all the cluster nodes become aware of the status of the other nodes of the cluster quickly. Failure detection occurs on the basis of the data exchanged via the gossip protocol. The platform uses this information to avoid requesting processing to unavailable servers.

DistSensing stores data replicas on multiple nodes to ensure fault tolerance and reliability. All the replicas are equally important, since there is no primary or master replica. Every object stored is given a single identifier key generated via hashing. Each server is responsible for an equal range of cluster keys, that are accessed through a Distributed Hash Table (DHT) [Karger et al. 1997]. When an object is required, the elected server is chosen by means of the DHT.

To ensure platform's elasticity, machines might be added or removed from the cluster at any moment. Whenever this happens, the DHT is rebuilt to reflect the new configuration of the cluster. When a new server joins the cluster it queries the DHT and sends a message to the server with the highest disk usage in the cluster to obtain half of the keys kept in this server. Upon removal of the server S its keys are distributed among the remaining servers and its objects are copied from replicas.

4.2. Distribution and Storage of Remote Sensing Images on Cluster Servers

The architecture of DistSensing allows several remote sensing images to be inserted in a distributed and parallel manner. Figure 2 shows step-by-step what happens when an image is inserted in the platform. Insertion starts with a client sending the image bands to the DistSensing platform using the API. Each image band is then sent to a randomly chosen server of the cluster.

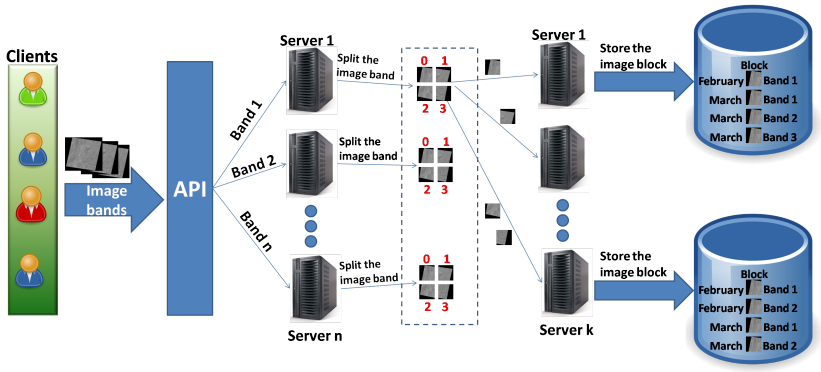


Figure 2. Distribution and Storage of Remote Sensing Images on DistSensing Platform.

In each server, each spectral band of the image is broken into smaller fixed-size blocks that allow a parallel processing of images during the queries. The block size directly affects the size of the job that has to be carried out in the query operations involving the remote sensing images. The greater the number of cores in the cluster servers, the smaller the remote sensing image block size must be to increase the parallelization and improve the efficiency of query processing. This block size must be set from tests conducted in the cluster environment where the platform is installed. To ensure reliability and fault tolerance, as well as providing load balancing, each block is stored on a server and replicated in two other servers.

As Figure 2 shows, block distribution is based on four dimensions of each block: the first two are x and y, and represent the spatial limits of the block, the third is the $time$

and the fourth one is related to the image $band$. Each server in the cluster has its own storage system regardless of the others servers, where the blocks are stored. Thus, blocks from the same geographical area collected at different times are stored at the same server, and this reduces network traffic during the execution of time series analysis. By means of this block distribution, the remote sensing algorithms that carry out the time-series analysis in each image pixel (e.g. NDVI) do not have to send data over the network, since the historical series of each pixel is stored in a server.

Each image block is linked to a geographical location and contains metadata from the original image from which it was generated, such as the date when the image was captured. The *Quad Tree* [Finkel and Bentley 1974] spatial index is built in each server of the cluster, allowing us to perform spatial filters. The index in each server is built from each spatial image block bounding stored in that server. The DistSensing platform is responsible for coordinating the distributed spatial filter amongst the several spatial indices.

The *Quad Tree* is a tree data structure in which each internal node has exactly four children and it was chosen because of its ability to filter remote sensing data [Fu et al. 2013]. The key idea is to partition the space into four quadrants. Each node in the tree either has exactly four children, or has no child. Each child represents a subquadrant of its parent. Quadrants that can no longer be subdivided are represented as leaf nodes that contain data corresponding to that specific subregion.

In addition, an index is built from the metadata (remote sensing image attributes, like the date) of each image in the database. This index is important since it provides with ways to, for instance, conduct an analysis at specific time intervals. The Apache Lucene [Lucene 2010] library was used to build this index. It is available at each server of the cluster and the DistSensing platform is responsible for coordinating the distributed filter amongst the Lucene search engines. The client sends a request that defines the restrictions using Apache Lucene query syntax, which allows it to filter images through phrase, wildcard, range and full-text-search queries.

4.3. Querying the Remote Sensing Database

The query execution process starts with the client sending the request to the DistSensing platform with the spatial filters and metadata attributes constraints. One of the platform's servers is then chosen as the Master that coordinates the execution of that request. Algorithm 1 describes the steps that must be followed inside the DistSensing platform.

The query request is sent to all the cluster servers, since each server stores remote sensing image blocks. After this, each server locally processes the spatial filters by means of the *Quad-Tree* and the image metadata, by using the Apache Lucene on the basis of client-specified restrictions. The result is stored in $block\_ids$ and sent to the Master server as a list of ids of image blocks, along with metadata from the blocks necessary for the execution.

The Master server is responsible for receiving the Ids and metadata and removing duplicates, since the data is replicated among the servers. The unique ids and metadata are stored in the variable $global\_block\_ids$. For each resulting id in $global\_block\_ids$, the DistSensing platform queries its DHT to discover which servers are storing the image block with that id. This DHT stores the id of each object as its key and the IP addresses

of servers as value. A server S is randomly chosen from this list and combined with that id. The table $server\_ids$ is built using the server S as key and the block ids stored in S as value. For each server S in $server\_ids$, the list of ids and metadata is retrieved and then sent to server S.

Algorithm 1: $Query(spatial\_filter, metadata\_filter)$

Data: $spatial\_filter$: spatial filter containing the spatial restriction, $metadata\_filter$: metadata filter using Lucene's query syntax
Result: $final\_result$

$Master$ server send $spatial\_filter$ and $metadata\_filter$ for all cluster servers
for *Server* S *of the cluster* **do**
    $metadata\_block\_ids \Leftarrow$ Ids from the block accepted by the metadata filter
    $spatial\_block\_ids \Leftarrow$ Ids from the block accepted by the spatial filter
    $block\_ids \Leftarrow metadata\_block\_ids \cap spatial\_block\_ids$
    Send $block\_ids$ for $Master$
end
$Master$ server receives $block\_ids$ and creates the set $global\_block\_ids$
for $id \in global\_block\_ids$ **do**
    $S \Leftarrow$ IP from one of the servers storing block with id id
    Insert id id on ids list with key S of the table $server\_ids$
end
for $S \in server\_ids$ **do**
    Send the list ids_list associated with the key S to the server S
end
for *Server* S *of the cluster that receives an* ids_list **do**
    for $id \in ids\_list$ **do**
        $b \Leftarrow$ retrieve block with id id
        $b\_result \Leftarrow$ run the remote sensing time-series analysis algorithm using b as input
        Add b_result on block list $local\_result$
    end
    $local\_result \Leftarrow$ result from local result aggregation algorithm on $local\_result$
    Send $local\_result$ to $Master$ server
end
$Master$ server receives $local\_result$ list from each server S and aggregates the result in $final\_result$
$Master$ server sends $final\_result$ to the client

In each server S, the blocks are retrieved from the storage system and the remote sensing time-series analysis is conducted with the algorithm requested by the client, such as the NDVI time-series analysis. The results obtained from processing this block list, are stored in $local\_result$ and sent to the master server. The master server receives replies from each server and aggregates them in order to create the final result, which is then

sent to the client. This aggregation algorithm is defined by the clients of the DistSensing platform in accordance with their requirements.

On the basis of the analysis shown in Section 3, which employs the normalized difference vegetation index (NDVI), the client sends the spatial and metadata restrictions to the DistSensing platform. The date attribute is then used to select remote sensing image blocks in a given time slice. The coordinator sends the client request to each server of the cluster which calculates the average NDVI for each image block. Once the average NDVI is known for all the image blocks, the values are aggregated by date to create the average NDVI for each date. These aggregate values are sent to the Master server which processes the final aggregation on the basis of data obtained from all the servers and generates the average NDVI for each date. These values are sent to the client which is then capable of creating graphs of NDVI variations over a period of time.

5. Performance Evaluation

An algorithm was implemented to evaluate the performance of the DistSensing platform and thus allow an analysis of the temporal variations in the NDVI values, as shown in Section 3. The response times from the DistSensing platform are compared with those of the HiTempo platform recommended by [Van Den Bergh et al. 2012], which is similar to the proposal put forward by [Song et al. 2015], that are unable to perform temporal and spatial filters during the query execution. The image processing module from the solution found by [Van Den Bergh et al. 2012] was implemented to evaluate a HiTempo platform with spatial and temporal constraints and in the final stage.

5.1. Experiments and Database

A horizontal scalability test was conducted to measure the performance of the DistSensing platform when adding servers to the cluster. Ideally the system should have linear horizontal scalability, which is not always possible: a) since the tasks distributed in the cluster have to be synchronized and b) the devices are subject to physical constraints as in the case of those used for networking. The tests were run on *Intel Core i7 3.4 GHz* machines, with 16 GB of RAM and 1 TB hard drives. The machines were connected to a 1Gbit/s Ethernet network and a 6248P Dell PowerConnect switch. 53 remote sensing images from Brazil were collected by the LandSat-8 satellite from 2013 to 2015, each image with approximately 2 GB. Each of the image's band was broken into blocks of 1024 x 1024 pixels during the insertion procedure.

The algorithm for analyzing temporal variation of NDVI values was evaluated with clusters of 1, 2, 4 and 8 servers and took account of several spatial and temporal filter configurations: i) No filter, covering 100% of the database, and mirroring the HiTempo strategy (HiTempo), ii) spatial constraints covering 27,1% of the database (Large_Spatial), iii) spatial constraints covering 6,1% of the database (Small_Spatial), iv) temporal constraints covering 34% of the database (Date_Large), v) temporal constraints covering 5% of the database (Date_Medium), vi) temporal constraints covering 1% of the database (Date_Small), vii) having spatial and temporal constraints that result in an empty response. Names in brackets are used in the charts shown in Section 5.2.

The algorithm was executed 20 times for each test configuration and the response time on the charts is the average of the 20 response times observed. A client library was

created so that data could be sent for insertion, and queries performed with spatial and relational constraints which involved measuring the response times for each specific test.

5.2. Evaluation

The first test was carried out to determine the scalability of the DistSensing and HiTempo platforms when processing remote sensing time-series analysis, as shown in Figure 3. The response time was obtained after the time-series analysis had been conducted on the entire database. Both platforms achieved a lower response time as more machines were added to the cluster. This scalable behavior was made possible by dividing the image into blocks, and allowing a distributed and parallel processing of these blocks on the cluster. The HiTempo platform had a similar behavior and response time, since there was no need for subsequent filtering.

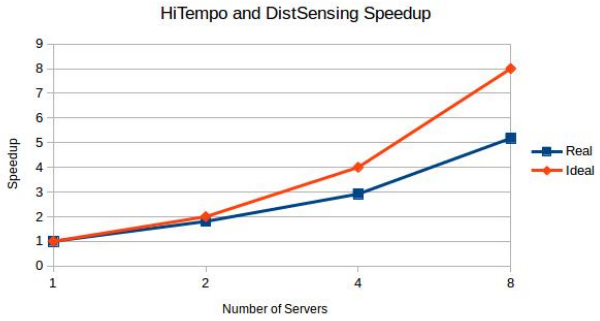
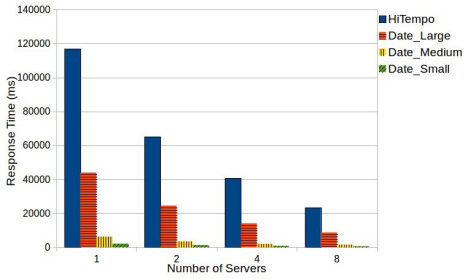


Figure 3. Horizontal Scalability.

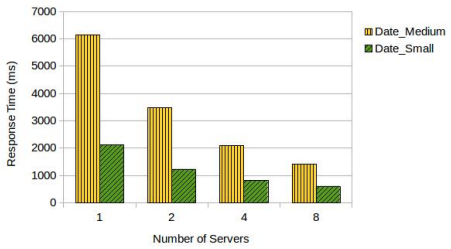
Figure 4 shows a comparison between HiTempo and DistSensing platform when running the analysis with temporal filters on the image database. Compared with the HiTempo platform, Figure 4(a), DistSensing achieves response times up to 53 times lower if the "Date_Small" temporal filter is selected for the execution. Since the HiTempo platform does not allow temporal filters at the query time, it must process the complete database and apply the filter later. With "Date_Small" temporal filter, the DistSensing platform was able to filter 99% of the database, leaving only 1% for processing the analysis on remote sensing images.

The DistSensing platform shows scalability when processing the algorithm with temporal filters, especially with filters that are less restrictive like "Date_Large" where there is a greater need for computational resources during the analysis of the image time series. As shown in Figure 4(b), the DistSensing platform shows scalability when processing algorithms with more restrictive filters than "Date_Large", although not at the same levels of scalability. However, adding more servers for restrictive filters like "Date_Small" and "Date_Medium" do not result in proportionally lower response times, unless the image database increases and requires more computational resources.

Spatial filters are important for remote sensing specialists since they allow them to conduct an analysis that takes into account their geographic context and interest. Figure 5 shows the performance of the DistSensing platform when the algorithm is processed with spatial filters. The DistSensing platform was up to 9 times faster than the solution



(a) Comparison between DistSensing platform and the solution proposed by HiTempo platform.



(b) Scalability on temporal filters.

Figure 4. Database temporal filtering.

proposed by the HiTempo platform when the spatial filter "Small_Spatial" was used by the algorithm. With the "Date_Small" temporal filter DistSensing was up to 53 times faster because this filter covers 1% of the database while the "Small_Spatial" spatial filter corresponds to 6,1%.

Tests were carried out with spatial and temporal constraints which lead to empty responses. The DistSensing platform showed constant response times of approximately 10 ms in these cases, regardless of the cluster size. The HiTempo platform performed up to 11,600 times worse, with response times of 116 seconds when only one server was used. With 8 servers HiTempo took 23 seconds to process the data, which is a performance 2300 times worse than that of DistSensing. The reason for this huge difference is that DistSensing is capable of filtering out images that do not meet the constraints, before processing the algorithm on the remote sensing images.

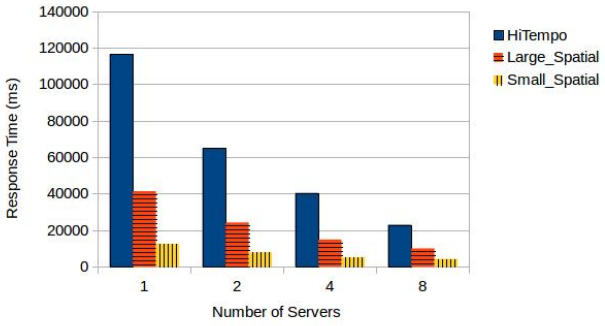


Figure 5. Database spatial filtering with HiTempo and DistSensing platforms.

The algorithm employed for analysing the temporal variation of the NDVI values, was found to be scalable, both with regard to the HiTempo platform and the DistSensing platform, as more servers were added to the cluster. The DistSensing platform had a better performance when temporal and spatial filters were used, with a speed up to 53 times faster when a temporal filter was used that corresponded to 1% of the database, and up to 9 times faster if a spatial filter was used that corresponded to 6,1% of the database. In cases where spatial and temporal filters lead to an empty dataset, the DistSensing platform was up to 11,600 times faster.

6. Conclusion

Time series analysis is crucial for detecting patterns, trends and changes, as well as providing us with ways to model and predict events on the earth's surface. The automatic execution of an analysis of remote sensing time series has become a challenge due to the increase in the amount of remote sensing data. To the best of our knowledge, there is no study on the methods employed for processing a time series analysis that allows the filtering of data based on a geographical region and time period.

This paper proposes a new platform called DistSensing, to conduct an analysis of remote sensing time series in a distributed manner. Spatial and relational indices were built to provide query functionality at the image database. DistSensing allows patterns, trends and changes on the earth's surface to be detected with lower response times. When temporal filters are used, DistSensing is up to 53 times faster than the HiTempo platform proposed by [Van Den Bergh et al. 2012], and if spatial filters are needed, the performance can be up to 9 times better than the HiTempo platform. Furthermore, if the combination of spatial and/or temporal filters generates an empty dataset, the DistSensing platform shows a response time that is 11,600 times lower the those of the HiTempo platform.

Tests on bigger clusters and with a larger volume of data will be conducted in future research. In addition, experiments will be carried out to find out what is the ideal size for the image blocks on the basis of to the cluster configuration.

References

Almeer, M. H. (2012). Cloud hadoop map reduce for remote sensing image analysis. *Journal of Emerging Trends in Computing and Information Sciences*, 3(4):637–644.

Carlson, T. N. and Ripley, D. A. (1997). On the relation between ndvi, fractional vegetation cover, and leaf area index. *Remote sensing of Environment*, 62(3):241–252.

da Silva Ferreira, R., Oliveira, D. A. B., Happ, P. N., da Costa, G. A. O. P., Feitosa, R. Q., and Bentes, C. (2015). Interimage cloud platform: Em direção à arquitetura de uma plataforma distribuída e de código aberto para a interpretação automática de imagens baseada em conhecimento. *XVII Simpósio Brasileiro de Sensoriamento Remoto - SBSR*, pages 5264–5271.

de Oliveira, M. G. and de Souza Baptista, C. (2012). Geostat-a system for visualization, analysis and clustering of distributed spatiotemporal data. In *GeoInfo*, pages 108–119.

Ferreira, K. R., de Oliveira, A. G., Monteiro, A. M. V., and de Almeida, D. B. (2015). Temporal gis and spatiotemporal data sources. In *GeoInfo*, pages 1–13.

Finkel, R. A. and Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9.

Fu, G., Zhao, H., Li, C., and Shi, L. (2013). Segmentation for high-resolution optical remote sensing imagery using improved quadtree and region adjacency graph technique. *Remote sensing*, 5(7):3259–3279.

Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., and Lewin, D. (1997). Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663. ACM.

Lin, F.-C., Chung, L.-K., Wang, C.-J., Ku, W.-Y., and Chou, T.-Y. (2013). Storage and processing of massive remote sensing images using a novel cloud computing platform. *GIScience & Remote Sensing*, 50(3):322–336.

Lucene, A. (2010). Apache lucene.

Lv, Z., Hu, Y., Zhong, H., Wu, J., Li, B., and Zhao, H. (2010). Parallel k-means clustering of remote sensing images based on mapreduce. In *Proceedings of the 2010 International Conference on Web Information Systems and Mining*, WISM'10, pages 162–170, Berlin, Heidelberg. Springer-Verlag.

Morton, D. C., DeFries, R. S., Shimabukuro, Y. E., Anderson, L. O., Del Bon Espírito-Santo, F., Hansen, M., and Carroll, M. (2005). Rapid assessment of annual deforestation in the brazilian amazon using modis data. *Earth Interactions*, 9(8):1–22.

Neagoe, V., Ciurea, A., Bruzzone, L., and Bovolo, F. (2014). A novel neural approach for unsupervised change detection using som clustering for pseudo-training set selection followed by csom classifier. In *Geoscience and Remote Sensing Symposium (IGARSS), 2014 IEEE International*, pages 1437–1440. IEEE.

Neves, A. K., Bendini, H. N., Körting, T. S., and Fonseca, L. M. G. (2015). Combining time series features and data mining to detect land cover patterns: a case study in northern mato grosso state, brazil. In *GeoInfo*, pages 174–185.

Plaza, A. J. and Chang, C.-I. (2007). *High performance computing in remote sensing*. CRC Press.

Rathore, M. M. U., Paul, A., Ahmad, A., Chen, B.-W., Huang, B., and Ji, W. Real-time big data analytical architecture for remote sensing application.

Romani, L. A., de Ávila, A. M. H., Zullo Jr, J., Traina Jr, C., and Traina, A. J. (2009). Mining climate and remote sensing time series to discover the most relevant climate patterns. In *SBBD*, pages 181–195.

Smits, P. and Bruzzone, L. (2004). *Analysis of multi-temporal remote sensing images*, volume 3. World Scientific.

Song, W., Jin, B., Li, S., Wei, X., Li, D., and Hu, F. (2015). Building spatiotemporal cloud platform for supporting gis application. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:55–62.

Subramaniyan, R., Raman, P., George, A. D., and Radlinski, M. (2006). Gems: Gossip-enabled monitoring service for scalable heterogeneous distributed systems. *Cluster Computing*, 9(1):101–120.

Van Den Bergh, F., Wessels, K. J., Miteff, S., Van Zyl, T. L., Gazendam, A. D., and Bachoo, A. K. (2012). Hitempo: a platform for time-series analysis of remote-sensing satellite data in a high-performance computing environment. *International journal of remote sensing*, 33(15):4720–4740.

Wang, G., He, G., and Liu, J. (2012). A new classification method for high spatial resolution remote sensing image based on mapping mechanism. In *Proceedings of the International Conference on Geographic Object-Based Image Analysis (GEOBIAâ12)*, pages 186–190.